

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Experiment Management Wafer and Wafer Group Graph

Tiago Duarte Carvalho



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Nuno Honório Rodrigues Flores

July 25, 2020



# **Experiment Management Wafer and Wafer Group Graph**

**Tiago Duarte Carvalho**

Mestrado Integrado em Engenharia Informática e Computação

July 25, 2020



# Abstract

In the 21<sup>st</sup> century, Industry 4.0 emerged as a promising approach to face global competition. However, its requirements demand profound changes in manufacturing technology, and the update of manufacturing execution systems (MES) used for more than two decades to manage and monitor the plant floor. Critical Manufacturing (CMF) is one of the companies making major contributions to the manufacturing world and MES. CMF MES was recently improved, and a module for the Design of Experiment (DoE) was added. DoE is a powerful technique, extensively used for the optimisation of almost all types of manufacturing processes and in product and process design and development. One of these manufacturing processes is semiconductor manufacturing, namely the wafer fabrication, which is the most expensive and time-consuming step of this manufacturing process. Even though the integration of DoE in CMF MES was a valuable addition, facilitating its application to such complex environments, because of its intricacy and the need to define wafer group level variations at different process steps, human errors occur. The main goal of this dissertation was the implementation of graph capabilities into the DoE module, to prevent logically invalid and or incorrectly designed experiments to be executed. The existing structure was a tabular one, and graphs and graph drawings are recognized as a good way of storing and presenting large quantities of information where the tabular view is hardly human-readable. Information about flow steps was stored in nodes, and about sub-materials was stored in nodes and in links. Graph representation was done with an open source library that was already integrated in CMF MES. Logical validity verification was done with a function that traversed the graphs using a recursively depth-first approach. Resorting to graphs to address logically invalid experiments, allowed the use of graph drawings in a more intuitive additional interface, facilitating users' identification of incorrectly designed experiments. The graph capabilities also included features for user interaction. Several tests with different types of experiments were performed and all existing errors were detected, thus validating the implementation of graph capabilities performed into the DoE module. Although this validation was done for DoE of wafer fabrication, the solution is not restricted to it, having the potential to be used in other manufacturing processes.

**Keywords:** Design of Experiments, Manufacturing Execution System, Semiconductor Manufacturing, Graph Theory, Graphs Drawings



# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                | <b>1</b>  |
| 1.1      | Context . . . . .                                  | 1         |
| 1.2      | Motivation . . . . .                               | 3         |
| 1.3      | Objectives . . . . .                               | 5         |
| 1.4      | Document Structure . . . . .                       | 5         |
| <b>2</b> | <b>Graph theory and algorithms</b>                 | <b>7</b>  |
| 2.1      | Basic Notions . . . . .                            | 7         |
| 2.2      | Trees . . . . .                                    | 10        |
| 2.3      | Directed graphs . . . . .                          | 12        |
| 2.4      | Graph Traversal . . . . .                          | 14        |
| 2.5      | Summary . . . . .                                  | 15        |
| <b>3</b> | <b>Graph drawings</b>                              | <b>17</b> |
| 3.1      | Criteria . . . . .                                 | 17        |
| 3.2      | Methods and Techniques . . . . .                   | 19        |
| 3.3      | Summary . . . . .                                  | 25        |
| <b>4</b> | <b>Wafer fabrication and DoE</b>                   | <b>27</b> |
| 4.1      | Wafer Fabrication . . . . .                        | 27        |
| 4.2      | Wafer DoE . . . . .                                | 29        |
| 4.3      | CMF Experiment Management Module . . . . .         | 31        |
| 4.4      | Summary . . . . .                                  | 38        |
| <b>5</b> | <b>Solution development and methodology</b>        | <b>39</b> |
| 5.1      | User stories and development methodology . . . . . | 39        |
| 5.2      | Preparation and Graph Visualization . . . . .      | 41        |
| 5.3      | Logical Verification . . . . .                     | 49        |
| 5.4      | Improvement of User Features . . . . .             | 50        |
| 5.5      | Testing and Validation . . . . .                   | 51        |
| 5.6      | Summary . . . . .                                  | 53        |
| <b>6</b> | <b>Validation and results analysis</b>             | <b>55</b> |
| 6.1      | Graph drawing of a complex experiment . . . . .    | 55        |
| 6.2      | Application of user features . . . . .             | 57        |
| 6.3      | Detection and visualization of errors . . . . .    | 61        |
| 6.4      | Summary . . . . .                                  | 68        |

|          |                              |            |
|----------|------------------------------|------------|
| <b>7</b> | <b>Conclusions</b>           | <b>69</b>  |
| 7.1      | Main contributions . . . . . | 70         |
| 7.2      | Future work . . . . .        | 70         |
| <b>A</b> | <b>Source code</b>           | <b>73</b>  |
|          | <b>References</b>            | <b>101</b> |



# List of Figures

|      |  |    |
|------|--|----|
| 2.1  | Example of a graph with nodes, links and a self-loop . . . . .   | 8  |
| 2.2  | Example of a simple graph . . . . .  | 8  |
| 2.3  | Example of digraphs ( $G1$ and $G2$ ) and of a simple digraph ( $G2$ ) . . . . .   | 8  |
| 2.4  | Example of a spanning subgraph ( $H1$ ) and of an induced subgraph ( $H2$ ) of a graph $G$ (adapted from [38]) . . . . .   | 9  |
| 2.5  | Example of a walk of a graph $G$ that is not a trail (nor a path) . . . . .  | 9  |
| 2.6  | Example of a walk of a graph $G$ that is a trail but not a path . . . . .  | 9  |
| 2.7  | Example of a walk of a graph $G$ that is a trail and a path . . . . .  | 10 |
| 2.8  | Example of a disconnected graph and its three connected components $G1$ , $G2$ and $G3$ (adapted from [33]) . . . . .  | 10 |
| 2.9  | Examples of non-tree and tree graphs (adapted from [38]) . . . . .   | 10 |
| 2.10 | Examples of trees component of a forest graph (adapted from [33]) . . . . .  | 11 |
| 2.11 | Example of a spanning tree of a graph (adapted from [33]) . . . . .  | 11 |
| 2.12 | Example of a digraph which is not strongly connected (a) and of its three strongly connected components (b) (adapted from [33]) . . . . .                                  | 12 |
| 2.13 | Example of a complete 3-ary tree ( $G1$ ) and of an incomplete 3-ary tree ( $G2$ ) (adapted from [61]) . . . . .   | 14 |
| 3.1  | Drawings of the same graph/digraph: (a) polyline; (d) straight-line; (c) orthogonal; (d) polyline grid; (e) planar polyline; (f) upward planar polyline (adapted from [7]) | 18 |
| 3.2  | Example of a non-planar drawing and of a planar drawing (adapted from [53]) . .  | 19 |
| 3.3  | Example of the spring analogy (adapted from [10]) . . . . .  | 20 |
| 3.4  | Example of a non-layered drawing and of a possible corresponding layered drawing (adapted from [6]) . . . . .  | 20 |
| 3.5  | Example of the layering process of a non-layered graph drawing (adapted from [40]))  | 21 |
| 3.6  | Examples of orthogonal drawings with bends and no bends (adapted from [27]) .  | 22 |
| 3.7  | Example of a graph drawing $G1$ that does not maintain the user's mental picture of the graph (adapted from [7]) . . . . .   | 22 |
| 3.8  | Example of changes made to a graph drawing being visualized in a timeline (adapted from [8]) . . . . .   | 23 |
| 3.9  | Example of possible positions for labelling a node (on the left) and for labelling a link (on the right) (adapted from [45]) . . . . .                                     | 24 |
| 3.10 | Example of a generic labelling and the corresponding integrated labelling (adapted from [83]) . . . . .  | 25 |
| 4.1  | Wafers at the national Museum of Scotland ([64] under the Creative Commons Attribution-Share Alike 3.0 Unported license) . . . . .   | 27 |

|      |   |    |
|------|---|----|
| 4.2  | A silicon ingot at the Intel Museum ([3] under the Creative Commons Attribution-Share Alike 3.0 Unported license) . . . . .   | 28 |
| 4.3  | A silicon wafer with 30 cm after the depositing of all the layers and before dies are cut (adapted from [69] under the Creative Commons Attribution-Share Alike 3.0 Unported license) . . . . . | 28 |
| 4.4  | Accessing CMF EM module in CMF MES (orange highlight) . . . . .   | 31 |
| 4.5  | Accessing an experiment in CMF EM module's main menu (orange highlight) . . . . .   | 32 |
| 4.6  | Additional features in CMF EM module's main menu (orange highlight) . . . . .   | 32 |
| 4.7  | Creation of an experiment in the CMF EM module's (orange highlight) . . . . .   | 36 |
| 4.8  | Addition of steps to an experiment in the CMF EM module's (orange highlight) . . . . .  | 36 |
| 4.9  | Example of a step and respective actions and material groups in the CMF EM module's . . . . .   | 38 |
| 5.1  | JointJS with hardcoded information . . . . .  | 42 |
| 5.2  | Graph drawing with hardcoded information in JointJS . . . . .   | 42 |
| 5.3  | Code necessary to represent the Version 1 of the experiment . . . . .   | 44 |
| 5.4  | Code necessary to represent the Version 2 of the experiment . . . . .   | 45 |
| 5.5  | Graph drawing of the Version 1 of the experiment (only main flow) . . . . .   | 45 |
| 5.6  | Graph drawing of the Version 2 of the experiment (main flow and an action) . . . . .  | 46 |
| 5.7  | Graph drawing of the Version 3 of the experiment (addition of an action causing a split) . . . . .  | 46 |
| 5.8  | Graph drawing of the Version 4 of the experiment (addition of an action causing a merge back from the split) . . . . .  | 47 |
| 5.9  | Graph drawing of the Version 5 of the experiment (addition of an action Terminate) . . . . .  | 48 |
| 5.10 | Graph drawing of the Version 6 of the experiment (addition of an action TemporaryOffFlow) . . . . .   | 48 |
| 5.11 | Graph drawing of the experiment with a missing sub-material error . . . . .   | 52 |
| 5.12 | Graph drawing of the experiment with a missing sub-material error filtered by the by the sub-material causing the error . . . . .   | 52 |
| 5.13 | Graph drawing of the experiment with a circular reference error . . . . .   | 53 |
| 5.14 | Graph drawing of the experiment with a circular reference error filtered by the sub-material causing the error . . . . .  | 53 |
| 6.1  | Graph drawing of the complex experiment . . . . .   | 57 |
| 6.2  | Graph drawing of the complex experiment with the unused nodes hidden . . . . .  | 58 |
| 6.3  | Graph drawing of the complex experiment with nodes position changed by the user . . . . .   | 58 |
| 6.4  | Graph drawing of the complex experiment filtered by sub-material 1 . . . . .  | 59 |
| 6.5  | Graph drawing of the complex experiment filtered by sub-material 3 . . . . .  | 59 |
| 6.6  | Graph drawing of the complex experiment filtered by sub-material 4 . . . . .  | 60 |
| 6.7  | Graph drawing of the complex experiment filtered by a group of sub-materials (3 and 4) . . . . .  | 60 |
| 6.8  | Graph drawing of the complex experiment filtered by sub-material 5 . . . . .  | 61 |
| 6.9  | Graph drawing of the complex experiment with missing sub-material errors . . . . .  | 63 |
| 6.10 | Graph drawing of the complex experiment with missing sub-material errors being filtered by the sub-material causing the errors (sub-material 1) . . . . .                                       | 64 |
| 6.11 | Graph drawing of the complex experiment with circular reference errors and respective missing sub-material errors . . . . .   | 66 |

|      |  |    |
|------|--|----|
| 6.12 | Graph drawing of the complex experiment with circular reference errors and respective missing sub-material error filtered by the sub-materials causing the errors (sub-material 4) . . . . . | 67 |
| 6.13 | Graph drawing of the complex experiment with circular reference errors filtered by the sub-material causing the errors (sub-material 2) . . . . .  | 67 |



# List of Tables

|     |   |    |
|-----|---|----|
| 4.1 | Description and illustration of stage 1 (Change set) of the experiments' definition                               | 33 |
| 4.2 | Description and illustration of part 1 of stage 2 (General data, information) of the experiments' definition      | 33 |
| 4.3 | Description and illustration of part 2 of stage 2 (General data, settings) of the experiments' definition         | 34 |
| 4.4 | Description and illustration of part 3 of stage 2 (General data, options) of the experiments' definition          | 35 |
| 4.5 | Description and illustration of stage 3 of the experiments' definition (Objectives)                               | 35 |
| 4.6 | Description and illustration of stage 4 of the experiments' definition (Material Groups)                          | 35 |
| 4.7 | Description and illustration of stage 1 (General data) of adding a step to a defined experiment                   | 36 |
| 4.8 | Description and illustration of stage 2 (Material groups) of adding a step to a defined experiment                | 37 |
| 4.9 | Description and illustration of stage 3 (Actions) of adding a step to a defined experiment                        | 37 |
| 5.1 | Node Data Structure   | 49 |
| 6.1 | Characteristics of the complex experiment   | 56 |
| 6.2 | Complex experiment with additional actions causing missing sub-material errors                                    | 62 |
| 6.3 | Complex experiment with additional actions causing circular reference and respective missing sub-materials errors | 65 |



# Abbreviations

|       |                                   |
|-------|-----------------------------------|
| API   | Application Programming Interface |
| ASMPT | ASM Pacific Technology Limited    |
| BFS   | Breath-First Search               |
| CMF   | Critical Manufacturing            |
| CPS   | Cyber-Physical Systems            |
| DFS   | Depth-First Search                |
| DoE   | Design of Experiments             |
| DAG   | Directed Acyclic Graph            |
| EM    | Experiments Management            |
| GUI   | Graphical User Interface          |
| IC    | Integrated Circuits               |
| IIoT  | Industrial Internet of Things     |
| IoT   | Internet of Things                |
| IVR   | Interactive Voice Response        |
| MES   | Manufacturing Execution System    |





# Chapter 1

## Introduction

This chapter addresses the importance of the project as well as this document's structure. Section 1.1 describes the area in which this project is inserted, as well as related topics and overall context. Section 1.2 presents the problem and its framework, justifies its relevance and indicates the solution to be implemented and why. The section 1.3 refers to the goal of this project, and the specific objectives that have enabled it to be achieved. Section 1.4 includes a brief description of the structure of this document.

### 1.1 Context

At the end of the 20<sup>th</sup> century, industries faced the challenge of dealing with the technological innovations motivated by the change from analogue to digital [81]. This evolved, in the 21<sup>st</sup> century, to a new paradigm, in which manufacturing is digitized and connected, known as Industry 4.0 [63]. Zhou et. al [84, p. 2149] describe it as “a complex and flexible system involving digital manufacturing technology, network communication technology, computer technology, automation technology and many other areas”, and Muller et al. [63, p. 247] point out it enabling of “real-time-capable horizontal and vertical Internet-based connectedness of people, machines, and objects, as well as information and communication technologies for the dynamic management of complex business processes”.

The term “Industry 4.0”, which was first used in Germany in 2011, “has been widely discussed, and has become a hotspot for most global industries” [84, p. 2147]. Similar technology-related initiatives emerged in several countries around the world [81], making it “one of the important topics in the realm of manufacturing” [52, p. 3624]. In Portugal, the Government recognizes that there is a lot to be done because 75% of the companies are unaware of this new industrial revolution [31]. To do this, in 2019, the second phase of the Government Industry 4.0 Program was launched, being expected the mobilization of public and private investments worth 600 million euros in the next two years [49].

According to Rojko [72], although Industry 4.0 is a promising approach to face nowadays global competition and market requests that continuously change, its requirements can only be met if the current manufacturing technology is radically improved. For the author [72, p. 77], “technical aspects of these requirements are addressed by the application of the generic concepts of Cyber-Physical Systems (CPS) and industrial Internet of Things (IoT) to the industrial production systems”.

The industrial Internet of Things (IIoT) “builds on not only the automation but also the information systems already in place to drive new insights by applying sensors and analytics and connecting the smart devices [which is the] foundation for Industry 4.0, Smart Manufacturing and Digitalization” [23, p. 4]. Because of that, the transition to the IIoT will require manufacturers to update their manufacturing execution systems (MES) [50], with some believing that “it will only succeed as a marriage with MES” [23, p. 2].

Manufacturing execution systems have been used for more than two decades to “deliver information that enables the optimisation of production activities from order launch to finished goods” [25, p. 526], by managing and monitoring the work that is been done on the plant floor [50]. So, the majority of MES are pre-IIoT, and are not prepared to make a smooth transition to the new paradigm brought by Industry 4.0. Although some argue that this shows that MES “is dead” [50], others consider that “IIoT boosts rather than destroys the need for MES” [23, p. 6] even though this means that a new MES will be needed [23].

In 2019, to evaluate the importance of MES to Industry 4.0, Mantravadi et al. [57, p. 594] conducted a systematic literature review and concluded that resorting to a “single advanced MES software has the potential to help the manufacturing enterprises achieve greater degree of operational visibility and traceability”. The authors [57, p. 594] argue that MES “play a central role in the company’s path towards Industry 4.0 because flexibility and transparency in the processes are the key objectives of Industry 4.0”. They suggest that future research about MES should focus on its potential to enhance the process performance, resulting in the optimization of manufacturing operations.

According to De Ugarte [25, p. 535], successful “MES solutions need to combine different solving tools with multiple data sources to elaborate solutions in reasonable time”, and incorporate recent advances in computing. The expectation for the new MES is that it would be able to provide real-time information to the operational departments about all resources involved in the production, benefiting from the advances in integration levels and computing technologies [57]. Because this real-time information is available, allowing the monitoring, controlling and operating of production processes, MES solutions have now the capability to optimize the production environment [2].

## 1.2 Motivation

The importance of MES in guiding companies towards Industry 4.0 [57], and its features in production optimization, justify the quite few solutions commercially available [36]. According to Manufacturing Technology Insights Magazine [60, para. 2], “several solutions providers are set to tackle the evolving challenges in the manufacturing industry (...) [promising] a significant rise in production efficiency”. Every year, the magazine compiles a list of Top 10 MES Solution Providers to highlight the companies making significant contributions to the manufacturing world [60]. In the second place of the 2019 edition (and of 2018), it is the Portuguese company Critical Manufacturing, S.A., founded in 2009 [19].

With its main headquarters in Porto (Maia), Portugal, “subsidiaries in Dresden, Germany, Suzhou, China, Austin, USA, and a branch office in Taiwan” [59, para. 2], Critical Manufacturing provides “state-of-the-art products and services” [58, para. 2], which includes its manufacturing execution intelligence system and related services and expertise in semiconductor and electronics manufacturing industries [20]. In August 2018, it joined the ASM Pacific Technology Limited (ASMPT), which is a “global technology and market leader (...) in solutions and materials for the semiconductor assembly and packaging industries” [76, para. 5]. ASMPT announced this integration as a strategic investment, recognizing the “significant expertise in the integration and networking of machines and systems, as well as IT systems and cloud solutions” of Critical Manufacturing [76, para. 1].

Critical Manufacturing (CMF) presents its MES as “the most modern and complete modular MES available” [21, p. 1], being “comprehensive and modular, deep and radically configurable (...) [, supporting] current and new processes, products and employees in ways unimaginable for users of older products” [17, para. 4] and offering “not only advanced analytics and continuous improvement tools, but also a manufacturing digital twin, and quick, intuitive looks at performance” [17, para. 3]. In 2018, Rodrigues [71] compared eight commercially available MES solutions, concluding that the CMF MES is the only (of the eight) that combines relevant features like connectivity, cloud, and mobile.

CMF modern MES modules are organized in seven functional areas: visibility and intelligence, scheduling, factory management, quality management, operational efficiency, enterprise integration, and factory automation [17]. The latest CMF MES version (V7) includes some new modules: augmented reality and BI cards (in the visibility and intelligence area), experiments management (in the quality management area), and weigh and dispense (in the factory automation area) [17]. The experiments management, in particular, is a breakthrough module in the achievement of optimum process performance. The module allows workers to perform Design of Experiments (DoE) with multiple input variables, and execute and monitor experiments seamlessly in a single system, which results in an easier, more efficient and effective process [17].

Giles Jr et al. define DoE as “a systematic approach to understanding how process and product parameters affect response variables such as processability, physical properties, or product performance” [37, p. 231]. As far as process optimization is concerned, DoE is one of its powerful

techniques, which “has been widely deployed in almost all types of manufacturing processes and is used extensively in product and process design and development” [4, p. 1]. Engineers across industries resort to DoE in a variety of diverse activities, “ranging from developing new products to improving product designs to controlling and optimizing manufacturing processes” [18, para. 2].

One of the industries that resorts do DoE is the semiconductor manufacturing [73]. This industry “is among the most complex manufacturing today, in which a lengthy and re-entrant process steps are employed with advanced tools to fabricate a variety of Integrated Circuits (IC)” [14, p. 961]. These IC are made of thin slices of semiconductor materials that are known as wafers [35]. Although DoE is a cost-effective approach when there are multiple factors simultaneously, Chien et al. [14] consider that applying it to semiconductor manufacturing is a difficult task. The reasons they associated with this are the high manufacturing costs (time and money) involved, the little time available in the production line to conduct experiments, and the manufacturing process itself being very complex. This complexity is in part due to the process not following a linear flow and being constantly split and merged into different lots [22]. Because of the large number of input variables and step variations to be defined while designing an experiment, this generates matrices with several hundred cells that need to be filled by the person designing the experiment [73].

The integration of DoE in MES is a valuable addition, and its Experiments Management (EM) facilitates DoE in complex environments, such as the semiconductor manufacturing and wafer production. However, because of its intricacy, human errors occur during the DoE process. These errors result in the waste of valuable production time and resources, due to the execution of: 1) Logically invalid experiments, that do not make logical sense, and may not be executed properly, or even executed at all; 2) Incorrect experiments, that having logical sense, do not correspond to what was intended.

Regarding the logically invalid experiments, the current CMF EM module does not provide the user with the necessary feedback to prevent its execution. This is mainly due to the information being stored solely in tables in a database, that need multiple time-consuming joins to reproduce all the relations and processes involved in the designed experiments. Because the problem is associated with the relations, one way to tackle it is to have an additional information structure that facilitates finding these relations and processes, like graphs. Gross and Yellen [38, p. 2] define graph as “any mathematical object involving points and connections between them” and graph drawing as the “geometric representation of graphs” [38, p. 1015]. According to Loh et al. [55], graphs are used to represent networks, i.e., systems with different elements or components connected.

As for the incorrect experiments, a user-friendly way to verify them does not exist in the CMF EM module. The reason for this is the fact that the module’s current interface was designed to resemble the previous DoE tools, to facilitate users’ adaptation. Resorting to graphs to address the problem of logically invalid experiments, will allow the use of graph drawings in the development of an additional interface, which will provide a more intuitive visualization of the designed experiment.

The implementation of graphs and graph diagrams in the EM module has the potential to facilitate and improve DoE, contributing to expand CMF MES features in process optimization, making it even a more valuable tool for companies towards Industry 4.0.

### 1.3 Objectives

The main goal is the implementation of graph capabilities into the CMF EM module, to prevent logically invalid and or incorrectly designed experiments to be executed. To attain this general objective, some specific objectives were defined.

- To save the data resorting to graphs as a structure, using its properties to facilitate and speed up the process of detecting logical errors.
- To visually represent the graphs, using graph drawings, namely a hierarchical layout because it is the one that best resembles the wafer fabrication process flow.
- To have the information on the nodes necessary to be easily filtered and to have a function that traverses all nodes and compares their information to the information by which it is filtering, presenting the intended result.
- To implement the additional features needed to facilitate user interaction with the GUI.

### 1.4 Document Structure

Chapter 1 is the introduction to this dissertation and includes the context and motivation for the problem, the objectives, and the structure of this document. Chapter 2 describes relevant aspects of graph theory and algorithms, namely basic notions, trees, directed graphs, and graph traversal. In Chapter 3, criteria, methods, and techniques of graph drawings are explained. Chapter 4 focuses on wafer fabrication and DoE, which includes a detailed description of the CMF Experiment Management Module. Chapter 5 presents the user stories, the solution development methodology and the several stages of the development of the solution, which are graph visualization, logical verification, improvement of user features and testing and validation. The validation and analysis of the results is in Chapter 6, and includes a graph drawing of a complex experiment, the application of the implemented user interaction features to that graph drawing and an illustration of the detection and visualization of errors also in the mentioned graph drawing. Chapter 7 holds the conclusions of the dissertation, resuming the problem, the objectives and the development methodology and presenting the main contributions and future work.



## Chapter 2

# Graph theory and algorithms

This chapter describes aspects of graph theory and algorithms, taking into consideration what is relevant for the implemented solution. Section 2.1 presents general concepts and terminology of graph theory. Section 2.2 addresses notions about the type of graphs known as trees. Section 2.3 includes notions about directed graphs, emphasizing directed acyclic graphs. Section 2.4 is dedicated to graph traversal, focusing on the depth-first search algorithm. Section 2.5 summarizes the highlights of this chapter.

### 2.1 Basic Notions

As stated before, in this dissertation graphs are used to tackle the problem of rapidly finding relations and processes regarding designed experiments for wafer manufacturing, also allowing its visualization. Because of that, in this chapter, some elements of graph theory are presented.

Graphs are mathematical objects consisting of two sets of elements: vertices or nodes and edges or links (in this document, the notation used is nodes and links). Each link connects two nodes, that are said to be its endpoints [38], [33]. A link that joins a single endpoint to itself is a self-loop [38]. The degree of a node in a graph is the number of links that have that node as its endpoint. For example, in Figure 2.1, link  $a$  connects nodes  $u$  and  $v$ . Link  $e$  is a self-loop. The degree of node  $u$  is one, the degree of node  $y$  is two and the degree of node  $z$  is three. The notation for graphs is  $G = (N, L)$ , where  $N$  represent nodes (in Figure 2.1,  $u, v, x, y$  and  $z$  are nodes) and  $L$  links (in Figure 2.1,  $a, b, c$  and  $d$  are links) [77].

A practical way to draw a graph in a plane is to use dots or circles to represent nodes and lines for links (see Figure 2.1) [33]. Dots and lines can be placed in many ways in the plane, meaning that graphs can have several possible representations [77]. The curvature and length of the lines has no particular meaning [38].

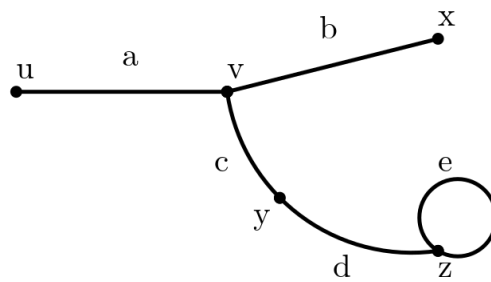


Figure 2.1: Example of a graph with nodes, links and a self-loop

There are several types of graphs, based on the number of nodes or links. When a graph has two distinct nodes and only one link connecting them it is a simple graph (see Figure 2.2) [38], [77].

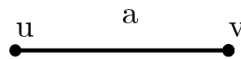


Figure 2.2: Example of a simple graph

Besides having nodes and links, graphs can have additional attributes. One of these attributes that is relevant for manufacturing processes is link direction. A link with direction is a directed link (see  $a$  and  $b$  in Figure 2.3), and a graph with a directed link is a directed graph or digraph (see  $G1$  and  $G2$  in Figure 2.3) [38].

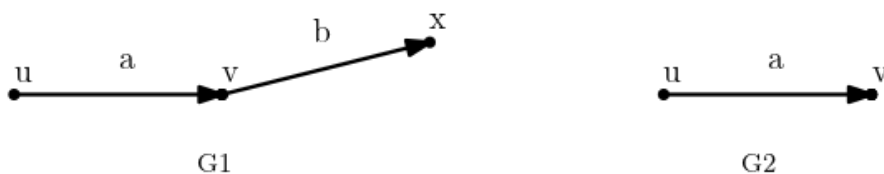


Figure 2.3: Example of digraphs ( $G1$  and  $G2$ ) and of a simple digraph ( $G2$ )

When all the nodes and links of a graph  $H$  are contained in the set of nodes and links of another graph  $G$ ,  $H$  is a subgraph of  $G$ . A subgraph is a spanning subgraph if it includes all the nodes of the graph (see  $H1$  in Figure 2.4, which is a spanning subgraph of graph  $G$ ). A subgraph is an induced subgraph on a set of nodes if it includes all the links of the graph that have the nodes of the subgraph as its endpoints (see  $H2$  on Figure 2.4, that is an induced subgraph of graph  $G$ ) [38], [77].



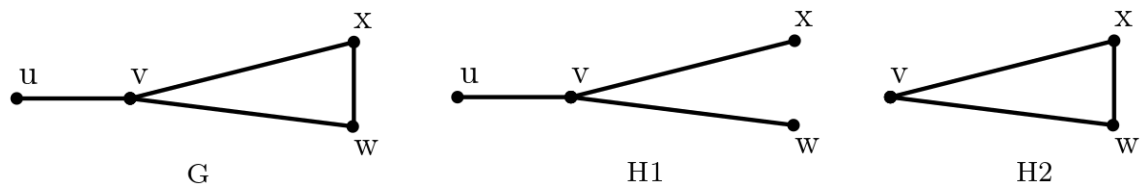


Figure 2.4: Example of a spanning subgraph ( $H1$ ) and of an induced subgraph ( $H2$ ) of a graph  $G$  (adapted from [38])

In a graph, a walk is a finite alternating sequence of nodes and links, that in a simple graph may be represented by just listing a sequence of nodes (see Figure 2.5). If the link is a directed link, then the walk is a directed walk. The first node of the walk is the initial node (see  $u$  in Figure 2.5), the last node of the walk is the final node (see  $y$  in Figure 2.5), and all the other nodes are internal nodes (see  $v$  and  $x$  in Figure 2.5). The number of links in a walk is the length of the walk (the length of the walk in Figure 2.5 is three). A walk can be closed (the initial node and the final node are the same) or open [38].

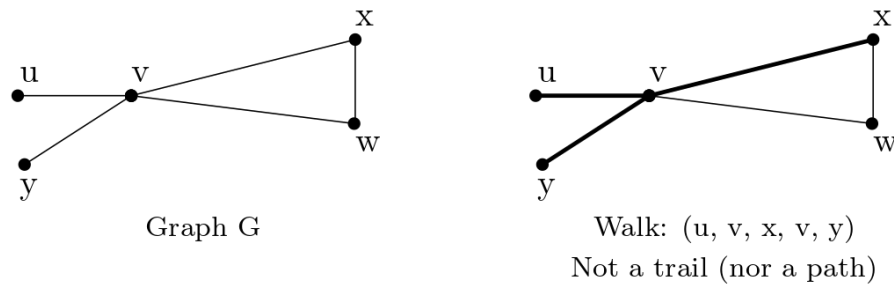


Figure 2.5: Example of a walk of a graph  $G$  that is not a trail (nor a path)

If in a walk, a link does not occur more than once, the walk is a trail (see Figure 2.6). When in a trail, an internal node is not repeated, the trail is a path (see Figure 2.7) [38]. This means that a trail is a walk where the links are all different, and the path is a walk where the links and nodes are all different [33].

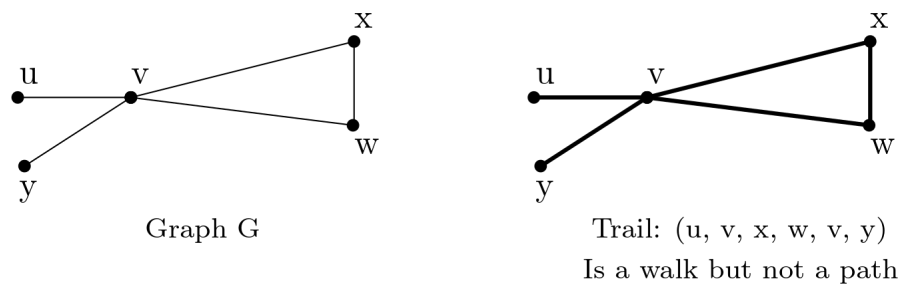


Figure 2.6: Example of a walk of a graph  $G$  that is a trail but not a path

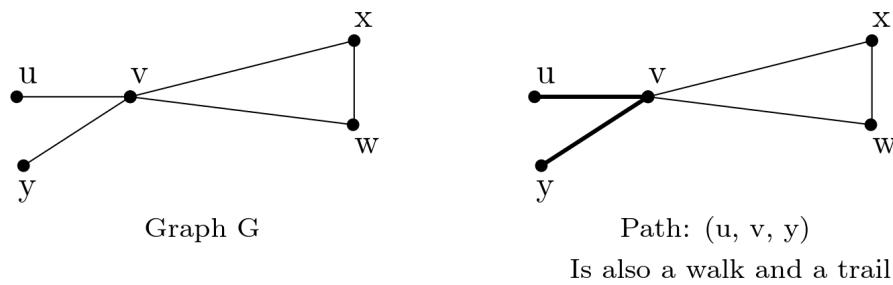


Figure 2.7: Example of a walk of a graph  $G$  that is a trail and a path

According to Thulasiraman [77, p. 31], “the graphs that are encountered in most of the applications are connected”. A graph is said to be a connected graph when any two of its nodes are linked by a path [33]. A graph that is not connected is a disconnected graph [33], [28]. A component of a graph is a maximal connected subgraph of the graph [38]. According to Fournier [33, p. 35], “the connected components of a graph are subgraphs pairwise disjoint, that is having pairwise no common vertices and no common edges”. These connected components define the unique decomposition of the graph (see Figure 2.8)).

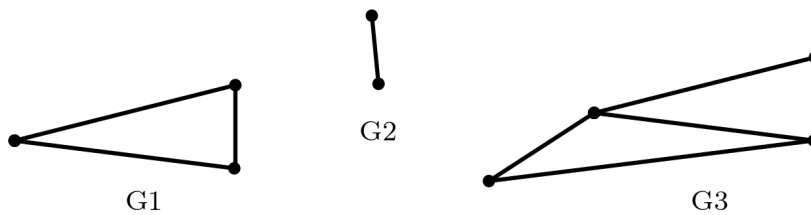


Figure 2.8: Example of a disconnected graph and its three connected components  $G_1$ ,  $G_2$  and  $G_3$  (adapted from [33])

## 2.2 Trees

Trees are a type of graph that Gross and Yellen [38, p. 15] consider “important to the structural understanding of graphs and to the algorithmics of information processing, and they play a central role in the design and analysis of connected networks”. For Estrada [28], the tree is the simplest type of graph. Fournier [33] defines it as a connected graph that has no closed paths, *i.e.*, cycles (see Figure 2.9), thus being a connected acyclic graph. When the links of a tree are directed, the resulting tree is a directed acyclic graph, known as a polytree [24].

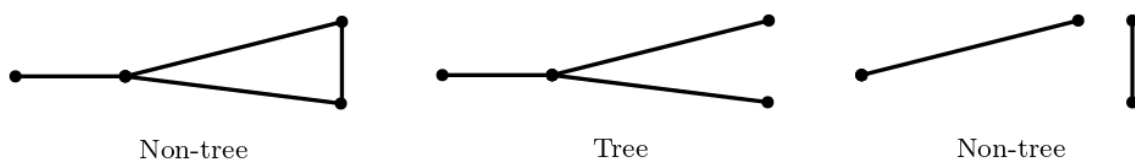
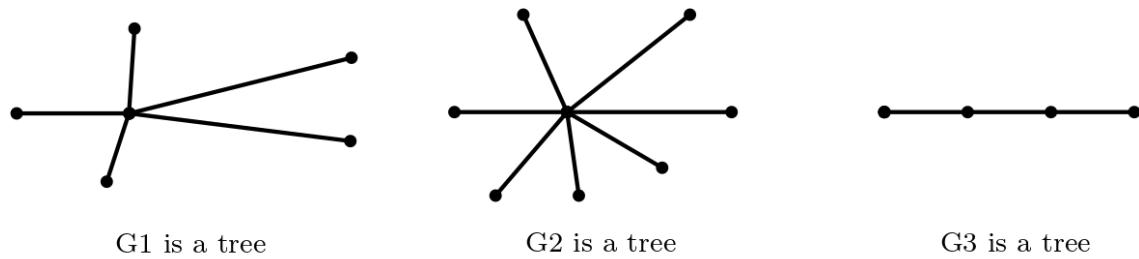


Figure 2.9: Examples of non-tree and tree graphs (adapted from [38])

Fournier [33] states the first properties of trees, where  $N$  is the number of nodes and  $L$  the number of links, as: 1) A tree with two or more nodes has at least two nodes of degree one; 2) If a graph is a tree, the number of links plus one equals the number of nodes ( $N = L + 1$ ); 3) Any two nodes of a tree are connected by a unique path.

A forest is a disconnected graph in which every connected component is a tree (see Figure 2.10) [28]. The forest is also an acyclic graph [38], [33]. In a forest, the number of links plus one is less than the number of nodes ( $N < L + 1$ ) [33].



G1, G2 and G3 are trees and components of graph  $G$ , which is a forest

Figure 2.10: Examples of trees component of a forest graph (adapted from [33])

A bridge of a graph  $G$  is a link whose absence turns a connected graph into a disconnected one. Eliminating the bridge in a graph creates one more connected component in the resulting graph [33]. Regarding bridges, Fournier [33, p. 48] states that “a link of a graph  $G$  is a bridge if and only if it does not belong to a cycle of  $G$ ” and “in a tree, any [link] is a bridge”.

A tree can also be a subgraph in a graph. For a given tree in a graph  $G$ , the nodes and links are called tree links and tree nodes, and the nodes and links that are not in the tree are called non-tree links and non-tree nodes [38].

A spanning subgraph of  $G$  which is a tree is a spanning tree of a graph  $G$  (in Figure 2.11, subgraph  $G1$  of graph  $G$  is a spanning tree) [33].

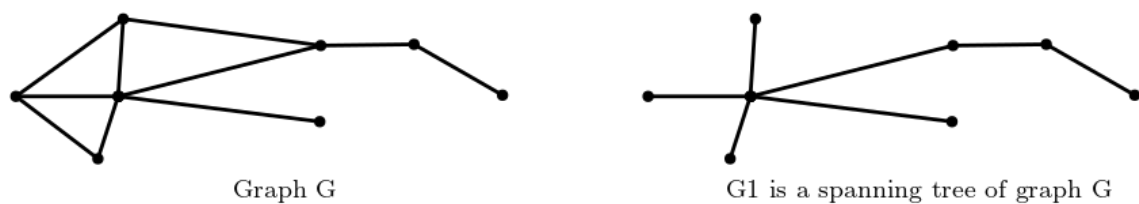


Figure 2.11: Example of a spanning tree of a graph (adapted from [33])

Fournier [33, p. 50] states some propositions regarding spanning subgraphs as “a connected graph has at least one spanning tree”, “a spanning subgraph of a connected graph  $G$  is a spanning tree of  $G$  if and only if it is connected and [link]-maximal”, and “a spanning subgraph of a connected graph  $G$  is a spanning tree of  $G$  if and only if it is acyclic and [link]-maximal”.

A spanning forest is a subgraph of the graph that includes every node and is a forest [28].

## 2.3 Directed graphs

Several situations that are not adequately represented by undirected graphs, because the direction of what is being represented is important [77]. As stated before, the direction is one of the attributes of links, and a graph with directed links is a directed graph or digraph.

Terminology associated with directed graphs is different according to the authors. For example, Gross and Yellen [38] refer to the nodes as the tail and the head, with the arrow pointing to the head, and Fournier [33] as successor and predecessor, with the link coming out of the predecessor and entering the successor.

It is possible to define a basis of a digraph, which is a minimal set of nodes such that every other node can be reached from some node in this minimal set by a directed path [38].

As with undirected graphs, digraphs can have self-loops or multiple links. When a digraph has two distinct nodes and only one directed link connecting them it is a simple digraph [38].

All directed graphs have an underlying graph, which is defined by not considering the direction of its links. Concepts like degree, connectedness, components, walk, trail, and path, all apply to directed graphs through its underlying graph [33]. Regarding the degree, the number of links directed to a node is the indegree of that node, and the number of links directed from a node is the outdegree [38]. As for connectedness, a directed connected graph is named a strongly connected graph [28]. A strongly connected component of a graph is a maximal strongly connected induced subdigraph of that graph [33]. According to Fournier, these strongly connected components are less simple to determine than the connected components of an undirected graph (see Figure 2.12).

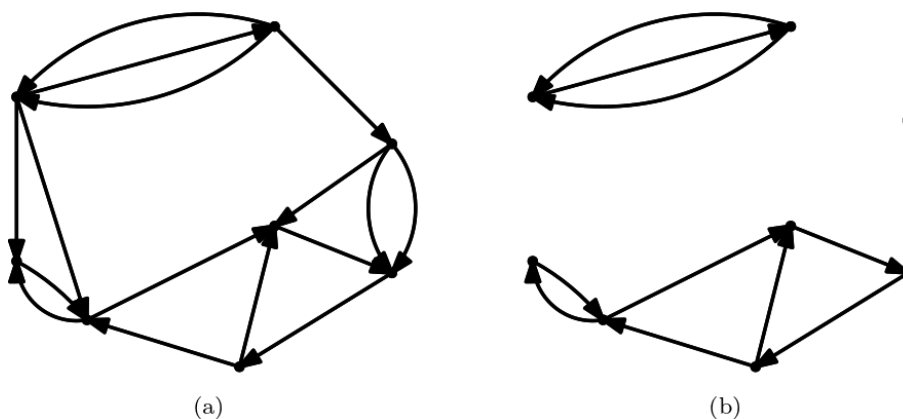


Figure 2.12: Example of a digraph which is not strongly connected (a) and of its three strongly connected components (b) (adapted from [33])

For a digraph, walks, trails, and paths are referred to as directed walks, directed trails, and directed paths. Cycles are directed cycles, that are known as circuits. A circuit is a closed directed path. When a digraph does not have circuits it is an acyclic digraph [33].

For Maurer [61, p. 142], acyclic digraphs or directed acyclic graphs (DAG) “arise quite naturally because [nodes] often have a natural ordering”, as nodes can represent events ordered in time

or by hierarchy. This is the case of wafer manufacturing and designed experiments to optimize its production process, which is the focus of this dissertation.

Some examples of the applicability of DAG include: 1) Operations research, as a large project involves a great number of tasks that precede each other, meaning that there are tasks that have to be completed before other tasks can start; 2) Sociology and socio-biology, as businesses, armies, societies, or ant colonies have a hierarchical dominance structure, where the nodes are the employees, soldiers, citizens or ants amongst which there is a dominance relation represented by the directed links; 3) Computer software design, as a large program groups several subprograms that can invoke each other, where nodes can represent the subprograms and directed links represent a subprogram invoking another subprogram; 4) Ecology, namely in food webs, where nodes represent species and links indicate which species eats another specie (food chain); 5) Genealogy, namely in family trees, even though there are no arrows, being implicit that later generations are down in the tree; 6) State diagrams, as the nodes can represent a set of states and links the possible changes between these states, preventing the states going back to the previous configuration [61].

In an acyclic digraph, there is a source and a sink. A source is a node with a zero indegree and a sink is a node with a zero outdegree [33].

When the underlying graph of a digraph is a tree, then this tree is a directed tree. A directed tree that has a root is known as a rooted tree [61] or an arborescence [33]. The root is a node such that the unique path from the root to another node is a directed path from the root to that node [61]. According to Maurer [61], if the root is marked in the graph drawing than the arrows are not included because the direction of each link is always from the root to the other nodes. The author [61] distinguishes out-tree from in-tree. The out-tree is another name for the rooted tree. In an in-tree, the direction of all the paths is toward the root.

Some terminology related to arborescence includes the notion of depth or level and height. The depth of a node is the distance from the root to this node, which is the number of links in the path from the root to the node. The height of a tree is the greatest depth of its nodes [61]. Nodes with the same depth are at the same depth level [33]. A layer of nodes is a subset of nodes that includes all the nodes that have the same depth [61].

Trees have also a part of its terminology that is influenced by genealogical trees [33]. The term child applied to a node means any successor of that node. Similarly, a predecessor node is the parent of a successor node. Nodes with the same parent are called siblings. A node without a child is a leaf and a node that is not a leaf is an internal node [61], [33].

Trees can be classified regarding the number of children that every node has. In a  $m$ -ary tree, every node has  $m$  or fewer children. A  $m$ -ary tree is complete when all its internal nodes have exactly  $m$  children and all the leaves are at the same level (see Figure 2.13).

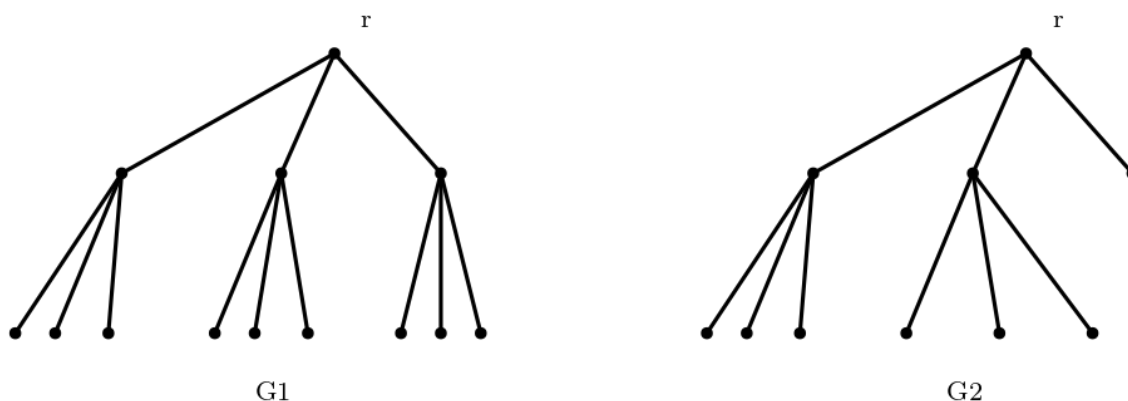


Figure 2.13: Example of a complete 3-ary tree ( $G1$ ) and of an incomplete 3-ary tree ( $G2$ ) (adapted from [61])

## 2.4 Graph Traversal

Graphs that are used to solve real-life problems are large and complex, and so, analysing them efficiently involves resorting to computer algorithms [77]. As part of solving these problems, it may be necessary to traverse the graph. The algorithms used to graph traversal are known as search algorithms. They are used when it is necessary to find or visit nodes or links of a graph [33]. Gabow [34, p. 953] defines search of a graph as “a methodical exploration of all the [nodes] and [links]” and Thulasiraman [77, p. 346] as “a systematic way for exploring a graph”.

According to Gabow [34], the two most important search methods are breath-first search (BFS) and depth-first search (DFS). The BFS is an efficient way to compute distances [34] and the DFS is useful for checking some properties of graphs [34] and for general graph traversal [12]. Chao [12] summarizes the differences between them as follows: 1) The BFS algorithm explores the graph one layer at a time, increasing the depth of the layer that is being explored; 2) The DFS algorithm travels along a path, from a starting node to some end node, repeating the search with another path from the same node. Because in this dissertation distances in graph drawings are not computed, only the DFS algorithm is described with more detail.

The DFS origin dates back to the 19<sup>th</sup> century when it was developed by Trémaux as a strategy for exploring a maze [34]. Imagining that the maze is the graph drawing, the DFS algorithm should “guarantee that the whole graph will be scanned without wandering too long in the maze and that the procedure will allow one to recognize when the task is done. However, before one starts walking in the maze, one does not know anything about its structure, and therefore, no preplanning is possible” [29, p. 46].

Thulasiraman [77] describes the procedure for the DFS as follows. A node  $v$  is chosen to start the search; this node  $v$  is the root of the DFS and is now visited. A link from node  $v$  is traversed to get to another node  $w$ ; the link  $(v, w)$  is said to examined. When at a node  $x$ , there are two possibilities: 1) If all the links from node  $x$  have already been examined, then the search continues from the parent of  $x$ ; node  $x$  is said to be completely scanned; 2) If there are some links from node

$x$  that have not been examined, then a link  $(x, y)$  is chosen and is now said to be examined. In this case, if  $y$  has not been visited, link  $(x, y)$  is traversed, node  $y$  is visited and the search continues from node  $y$ . If  $y$  has been previously visited, then another unexamined link from  $x$  is selected. When a node is visited for the first time, it is assigned an integer. These integers indicate the order in which nodes were visited. The DFS ends when the search returns to the root and all the nodes have been visited.

The DFS algorithm has some differences depending on whether the graph is undirected or directed. In directed graphs, a link can only be traversed along its orientation [77]. Because of this constraint, when at a node  $v$  of an unexamined link  $(v, w)$ , one of four categories can be assigned to this link: tree link, forward link, back link, cross link. Link  $(v, w)$  is a tree link if node  $w$  has not yet been visited. Link  $(v, w)$  is a forward link if node  $w$  has already been visited, and node  $w$  is a descendant of  $v$  in the DFS forest. Link  $(v, w)$  is a back link if node  $w$  has already been visited, and node  $w$  is an ancestor of  $v$  in the DFS forest. Link  $(v, w)$  is a cross link if its nodes are not related in the DFS forest and the integer assigned to  $w$  is smaller than the one assigned to  $v$ .

## 2.5 Summary

In this chapter, relevant aspects of graph theory and algorithms were presented. Of these, it is important to highlight that:

- Graphs are mathematical elements with nodes and links that can be represented by dots and lines, respectively. Graphs can have and/or be subgraphs, which in turn can be spanning subgraphs or induced subgraphs. Some graphs have walks, or trails or paths. Depending on the existence of paths, graphs can be connected or disconnected. Some graphs have components.
- Digraphs are directed graphs, *i.e.* graphs with directed links. Degree, connectedness, components, walk, trail, and path also apply to digraphs. Digraphs can have directed walks, directed trails, and directed paths. Directed cycles are called circuits. Directed acyclic graphs have no circuits.
- One of the simplest types of graphs is the tree. The tree is a connected acyclic graph (has no closed paths). Trees can be part of forests. Trees can also be subgraphs of graphs, which include spanning trees. Trees can be directed graphs. Trees with roots are rooted trees. Depth and height can be defined for rooted trees. Trees related terminology includes the term child, parent, and sibling.
- Graph traversal is made easier resorting to algorithms. The DFS algorithm, which is useful for checking some properties of graphs, traverses the graph along each of its paths. It can be used with indirect and directed graphs, with the proper adjustments.





## Chapter 3

# Graph drawings

This chapter is dedicated to graph drawings. Section 3.1 addresses issues related to graph representation and conventions, and the aesthetic quality of graph drawings. Section 3.2 describes methods and techniques for drawing graphs, giving more emphasis to those used in the implemented solution. Section 3.3 summarizes the highlights of this chapter.

### 3.1 Criteria

Graph drawings became popularized as a good way for presenting large quantities of information since the tabular view “is hardly human readable” [66, p. 1]. They “take advantage of the powerful human visual perception system, (...) [being] often used as a visual tool for the purposes of communication and understanding of non-visual graph data” [44, p. 444]. However, this does not mean that graphs can be drawn in any way, or that all graph drawings are good graph drawings [32], [44]. Nodes and links can be drawn in several ways, the same being true for the information corresponding to these nodes and labels. Different colours and thickness can be used, amongst a variety of other elements, and a two- or three-dimensions drawing can be done. Fleischer and Hirsch [32] call these drawing style considerations the representation of a graph.

According to Battista et al. [7], there are basic rules that graph drawings must satisfy to be admissible, *i.e.* there is a drawing convention for graphs. For the author [7], some of these widely used conventions are (see Figure 3.1): 1) Polyline drawing: links are drawn as polygonal chains; 2) Straight-line drawing: links are drawn as straight line segments; 3) Orthogonal drawing: links are drawn as polygonal chains of horizontal and vertical segments; 4) Grid drawing: nodes, crossings and link bends are integer coordinates; 5) Planar drawing: links do not cross; 6) Upward (or downward) drawing: for acyclic graphs, links are drawn facing up (or down) accordingly.

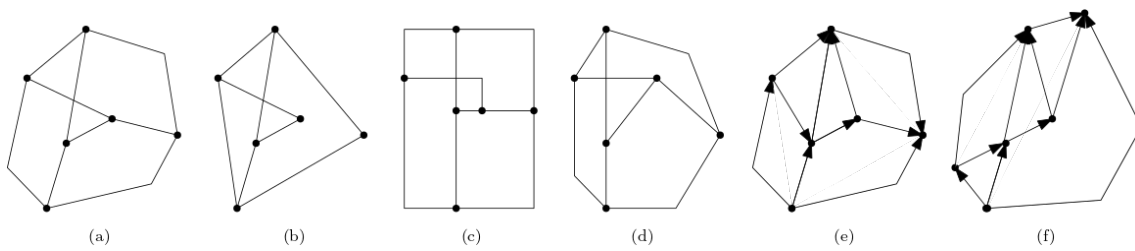


Figure 3.1: Drawings of the same graph/digraph: (a) polyline; (d) straight-line; (c) orthogonal; (d) polyline grid; (e) planar polyline; (f) upward planar polyline (adapted from [7])

Huang et al. [44] point out that a good layout facilitates the way a graph is perceived, while a poor layout may hinder the process. For Fleischer and Hirsch [32], a good layout for a graph drawing should take into account the structural properties of a graph (the type of graph), but also the knowledge about what the graph is representing. In graphs for human consumption, the aesthetics criteria should be taken into consideration, whereas in other purposes criteria like the technical criteria, might be more relevant.

Some of the commonly used aesthetics criteria are, according to Fleischer and Hirsch [32], the following: 1) Crossing minimization between links, to facilitate finding which nodes are connected to which links; 2) Bend minimization of links, because for the human eye straight lines are easier to follow; 3) Area minimization, in a way that the nodes and links are evenly distributed; 4) Angle maximization because having the links as far apart from each other as possible improves the readability of the layout; 5) Length minimization, as links can correspond to real wires, where information travels faster as shorter these links are; 6) Symmetries, when present, should be reflected in the layout; 7) Clustering the nodes when its necessary to facilitate a better understanding of the graph's structure; 8) Layered drawings when is necessary to restrict the node positions to distinct layers. Panjtar [66] additionally recommends bounding the graph with a frame.

The problem with these aesthetics criteria is that the recommendations conflict with each other, not being possible “to implement all of them to the fullest at the same time” [44, p. 445]. So, many different methods have been developed that aim to optimize one or more aesthetic criteria [48]. Also, depending on the applications, some criteria may be more important than others [32]. For Klammler et al. [47], knowing what makes a drawing of a graph aesthetically pleasing is a question central to the field of Graph Drawing. The authors [47] sum up what has been done in this field, namely: 1) Attempts to use quality metrics, that can be simple metrics like the number of crossings, or advanced metrics, like energy or stress; 2) Resorting to performance studies on readability and clarity of representation that emphasize the purpose of a drawing or the way it facilitates possible user actions. They offer a different perspective, pointing out the importance of a drawing being aesthetically pleasing, and stating that, instead of trying to produce the best layout, several layouts can be obtained by different methods and then compared. So, even though new trends are emerging, classical methods for graph drawing are still current.

## 3.2 Methods and Techniques

There are several methods and techniques for drawing graphs. Fleischer and Hirsch [32] summarize some of the better techniques to address graph drawings optimisation, namely, planarization, force-directed methods, Sugiyama-like methods, flow methods, interactive drawings and labelling. A description of each of them is made and some examples are presented, giving more emphasis to those used in this dissertation.

Planarization consists in the elimination of crossings between links. When a graph drawing has no crossings, it is a planar drawing (see Figure 3.2) [53]. A graph is planar when a planar representation of this graph exists [79]. According to Fleischer and Hirsch [32, p. 20], “planar layouts are usually much more appealing than nonplanar layouts”. Algorithms to test for planarity date back to the 1960s [79]. As for making a graph planar, there are several ways to do it: deleting nodes, splitting nodes, inserting new nodes and deleting links [79]. Some methods rely on a special ordering of the vertices known as the canonical ordering [79]. Any non-planar graph can be converted to a planar graph [66].

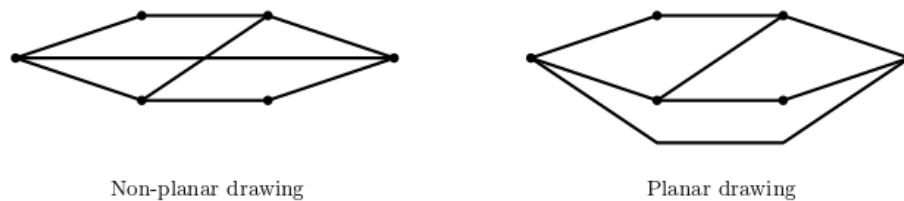


Figure 3.2: Example of a non-planar drawing and of a planar drawing (adapted from [53])

Force-directed methods “interpret a graph as a physical system with forces between the nodes and then try to minimize the energy of the system to obtain a nice drawing” [32, p. 20]. In this physical system, the goal is to find positions for each body, such that the sum of forces is zero [7].

According to Battista et al. [7], there are many force-directed methods, and they comprise two parts: 1) The model, which is a force system defined by nodes and links; 2) The algorithm, *i.e.* a technique to find the equilibrium state of the system, where the sum of forces is null. The model can also be defined as an energy system, in which case, the algorithm seeks to minimize the energy.

These methods are popular because the physical analogy makes them easy to understand and simple to code, and their results have quality [7], [10]. These algorithms also ensure that link lengths are uniform and that symmetries are visualized [66]. The fundamental concept behind physical modelling in these methods is having charged balls replacing nodes and springs replacing links (see Figure 3.3).

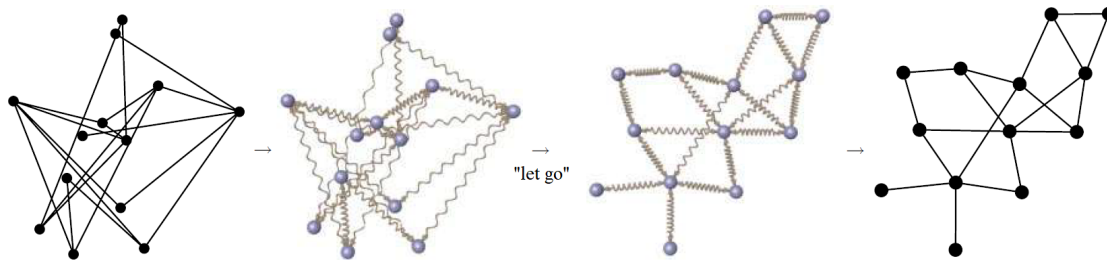


Figure 3.3: Example of the spring analogy (adapted from [10])

Sugiyama-like methods are the most widely used algorithms for drawing layered graphs [32], [40] because they produce layered layouts while also trying to minimize the number of crossings or the area of the layout [32]. In a layered layout, nodes at the same distance from the root (*i.e.*, the same depth) are in the same horizontal line (see Figure 3.4) [53]. Because these layers can also represent a hierarchy of existing relationships, which is common in directed graphs, these drawings are also known as hierarchical drawings [40]).

Algorithms for computing layered drawings date back to the 1970s, being the most popular the one from Sugiyama et al. in 1981 that was extended in 1990 by Eades and Sugiyama [6].

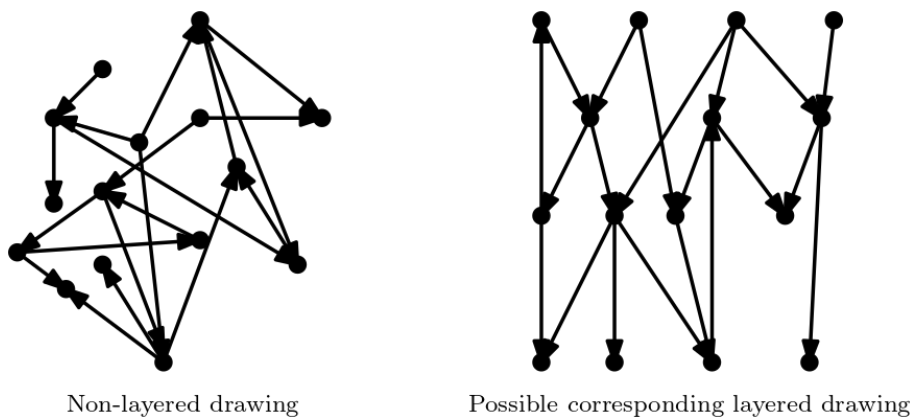


Figure 3.4: Example of a non-layered drawing and of a possible corresponding layered drawing (adapted from [6])

For Bastert and Matuszewski [6], these layered drawings should take into account some aesthetic and readability criteria that the authors summarize as: 1) When directed, upward links should be avoided; 2) Nodes should be uniformly distributed; 3) Crossings should be avoided; 4) Links should be as straight and vertical as possible. For Healy and Nikolov [40, p. 411], the aesthetics that should be taken into consideration are stated as: “links should point in a uniform direction, short links are more readable, uniformly distributed nodes avoid clutter, link crossings obstruct comprehension, straight links are more readable”.

Meeting all these criteria can be difficult and sometimes impossible and some authors [6], [7] suggest the following approach: 1) Cycle removal; 2) Layer assignment; 3) Cross reduction; 4) Horizontal coordinate assignment. These steps do not necessarily need to be all performed.

In cycle removal, the goal is to have links temporally drawn in one direction, reversing as few links as possible to have an acyclic graph [6], [7]. In layer assignment, nodes are assigned to horizontal layers in such a way that all links point downward, thus determining the nodes y-coordinate [6], [7]. Crossing reduction consists in reducing the number of crossings between links by ordering the nodes within each layer, which is usually done by examining adjacent layers and links between them [6], [7]. In the horizontal coordinate assignment, the horizontal positions of the nodes are such that they do not overlap and that no nodes lie on the straight lines between two adjacent nodes [6], [7]. In Figure 3.5 an example of the application of this approach is illustrated.

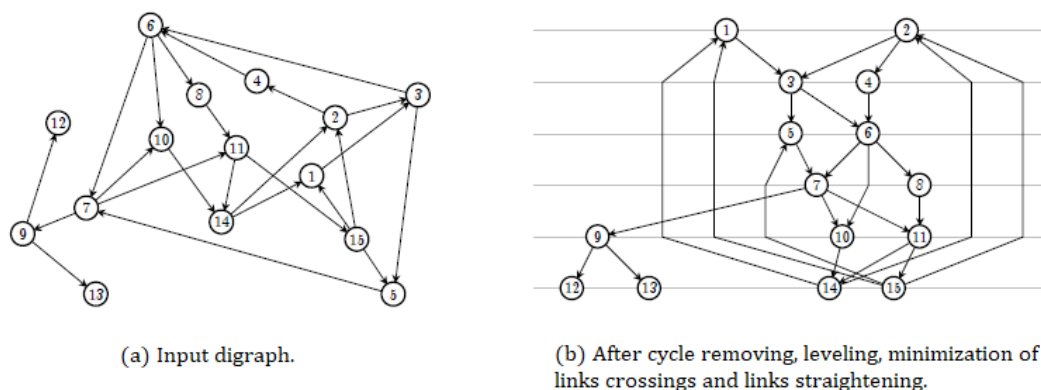


Figure 3.5: Example of the layering process of a non-layered graph drawing (adapted from [40])

Flow methods are used to minimize bends, which “can efficiently be solved by reduction to a network flow problem” [32, p. 21]. These techniques are also useful to maximize angles between links [32], addressing the problem, for the drawing legibility, of having links that are too close together [27]. For Eiglsperger et al. [27], a way to guarantee maximal distinctiveness of adjacent links is to force all angles between adjacent links to be multiples of 90 degrees, giving rise to the area of orthogonal graph drawing. However, to do so, bends must be included in the path representing a link. To avoid complicated paths, the number of these bends should be minimized. This can be done by introducing additional nodes where changes in direction occur in a path (see Figure 3.6). One disadvantage of having angles of 90 degrees is the fact that nodes cannot have more than four links each [26]. When more than four links per node are needed, the alternative is to use more general polyline drawing techniques, instead of standard orthogonal drawing techniques. In this case, and because it is not possible to have only 90 degree angle, the goal not have angles below a certain threshold [26]. According to Eiglsperger et al. [27, p. 121], this is “probably one of the most prolific in all of graph drawing(. . .) [ , becoming] far more important than the issues of angles in drawings”. For Blasius et al. [9, p. 860] “orthogonal graph drawing is one of the most

important techniques for the human readable visualization of complex data. Its aesthetic appeal derives from its simplicity and straightforwardness”.

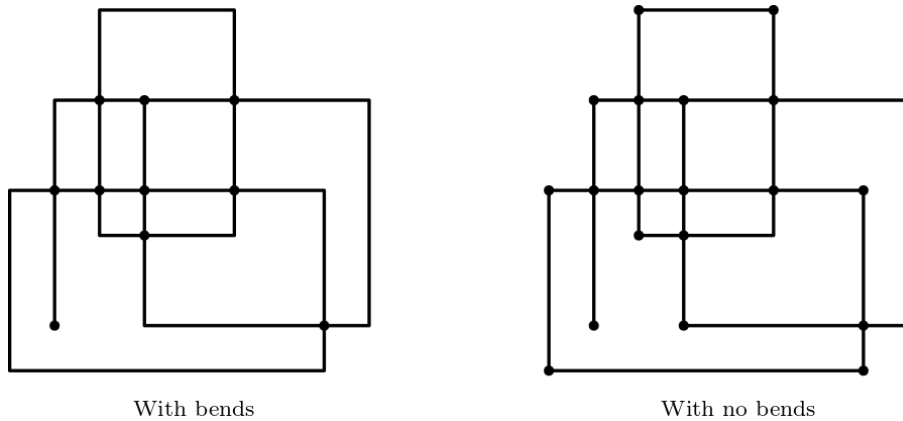


Figure 3.6: Examples of orthogonal drawings with bends and no bends (adapted from [27])

Interactive drawings apply to graphs that are not static [32] because the structure of nodes and links can change over time [8]. The field emerged in the 1990s since many situations required the “repeated redrawing of the graph after frequently occurring changes to the graph structure and/or some layout properties” [11, p. 228]. This happens when the user is allowed to manually edit the graph, adding, removing and moving nodes and links, or by setting additional constraints [7], [11]. According to Branke [11], the easiest solution for this problem is to treat the changed graph as a new graph and apply a static graph drawing algorithm every time that an alteration is made. However, this is not efficient, because, in the case of slight modifications, much of the previous drawing can be reused, saving computation time. On the other hand, if the user is still working in that graph drawing and have a mental picture of it, having it change at every slight alteration will require a considerable effort for the user to re-familiarize with it [11]. Battista et. al [7] argue that for the example given in Figure 3.7, where the link  $(b, d)$  is added in graph drawing  $G$ , graph drawing  $G_2$  is preferable to graph drawing  $G_1$ , because it maintains the overall look of the drawing, even though this implies having a crossing in  $G_2$  that  $G_1$  does not have.

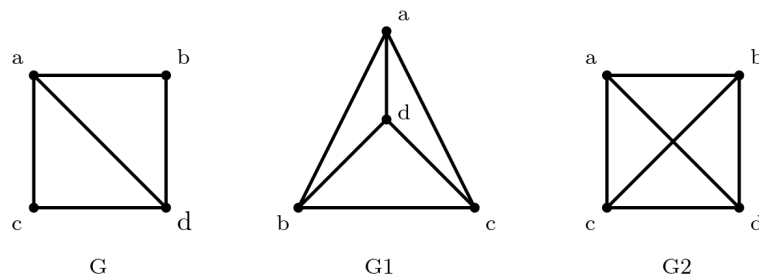


Figure 3.7: Example of a graph drawing  $G_1$  that does not maintain the user’s mental picture of the graph (adapted from [7])

According to Battista et al. [7, p. 220], software with “interactive drawing features should be able to: (a) create a drawing of a given graph under some layout standard, and (b) give the user the ability to interact with the drawing”. This interaction should allow the user to insert a link between two nodes, insert a node and its links, delete links, nodes or sets of links and nodes, move a node and move a set of nodes and links. The way the interactive graph drawing responds to these alterations depend on the amount of control that users have on the graph elements and on how different the new drawing will be, compared to the current one.

So, interactive drawings, also known as dynamic graph drawings, emerged to address the need to frequently change the graph itself, adjusting its drawing to the changes, whilst considering the usual optimization criteria for graph layout and preserving the users’ mental map, which is also known as “maintaining dynamic stability” [11, p. 230]. At that time, according to Branke [11, 229-230], two ways were being used to deal with the issue of maintaining the dynamic stability: “Support the user by animating and highlighting the changes so that the changes can be easily recognized and the transitions are smooth; 2) Minimize the changes such that the effort to regain familiarity is minimized”.

Although “the field was first understood as a subproblem of graph drawing (. . .), after the millennium, with the availability of more and more time-varying datasets, dynamic graph diagrams were discovered as an information visualization technique. Alternatives to animated node-link diagrams that plot the graph onto timelines were introduced [see Figure 3.8]. By 2010, the visualization of dynamic graphs was established as a standard visualization discipline” [8, p. 133].

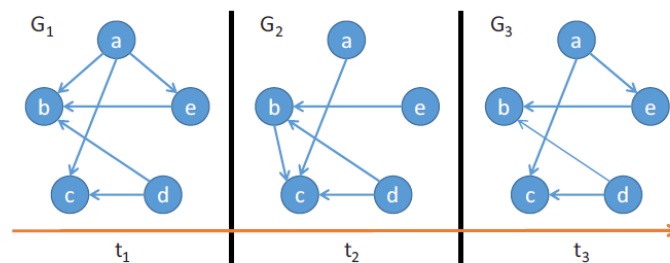


Figure 3.8: Example of changes made to a graph drawing being visualized in a timeline (adapted from [8])

Labelling consists of naming nodes and links in a graph drawing [32], which means that “text labels have to be associated with graphical features” [65, p.247]. For Kakoulis and Tollis [45, p. 489], “because the labelling process is a monotonous and very demanding task, its automation is very desirable”. However, for the authors, it is unlikely that computer-based systems can produce labelling of the same quality of experienced professionals. So, they consider that “semi-automated interactive name placement systems may be the most practical approach at the present time” [45, p. 489]. The difficulty of labelling is caused by the proximity between the object to be labelled and other objects, restricting the freedom to place the label. So, the labelling must take into

consideration “not only the position of a label with respect to its associated object, but also how it relates to other labels and objects in the surrounding area” [45, p. 490].

Regarding general labelling quality evaluation, Kakoulis and Tollis [45] summarize it as: 1) A label cannot overlap other labels or other drawing elements; 2) Identification of exactly what label corresponds to what drawing elements must be easy; 3) A label must be placed in the best possible position. For the authors [45], a typical rule when labelling nodes is to place labels to the right and above the point representing the node. The label can touch the point but not overlap it (see Figure 3.9). As for links, a label can touch the link to which it belongs, but not any other drawing element (see Figure 3.9).

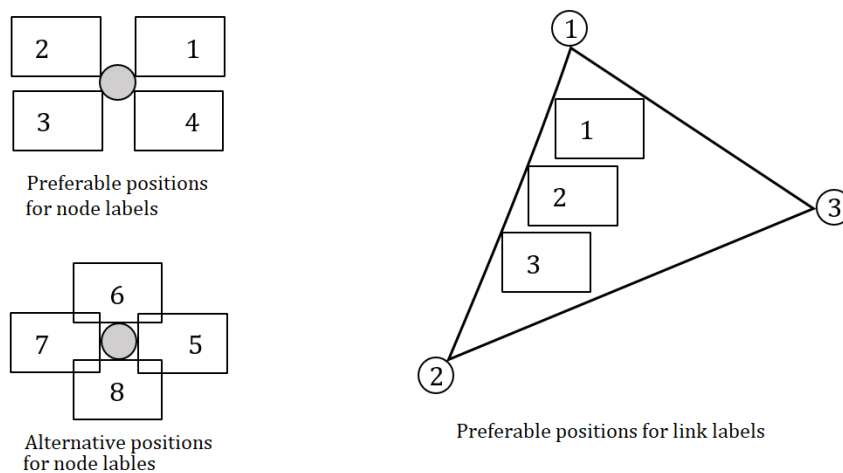


Figure 3.9: Example of possible positions for labelling a node (on the left) and for labelling a link (on the right) (adapted from [45])

Despite the enunciated rules to facilitate labelling and improve its quality, “there are cases where the best methods available do not always produce an acceptable or legible solution even if one exists. Furthermore, there are cases where no feasible solution exists” [45, p. 511]. When this happens, the only solution is to modify the drawing. For the authors, there are two algorithmic approaches to do this: "1) Modify the existing layout of a graph drawing to make room for the placement of labels; 2) Produce a new layout of a graph drawing that integrates the layout and labeling process" [45, p. 512].

For Neyer [65], labelling should occur when the graph is being drawn, to avoid situations where the drawing is too dense to be labelled. This is known as integrated labelling (see Figure 3.9) because labels are an integrated part of the layout calculation. As labels are applied during the layout process, this allows for considering labels as part of the overall layout [83]. Nonetheless, algorithms that add labels to previously existing drawings, without modifying them are useful, because in many situations the drawings represent geographical or technical information that cannot be changed. This type of labelling is called generic labelling (see Figure 3.10) and it consists of placing labels on a graph drawing without changing its layout [83].



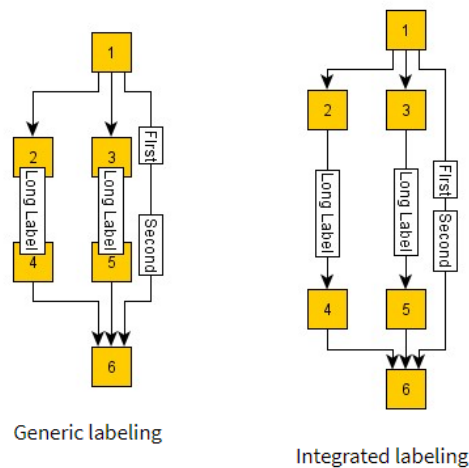


Figure 3.10: Example of a generic labelling and the corresponding integrated labelling (adapted from [83])

### 3.3 Summary

In this chapter, graph drawings and related issues were addressed. Of these, it is important to highlight that:

- Graphs can be represented in several ways. Graph drawings must satisfy a set of drawing conventions. Good layouts depend on aesthetics criteria and technical criteria. There are a set of recommendations regarding the aesthetics criteria. Some of these recommendations conflict with each other, given rise to different methods and techniques for optimising graph drawings.
- The better techniques to address graph drawings optimisation are planarization, force-directed methods, Sugiyama-like methods, flow methods, interactive drawings and labelling.
- Sugiyama-like methods are the most widely used algorithms for drawing layered graphs and hierarchical drawings, taking into consideration the direction, orientation and crossing of the links and the nodes distribution.



## Chapter 4

# Wafer fabrication and DoE

This chapter includes aspects of wafer fabrication and DoE. Section 4.1 describes semiconductor manufacturing, focusing on wafer fabrication. Section 4.2 addresses DoE and its application to semiconductor manufacturing and wafer fabrication. Section 4.3 presents the CMF module for DoE and its main features and functionalities, making the connecting with the problem addressed in this dissertation. Section 4.4 summarizes the highlights of this chapter.

### 4.1 Wafer Fabrication

A great variety of electronic equipment such as smartphones, digital cameras, personal computers, and other use semiconductor devices [78]. Semiconductor devices are electronic components like integrated circuits (IC), diodes, and transistors that are made of semiconductor materials [42], which are substances “possessing an electrical conductivity somewhere between that of a conductor and that of an insulator. Silicon is a representative example of a semiconductor” [42, pt. S]. The physical unit that is used for manufacturing semiconductor devices is called a wafer (see Figure 4.1) [42]. The wafer is, in general, “made by slicing a silicon ingot (a cylindrical mass) into disk-shaped pieces of about 0,5 mm to 1 mm thick” (see Figure 4.2) [42, pt. W].

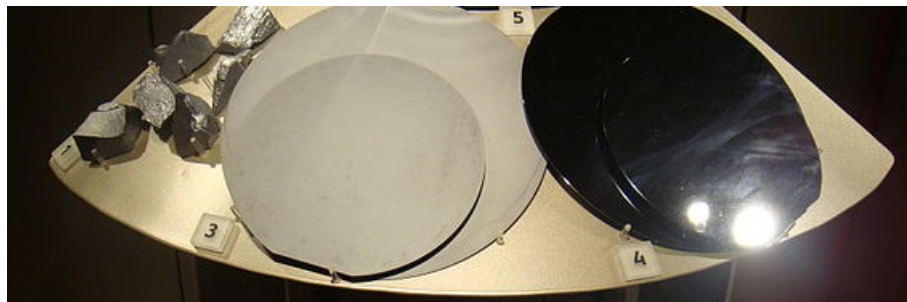


Figure 4.1: Wafers at the national Museum of Scotland ([64] under the Creative Commons Attribution-Share Alike 3.0 Unported license)



Figure 4.2: A silicon ingot at the Intel Museum ([3] under the Creative Commons Attribution-Share Alike 3.0 Unported license)

The raw wafer is submitted to several manufacturing operations, that typically consist of “depositing insulating or dielectric layers, conductive layers, and semi-conductive layers of material over a semiconductor substrate, and patterning the various material layers using lithography to form circuit components and elements thereon” [78, para. 1]. These layers will vary, depending on the device that is being made. For example, “in the manufacturing process of IC, electronic circuits with components such as transistors are formed on the surface of a silicon crystal wafer [42, para. 2]. A single wafer has more than one hundred semiconductor dies [43], which is “the minimum unit of the semiconductor device to be cut out individually” (see Figure 4.3) [42, pt. D].

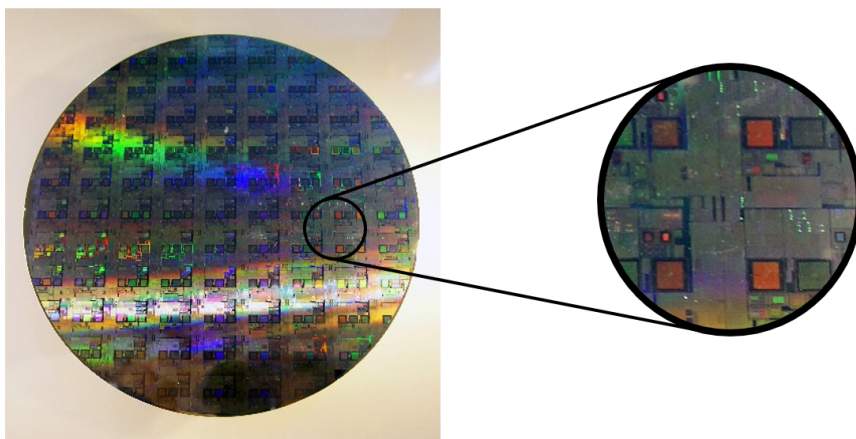


Figure 4.3: A silicon wafer with 30 cm after the depositing of all the layers and before dies are cut (adapted from [69] under the Creative Commons Attribution-Share Alike 3.0 Unported license)

Wafer fabrication is the first stage in the manufacturing of semiconductors. Because the resulting components depend on “chemical purity and near-perfect crystalline properties of wafers, (...) the production of silicon wafers has achieved a high degree of intricacy with many complex phases” [39, p. 640]. This complexity is corroborated by May and Spanos [62, p. 8] when they consider that “the modern semiconductor manufacturing process sequence is the most sophisticated unforgiving volume production technology that has ever been practiced successfully. It consists of a complex series of hundreds of unit process steps that must be performed very nearly flawlessly”.

The manufacturing process has steps. Each step has inputs, outputs, and specifications. Multiple process steps are linked together to form a process sequence, which is called a process flow [62]. According to Gupta [39, p. 640], “most wafer fabs contain over 100 machines and often there are dozens of process flows, each with a very specific set of 300–500 processing steps”. This is due not only to the complexity of the process itself but also to the wide variety of ICs that it is necessary to produce, as the demands from the automobile industries and medical and healthcare industries, amongst others, are increasing rapidly [80].

Because of the increasingly high costs involved mainly due to the equipment, and of the decreasingly low IC chip prices [80], manufacturers are under pressure to produce larger quantities and improve quality and delivery times [39]. So, they strive to schedule the various orders (jobs) in their factory in a way that will allow them to make the most of all resources. This “requires scheduling of jobs on a large number of machines where jobs re-enter processing several times after [they have been] processed by subsequent machines” [39, p. 644], thus creating multiple process flows with split and merge operations. In these operations, “two or more lots can be merged into a single lot if routes and all the processing conditions of the lots are the same for a number of subsequent operations, and the merged lot is split into the original lots at the point where the routes or processing conditions become different” [5, p. 2209]. Bang et al. denote as wafer group, the “wafer lots that can be merged together, *i.e.*, those with the same processing conditions and the same next workstation to visit” [5, p. 2210].

According to Liu and Li, “many production systems have split and merge operations to increase production capacity and variety, improve product quality, and implement product control and scheduling policies” [54, p. 4373].

## 4.2 Wafer DoE

On the one hand, because wafer fabrication is “the most costly and time consuming of the semiconductor manufacturing steps” [39, p. 640], any improvements that can be made are very important to manufacturers [39]. On the other hand, “as wafer fabrication processes become increasingly complicated in nano technologies, many factors (...) affect the yield of fabricated wafers, while spec tolerance is significantly reduced owing to physical limits. Hence it is increasingly difficult to quickly identify the root causes and suggest the corresponding corrective actions when yield loss occurs to minimize the incurred excursion cost” [14, pp. 961-962].

A method that has been used to provide a quick and cost-effective way to optimize manufacturing is the DoE [75], [14]. Abreu et al. [1, p. 2] define DoE as “a methodology that allows us to select the best combination of factors levels in order to optimize the value (s) of the quality characteristic (s) under study, both in terms of its mean value such as the reduction of variability, *i.e.*, allows to determine which controllable factors affect certain quality characteristics and which are the best levels of these factors in order to increase the resistance of the product to the noise factors, thus satisfying the requirements of the various stakeholders in the performance of an organization”. For Giles et al. [37, p. 291], “the advantage of using a DOE approach is that systematic data are generated, summarized, and evaluated to definitively determine whether a project should be carried forward or if it is fundamentally impossible to resolve and needs to be dropped”.

Farooq et al. [30, p. 155] state the process improvement from the application of DoE as being the “improved process yields, reduced process variability and reduced overall costs” and that “over the past many years, industries have successfully applied DoE to improve process performance and reduce variability”.

Nonetheless, and according to Chien et al. [14], even though DoE has been successfully applied to manufacturing, employing it to semiconductor manufacturing is a difficult task. The authors [14] present three reasons for that difficulty: 1) This industry is very capital intensive, with a 24/7 equipment utilization to have an early return on the capital investment, that leaves no room for other uses of the equipment, like experimentation; 2) Experiments itself have also the high costs of time and money associated with wafer fabrication and to the experimentation process, which limits the number and variety of experiments that can be done; 3) Wafer fabrication involves hundreds of re-entrant process steps (splits and merges) which makes the experiments very hard to follow, due to the complex data variety.

To better understand how DoE has been used in semiconductor manufacturing and wafer fabrication, a search was conducted in the Scopus database. The inclusion criteria were: 1) Being published after 1999 (last 20 years); 2) Having (“semiconductor manufacturing” and “design of experiments”) or (“wafer fabrication” and “design of experiments”) in the keywords (so that the focus of the documents were the intended topics). This search returned 28 document results for “semiconductor manufacturing” and “design of experiments”. Of these, 15 (54%) are conference papers, and 13 (46%) are articles. More than half (54%) are from North America, 32% are from Asia and 14% are from Europe. The last decade accounts for 68% of these documents. As for the content, it can be summarized as being related with the use of DoE in the semiconductor manufacturing: 1) With other techniques or methods like data mining (*e.g.* [14]) and machine learning (*e.g.* [67]); 2) For simulation (*e.g.* [46]); 3) To improve metrological (*e.g.* [13]) or technological aspects (*e.g.* [51]) related with the manufacturing process; 4) To improve time-related issues of the manufacturing process (*e.g.* [70]). As for the search with “wafer fabrication” and “design of experiments”, the results were eight documents equally distributed between conference papers and articles. The majority (67%) are from Asia and the rest (33%) from North America. The last decade saw a decrease in the publication from 67% to 33%. The content was all related to very specific critical technical issues of wafer fabrication (*e.g.* [82], [56]). Three of the eight identified

documents had “semiconductor manufacturing”, “wafer fabrication”, and “design of experiments” in the keywords and had also shown up as results of the first search.

Additionally, these documents were inspected for illustrations of the DoE interface. Some had illustrations (*e.g.* [56]), but they were all of the tabular types. No mention of graphs or graph drawings were found.

### 4.3 CMF Experiment Management Module

To allow and facilitate the DoE, making it an easier, more efficient, and effective process, CMF has recently integrated into its MES the EM module [17]. According to CMF, the EM module will allow to “reduce time by conducting multi-parameter experiments rather than test one variable at a time; conduct experiments transparently, side-by-side, in a mass production factory; understand and gain in-depth knowledge of the process variables and their effect; improve Quality and Reliability of process and products; decrease product development and production costs; [and] speed Time-to market” [18, para. 4]. Besides these benefits, because this module is “an integral part of the MES, experiments can be conducted alongside daily production schedules with complete transparency” [68, para. 7]. The data collection of the shop floor that is a part of the CMF MES is available in the CMF EM module, which “means engineers can readily gain in-depth knowledge of the process variables and their effect on production” [68, para. 7].

The CMF EM module can be accessed from CMF MES selecting «Business Data» and «Experiment Definition» (see Figure 4.4).

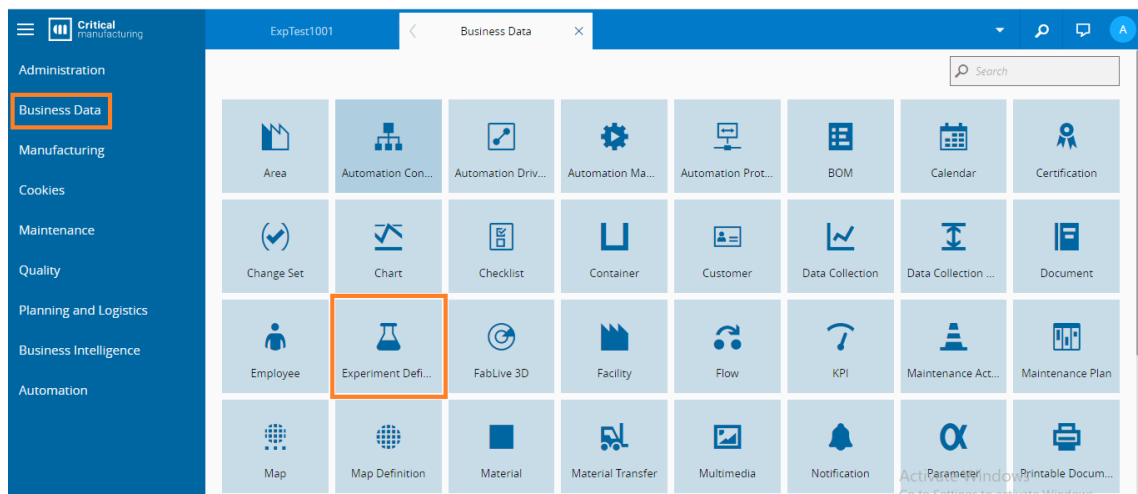


Figure 4.4: Accessing CMF EM module in CMF MES (orange highlight)

Upon entering, the user can access a previously defined experiment, import an experiment defined somewhere else (a .xml file), or define/create a new one (see Figure 4.5). Additional features include the possibility of refreshing the information being shown, searching and querying (see Figure 4.6).

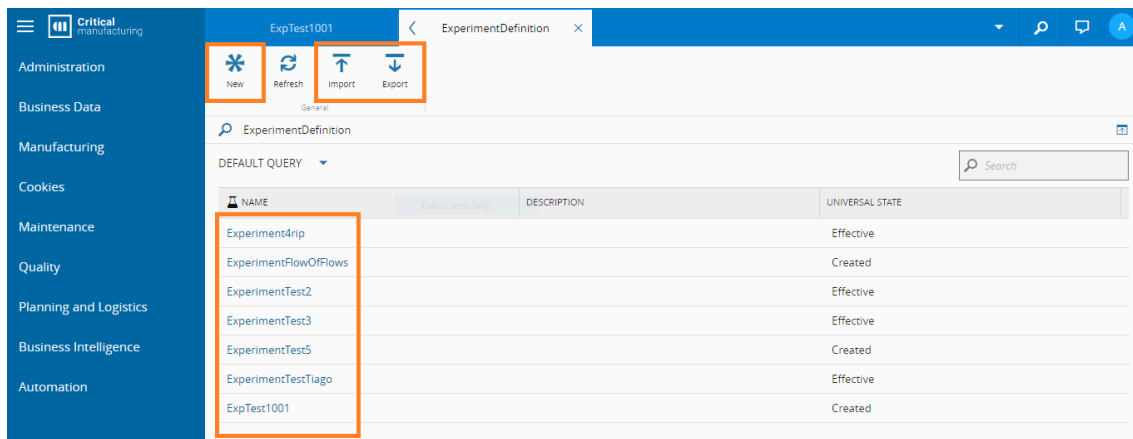


Figure 4.5: Accessing an experiment in CMF EM module's main menu (orange highlight)

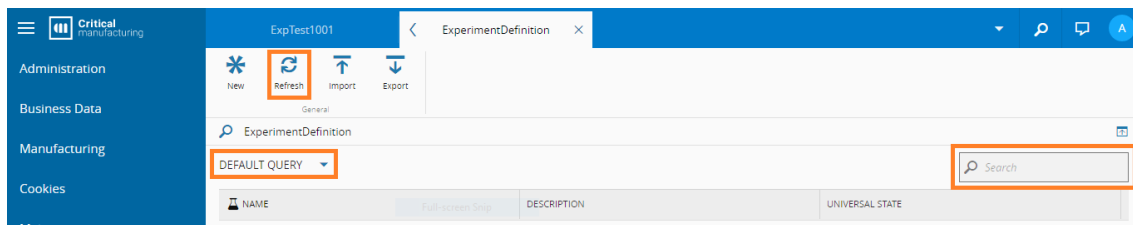


Figure 4.6: Additional features in CMF EM module's main menu (orange highlight)

Before presenting the stages involved in the definition and creation of an experiment and how to do it in the CMF EM module, it is important to understand what elements are necessary to design an experiment for wafer fabrication in this module.

In general, an experiment consists of subjecting a set of materials (sub-materials) to a set of manufacturing process steps (flow). Thus, for each experiment, it is necessary to choose the main flow, from a set of available flows. This flow can itself be a set of flows, known as a Flow of flows. Other steps from other flows can be added to the selected main flow. At each step, the materials can be subject to actions. Different actions can be chosen for different materials. There is a set of eleven existing actions to choose from. Some actions (ChangeFlowAndStep, TemporaryOffFlow and Terminate) determine which materials to subject to which steps of the flow. The actions ChangeFlowAndStep and TemporaryOffFlow cause the materials to change to another flow, i.e. they create splits and/or merges in the flow. Applying the action Terminate to a set of materials prevents them to advance to other steps of the flow, which can also create a split. The other available actions are Hold, SetBOM, SetChecklist, SetDataCollection, SetDurables, SetMeasureAll, SetNote, and SetRecipe. It is also necessary to indicate, for each action being applied, in what moment of the respective step it will be performed. This information is given by the events. Each action is associated with certain events from a pre-existing set of four (Track-In, Track-Out, Queued, Processed).



To do the DoE in the CMF EM module it is necessary to specify all the materials, the main flow, other steps from other flows, the actions, to which material in which steps these actions apply, and the respective events. This is done in the definition of an experiment and its creation.

In the CMF EM module itself, the definition of experiments comprises several stages: 1) Change set; 2) General data; 3) Objectives; 4) Material groups. The stages are presented in Table 4.1, Table 4.2, Table 4.3, Table 4.4, Table 4.5, and Table 4.6 applied to the wafer fabrication.

Table 4.1: Description and illustration of stage 1 (Change set) of the experiments' definition

| DESCRIPTION AND ILLUSTRATION   |
|--|
| <p>- To select a change set for the experiment is mandatory. The experiment gets its universal state from the change set.</p> <div data-bbox="437 757 1369 1048"> </div> |

Table 4.2: Description and illustration of part 1 of stage 2 (General data, information) of the experiments' definition

| DESCRIPTION AND ILLUSTRATION   |
|--|
| <p>- A name and a description can be attributed to the experiment. The name and the type are mandatory and the type will allow better future organization and search of information.</p> <div data-bbox="437 1397 1369 1576"> </div> |
| <p>- Information about the owner's role and purpose of the experiment can be included. It is also possible to add some notes about the experiment.</p> <div data-bbox="437 1666 1369 1968"> </div>                                   |

Table 4.3: Description and illustration of part 2 of stage 2 (General data, settings) of the experiments' definition



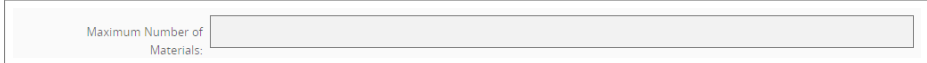




| DESCRIPTION AND ILLUSTRATION   |  |
|--|--|
| <p>- Users can choose whether the experiment will be applied to a product or a product group. This changes the information presented to them and from which they can make their following selections.</p>  |    |
| <p>- The inclusion of information about the product is not mandatory but recommended. Once the product is chosen, its flow (a unique set of associated manufacturing processes) is automatically selected, although it can be changed (flow is mandatory).</p>   |    |
| <p>- Users have the option to include information about the maximum number of materials.</p>   |    |
| <p>- Users can also associate the required material type and the required material form with the experiment. These are optional.</p>   |  |
| <p>- For mode, the options are sub-materials or full material. Full material is a block of raw material and sub-materials are portions of full materials. This means that sub-materials can be separated and therefore subject to different flows, while a full material because it is a single block, can only be subjected to a single flow. This mode selection changes the information that is presented to users and from which they must choose. Because wafers are sub-materials, this is the selection for mode.</p> |  |
| <p>- The indication of the required sub-materials count, and the required sub-materials form is mandatory. The form is selected from a pre-existing set of options, that are available in CMF MES.</p>   |  |
| <p>- Information about the planned start date and the planned end date can also be included.</p>   |  |

Table 4.4: Description and illustration of part 3 of stage 2 (General data, options) of the experiments' definition

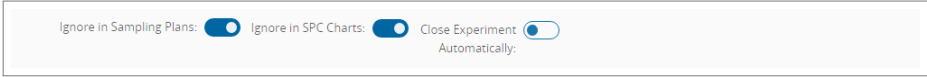
| DESCRIPTION AND ILLUSTRATION   |
|--|
| <p>- Users must select the options from the following yes or no possibilities: ignore in sampling plans, ignore in SPC charts, and close experiment automatically.</p> |
|    |

Table 4.5: Description and illustration of stage 3 of the experiments' definition (Objectives)


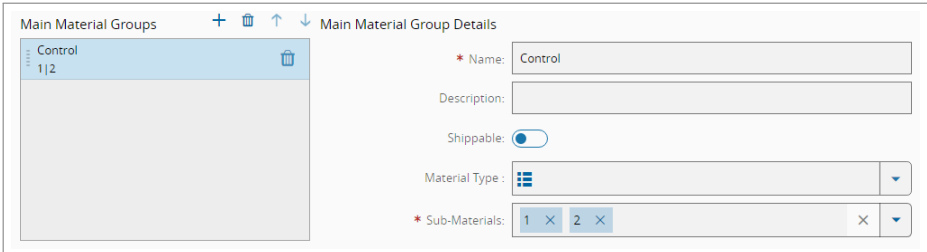
| DESCRIPTION AND ILLUSTRATION  |
|---|
| <p>- Objectives for the experiment can be added. After this, they can be deleted and sorted.</p> <p>- Naming the objective is mandatory. Additional information about the description of the objective and its target can also be included.</p> |
|    |

Table 4.6: Description and illustration of stage 4 of the experiments' definition (Material Groups)

| DESCRIPTION AND ILLUSTRATION   |
|--|
| <p>- The definition of the material groups must be done. The advantage of having these groups is the possibility of subjecting them to different processes (flows). At least one group must be added, but there can be as many groups as sub-materials (required sub-materials count). These groups can also be deleted and sorted.</p> <p>- Naming the material groups is mandatory. Additional information required is the description (optional), whether it is shippable (required; yes or no), and the type of material (optional). For each group, users must select their respective sub-materials. All sub-materials (required sub-materials count) must be part of a group.</p> |
|    |

After the experiment is defined, it must be created, which is done by hitting «create» (see Figure 4.7).

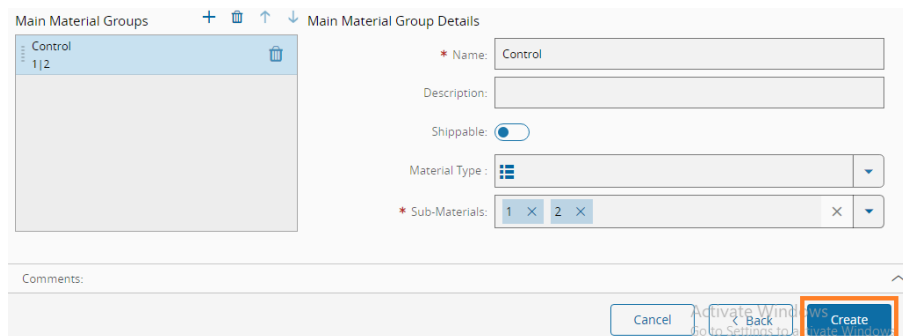


Figure 4.7: Creation of an experiment in the CMF EM module's (orange highlight)

Hitting «create» allows steps to be added to the defined experiment (see Figure 4.8).

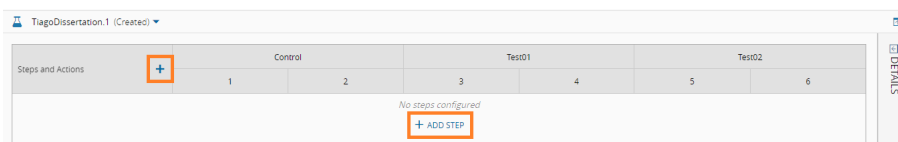


Figure 4.8: Addition of steps to an experiment in the CMF EM module's (orange highlight)

The addition of steps to an experiment comprises three stages: 1) General data; 2) Material groups; 3) Actions (see Table 4.7, Table 4.8 and Table 4.9).

Table 4.7: Description and illustration of stage 1 (General data) of adding a step to a defined experiment

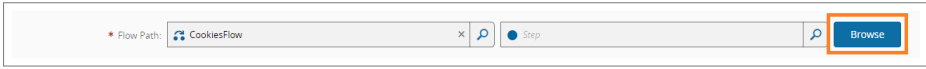

| DESCRIPTION AND ILLUSTRATION  |
|---|
| <p>- The steps to be selected are part of a flow path. Because of that, the selection of a flow path is mandatory. By default, the flow path is the one chosen in the experiment definition, but it can be altered. Clicking «browse» shows the steps of the selected flow (orange highlight).</p>  |
| <p>- After clicking «browse», the steps of the selected flow path are shown and can be selected (orange highlight).</p>   |

Table 4.8: Description and illustration of stage 2 (Material groups) of adding a step to a defined experiment

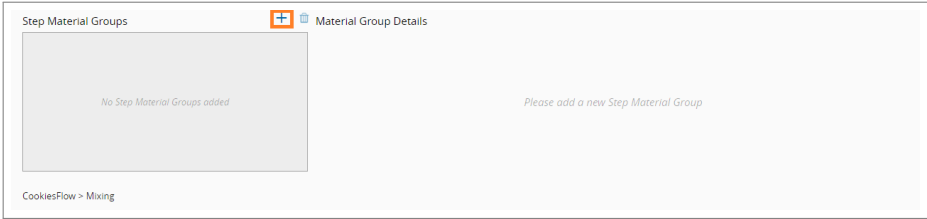
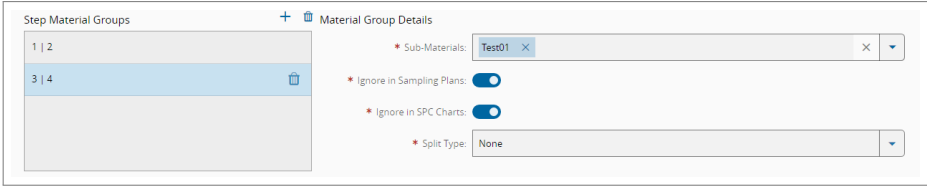
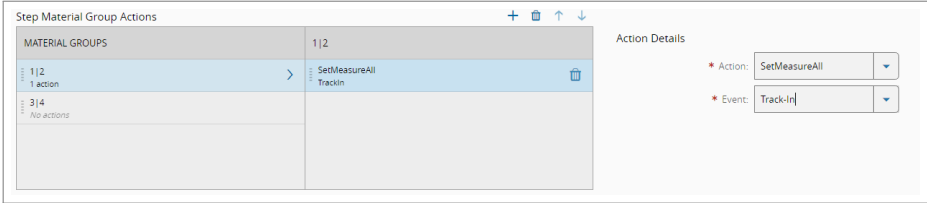
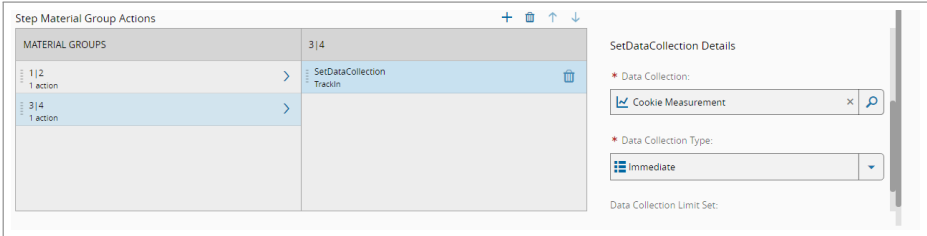
| DESCRIPTION AND ILLUSTRATION   |  |
|--|--|
| <p>- For the selected step, material groups must be chosen (orange highlight).</p>   |  |
| <p>- In the material groups details, the selection of sub-materials is mandatory. These sub-materials can be selected from the material groups previously defined (see Table 4.6), selecting a group itself or any of its elements separately. In the example, two groups were selected, each with two sub-materials (1 and 2, and 3 and 4, respectively for group «Control» and for group «Test01»).</p> <p>- It is also mandatory to indicate the type of split to apply to the selected sub-materials. Split type can be none, physical or logical.</p>  |  |

Table 4.9: Description and illustration of stage 3 (Actions) of adding a step to a defined experiment

| DESCRIPTION AND ILLUSTRATION   |  |
|--|--|
| <p>- For each one of the step materials group, one or more actions and events must be selected.</p>                                      |  |
| <p>- Fill-in all the information regarding the actions and events, that can be different depending on what was chosen, and save it.</p>  |  |

After a step definition, the user can see all the information for that step, which can be edited if necessary (see Figure 4.9). Additional steps can be added, repeating the process described. As stated before, and as illustrated in Figure 4.9, there is only available a tabular view, which does not facilitate user interaction and does not prevent the execution of logically and incorrect experiments.

| Steps and Actions           | Control |   | Test01 |   | Test02 |   |
|-----------------------------|---------|---|--------|---|--------|---|
|                             | 1       | 2 | 3      | 4 | 5      | 6 |
| SetMeasureAll (TrackIn)     | 1       | 2 |        |   |        |   |
| SetDataCollection (TrackIn) |         |   | 3      | 4 |        |   |

Figure 4.9: Example of a step and respective actions and material groups in the CMF EM module's

## 4.4 Summary

This chapter was dedicated to wafer fabrication and DoE and the CMF EM module. Of the addressed topics, it is important to highlight that:

- Wafer fabrication is the first stage in the manufacturing of semiconductors. The semiconductor manufacturing industry is very complex, involving increasingly higher costs and greater demands. The optimisation of the process flow requires multiple splits and merges.
- DoE is a method to provide a quick and cost-effective way to optimize manufacturing, even though applying it to semiconductor manufacturing is a difficult task.
- To allow and facilitate the DoE, making it an easier, more efficient, and effective process, CMF has recently integrated into its MES the EM module. Nonetheless, the designed experiment being shown only in a tabular view does not facilitate user interaction and does not prevent the execution of logically and incorrect experiments.

## Chapter 5

# Solution development and methodology

This chapter describes the methodology used for the development of the implemented solution, according to the user stories and the objectives of this dissertation. Thus, section 5.1 presents the user stories, the tasks that needed implementation, and the methodology used to do it. The following sections describe the development of the solution. Section 5.2 describes all aspects related with graph visualization, section 5.3 with logical verification, section 5.4 with the improvement of user features and section 5.5 with testing and validation. Section 5.6 summarizes the highlights of this chapter.

### 5.1 User stories and development methodology

Although the existence of the CMF MES EM module facilitated the DoE, there are designed experiments that are executed despite having logical errors. Because of that:

- Users want to detect logical errors, to correct them, thus preventing the execution of logically invalid experiments.
- Users want a visualization of the designed experiment that is similar to the wafer fabrication process flow, to facilitate the interpretation of what is being visualized.
- Users want to be able to filter the visualization of the designed experiment by a sub-material or set of sub-materials, to individually analyse the respective process flow.

Taking the user stories into consideration, the general objective and the specific objectives of this dissertation were defined, and a set of tasks that allow their accomplishment were identified.

To facilitate the presentation of the development of the solution, these tasks were organized in four stages: preparation and graph visualization, logical verification, improvement of user features, and testing and validation. These tasks, and respective stages, are the following:

#### STAGE 1: PREPARATION AND GRAPH VISUALIZATION

- To get acquainted with the technology, processes, concepts, and terminology involved in semiconductors manufacturing, namely wafers fabrication, CMF MES, in general (developer perspective) and CMF MES EM module, in particular (developer and user perspective).
- To research and choose a graph drawing tool, checking its adequacy, and testing it with hardcoded information.
- To test the selected graph drawing tool in drawing graphs with information parsed from CMF MES, regarding an experiment with no splits and merges in the process flow and to save the graph information.
- To adjust visual details and implement basic features for user interaction.
- To draw graphs of different types of experiments, by adding processes resulting from splits and merges to the process flow.

#### STAGE 2: LOGICAL VERIFICATION

- To analyse the errors from designed experiments caused by splits and merges in the wafer fabrication to identify how they translate into occurrences in graph drawings.
- To adapt the coded information structure of the graphs to facilitate logical verification of the graph drawings occurrences, to identify relevant graph information for the logical verification, making it as trivial as possible.
- To implement the logical verification, including simple feedback for users.

#### STAGE 3: IMPROVEMENT OF USER FEATURES

- To research algorithms to organize the graph drawings, improving their quality and their resemblance to the manufacturing process.
- To improve the previously developed basic features for user interaction, making them more versatile and adjusted to the information structure of graphs used for logical verification.
- To implement features that facilitate users' identification of errors' cause and related information, taking into consideration users' requests.

#### STAGE 4: TESTING AND VALIDATION

- To populate the database with experiments with known errors, drawing its graphs and analysing the results with experts.

To complete these tasks and develop the solution to be implemented, an agile software process model was chosen, more specifically the SCRUM framework. "Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering



products of the highest possible value" [74]. SCRUM was chosen because it is the methodology adopted by CMF in their projects, and because of its adequacy for this specific project. This adequacy is due to SCRUM being used when: "1) Requirements are not clearly defined; 2) The probability of changes during the development is high; 3) There is a need to test the solution; 4) The Product Owner is fully available; 5) The client's culture is open to innovation and adapts to change" [41].

In the SCRUM framework, projects are developed in two weeks periods, known as sprints. The sprint begins with the sprint planning, in which priority tasks that can be accomplished in the duration of a sprint are chosen. At the end of the sprint, the sprint review and the sprint retrospective are done. In the case of this project, these included a presentation for CMF staff with knowledge about this project and the CMF MES.

The tasks were analysed to evaluate the priority of each one, and, at the beginning of each sprint, moved to the sprint backlog.

## 5.2 Preparation and Graph Visualization

To get acquainted with the necessary aspects of semiconductors manufacturing, namely wafers fabrication, and the CMF MES, in general, and CMF MES EM module, in particular, was a priority task that was done with the help of CMF staff and experts, at the early stages of the internship.

When researching for a graph drawing tool to be used to draw the graphical representation of a graph, it was found that there are several open source libraries available. The CMF MES already uses one of these libraries, JointJS, to draw a graphical representation of different types of information in other modules.

JointJS is an open source javascript library that allows users to "create static diagrams or fully interactive diagramming tools such as workflow editors, process management tools, IVR systems, API integrators, presentational applications and much more" [16, para. 1]. Its Model-View-Controller "architecture separates graph, element and link models from their rendering. A diagram in JointJS is represented by a Graph model (`joint.dia.Graph`) to which models of cells, either Elements (subtypes of `joint.dia.Element`) or Links (subtypes of `joint.dia.Link`), can be added" [15, para. 1]. JointJS provides "a visual library of common geometric shapes, as well as an extensive collection of ready-to-use components from several well-known diagramming languages" [15, para. 1]. JointJS also provides functions to automatically position the nodes resorting to an algorithm and its respective parameters. JointJS Core library is licensed under the Open Source Mozilla Public License Version 2.0 [16].

To assess the adequacy of JointJs in drawing graphs for the CMF EM module, it was tested with a basic set of hardcoded information (see Figure 5.1), to create a few nodes and links, and checking if it produced a good graph drawing. The obtained result (see Figure 5.2) showed that JointJS is adequate for this situation, because of its ability to represent the necessary drawing elements.

```

// Create graph and paper
const graph = new joint.dia.Graph;

const paper = new joint.dia.Paper({
  el: this.holder.nativeElement,
  model: graph,
  width: 1360,
  height: 500,
  interactive: function (cellView) {
    if (cellView.model instanceof joint.dia.Link) {
      return { vertexAdd: false };
    }
    return true;
  },
});

// Create nodes
const rect1 = new joint.shapes.basic.Rect({
  position: { x: 100, y: 50 },
  size: { width: 100, height: 60 },
  step: 'Step01'
});

const rect2 = new joint.shapes.basic.Rect({
  position: { x: 100, y: 250 },
  size: { width: 100, height: 60 },
  step: 'Step02'
});

// Create links
const link1 = new joint.dia.Link({
  source: rect1,
  target: rect2,
});

// Add elements to graph
rect1.addTo(graph);
rect2.addTo(graph);

link1.addTo(graph);

```

Figure 5.1: JointJS with hardcoded information

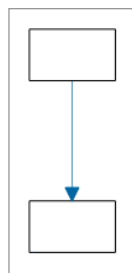


Figure 5.2: Graph drawing with hardcoded information in JointJS

After this, JointJS was tested with an experiment defined in the CMF EM module for this purpose. In this experiment, there are six sub-materials organized in three groups of two sub-materials each, and there is a main flow with five steps (from step 00 to step 04). The experiment was initially defined in its simpler version (Version 1) and was progressively incremented and changed to test the representation of different types of actions in graph drawings, namely the ones that can cause the errors. Six versions (Version 1 to Version 6) of the experiment were created to

test and illustrate the development of the solution. All versions of the experiment tested were the following:

- Version 1 is the experiment with the main flow (Step 00 to Step 04) and no actions (see graph drawing in Figure 5.5);
- Version 2 is Version 1 of the experiment with a SetMeasureAll action (Track-In event) in Step 01 to sub-materials 3, 4, 5 and 6 (see graph drawing Figure 5.6);
- Version 3 is Version 2 of the experiment with a ChangeFlowAndStep action (Processed event) in Step 02 to Step 20, with sub-materials 5 and 6, and another flow with Steps 20 to Step 23 (see graph drawing Figure 5.7);
- Version 4 is Version 3 of the experiment with a ChangeFlowAndStep action (Processed event) in Step 23 to Step 04 (back to the main flow), with sub-materials 5 and 6 (see graph drawing Figure 5.8);
- Version 5 is Version 3 of the experiment with a Terminate action (Processed event) in Step 21, with sub-materials 5 and 6 (see graph drawing Figure 5.9);
- Version 6 is Version 3 of the experiment with a TemporaryOfFlow action (Processed event) in Step 04, with sub-materials 3 and 4, and another flow with Steps 30 and Step 31 (see graph drawing Figure 5.10).

For this test, and for every future experiment, the first thing that was done was to get all the information related to the experiment from the database, which includes: 1) The number of sub-materials; 2) The main flow; 3) Every step, from the main flow or not, that has one or more actions. Following this, the retrieved information was parsed and fed to the respective functions of JointJS. As an example, excerpts of the code from Version 1 and Version 2 of the experiment are presented in Figure 5.3 and Figure 5.4 (the rest of the code of this and other versions of the experiment is included in the Appendix A).

To draw the nodes it was used the `shapes.basic.Rect()` function and to draw the links, the `dia.Link()` function. Nodes are represented as rectangular boxes with the name of the step. Nodes with no actions are filled with light green and nodes with actions have a darker shade. Links are blue lines and have arrows indicating the flow direction. A flow is represented by nodes and links. When a flow is a Flow of flows, additional links are used to connect the last node of a flow with the first node of the next one. The color yellow is used for nodes that represent steps that were added due to TemporaryOfFlow actions. The color red is used to signal the presence of logical errors and takes priority over both greens. The visual aspect of the nodes and links serves only as a proof of concept.

```

// API calls
// ...

// Save steps that have an experiment
const stepsExp = outputExpDef.ExperimentDefinition.Steps;

// Save steps of the main flow
const mainFlowId = outputFlow.Flow.Id;
let steps = [];
if (outputFlow.Flow.FlowSteps === undefined) {
  for (let i = 0; i < outputFlow.Flow.SubFlows.length; i++) {
    steps = steps.concat(outputFlow.Flow.SubFlows[i].TargetEntity.FlowSteps);
  }
} else {
  steps = outputFlow.Flow.FlowSteps;
}

// Create Map of Flows (ID, [Step])
const flows = new Map();
flows.set(mainFlowId, []);

// Create graph and paper
const graph = new joint.dia.Graph;
const paper = new joint.dia.Paper({
  el: this.holder.nativeElement,
  model: graph,
  width: 1360,
  height: 500,
  interactive: function (cellView) {
    if (cellView.model instanceof joint.dia.Link) {
      return { vertexAdd: false };
    }
    return true;
  },
});

// Represent main flow
for (let i = 0; i < steps.length; i++) {
  const mainStep = new joint.shapes.basic.Rect({
    position: { x: 60, y: 20 + 100 * i },
    size: { width: 220, height: 30 },
    stepId: steps[i].TargetEntity.Id,
    stepName: steps[i].TargetEntity.Name,
    stepDesc: steps[i].TargetEntity.Description,
    mainLink: undefined,
    materials: [],
    materialsExp: new Map(),
    materialsExpSorted: [],
    actions: new Map(),
    events: new Map(),
  });
  graph.addCell(mainStep);
  flows.get(mainFlowId).push(mainStep);
}

// Add links for main flow
const mainSteps = flows.get(mainFlowId);
for (let i = 1; i < mainSteps.length; i++) {
  const mainLink = new joint.dia.Link({
    source: mainSteps[i - 1],
    target: mainSteps[i],
    materials: [],
    type: 'Main Flow'
  });
  graph.addCell(mainLink);
  mainSteps[i - 1].attributes.mainLink = mainLink;
}

```

Figure 5.3: Code necessary to represent the Version 1 of the experiment

```

// Add experiments to nodes
for (let i = 0; i < stepsExp.length; i++) {
  const materialGroups = stepsExp[i].MaterialGroups;
  const materialsExp = new Map();
  const actionsExp = new Map();
  const eventsExp = new Map();
  for (let j = 0; j < materialGroups.length; j++) {
    if (!materialGroups[j].NoActions) {
      materialsExp.set(j, materialGroups[j].TargetEntity.SubMaterialNumbers.split(';'))
      actionsExp.set(j, []);
      eventsExp.set(j, []);
      for (let k = 0; k < materialGroups[j].Actions.length; k++) {
        actionsExp.get(j).push(materialGroups[j].Actions[k].Action);
        eventsExp.get(j).push(materialGroups[j].Actions[k].Event);
      }
    }
  }
  const nodes = flows.get(stepsExp[i].Flow.Id);
  for (let j = 0; j < nodes.length; j++) {
    if (nodes[j].attributes.stepId === stepsExp[i].Step.Id) {
      nodes[j].attributes.materialsExp = materialsExp;
      nodes[j].attributes.actions = actionsExp;
      nodes[j].attributes.events = eventsExp;
      let materialsExpSorted = [];
      for (let k = 0; k < materialsExp.size; k++) {
        materialsExpSorted = materialsExpSorted.concat(materialsExp.get(k));
      }
      materialsExpSorted.sort();
      nodes[j].attributes.materialsExpSorted = materialsExpSorted;
      nodes[j].attr({
        rect: { fill: actionColor }
      });
    }
  }
}

```

Figure 5.4: Code necessary to represent the Version 2 of the experiment

The graph drawings of Version 1 and Version 2 of the experiment are presented in Figure 5.5 and Figure 5.6, respectively. Because Version 1 has no actions, all nodes are represented in light green, according to the color scheme that was chosen for graph drawings. Version 2 has a SetMeasureAll action in Step 2, that is colored in a darker shade of green.

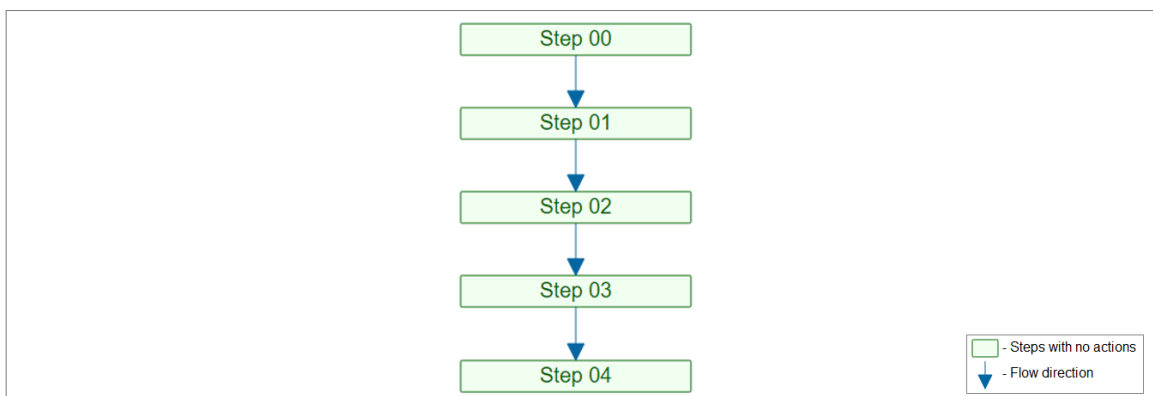


Figure 5.5: Graph drawing of the Version 1 of the experiment (only main flow)

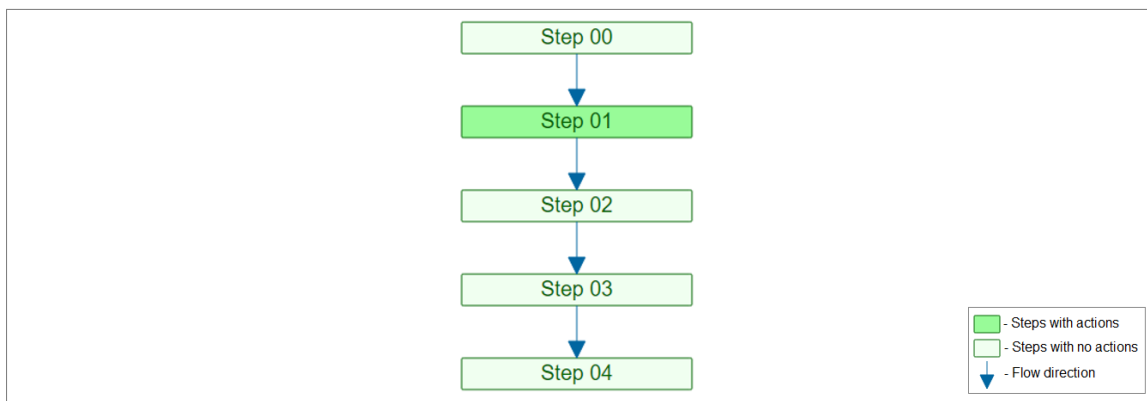


Figure 5.6: Graph drawing of the Version 2 of the experiment (main flow and an action)

In Version 3 of the experiment, it was added an action `ChangeFlowAndStep` to Step 02 for the sub-materials 5 and 6 (see Figure 5.7). This type of action may create a physical split, meaning that one or more sub-materials will follow a different flow from the rest. This can happen for multiple steps if it is merged later, or permanently if there is no future merge, which was the case. Because of this, the nodes were automatically repositioned, giving a more pleasant look to the graph drawing by keeping a hierarchical layout. For the representation itself, it was not enough to draw an extra node with its respective link. It was also needed to draw the rest of the flow for which those sub-materials were changing into. To do this, it was necessary to get the information about the new flow from the database and parse it in a very similar way to the main flow and then represent it. After having the new flow, it was only necessary to add the remaining link from the step that had the action `ChangeFlowAndStep` into the new step from the new flow.

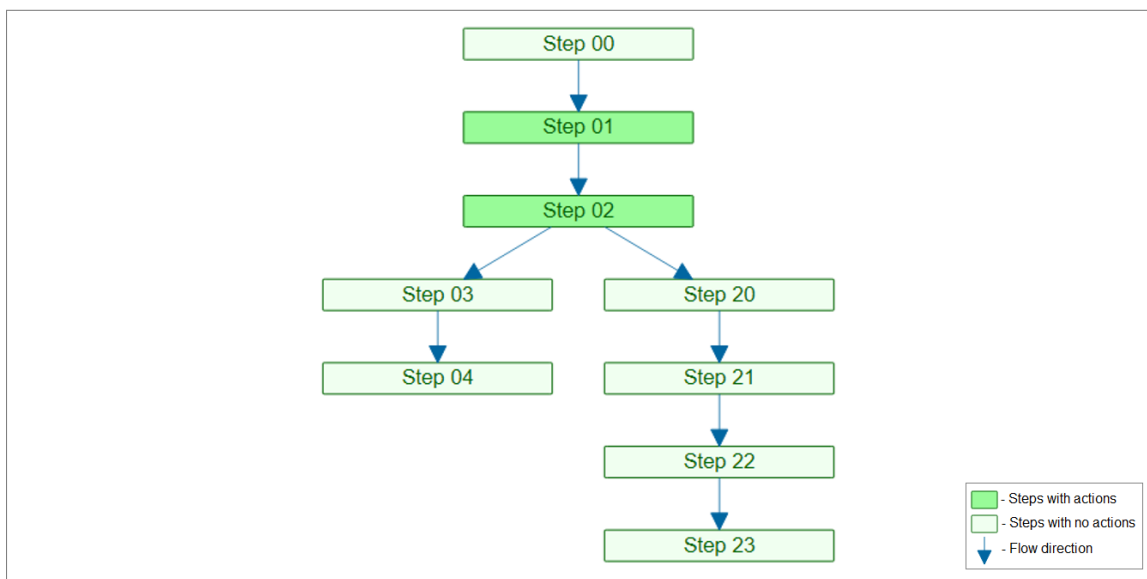


Figure 5.7: Graph drawing of the Version 3 of the experiment (addition of an action causing a split)

In Version 4 of the experiment, an action `ChangeFlowAndStep` was used to merge back the previously split sub-materials 5 and 6 (see Figure 5.8). This action was done in Step 23 to Step 04. Because of that, Step 23 was colored in a darker shade of green. As in the previous version, also this one, the nodes were automatically repositioned, maintaining a layered layout.

As before, it was required to check if both flows were already represented. If one of them or both were not represented, their information had to be retrieved from the database. If both were already represented in the graph, then it was only necessary to add the new link between both flows, which was the case.

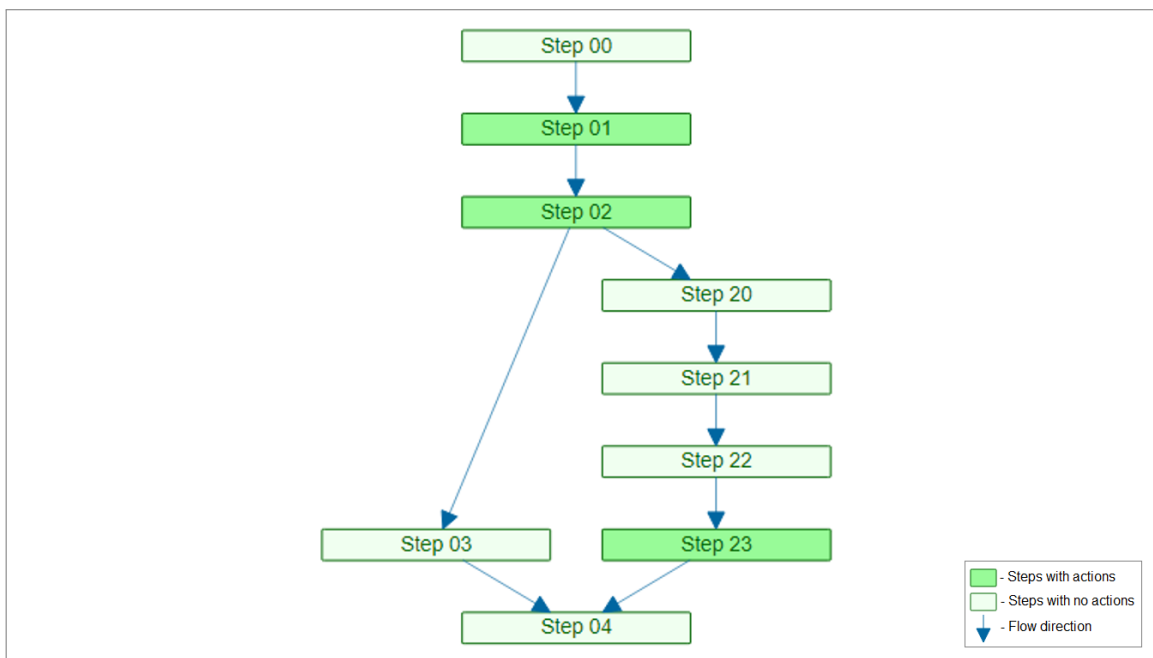


Figure 5.8: Graph drawing of the Version 4 of the experiment (addition of an action causing a merge back from the split)

In Version 5 of the experiment, it was added an action `Terminate` on Step 21 to sub-materials 5 and 6 (see Figure 5.9). If one or more sub-materials are subjected to this action in a step, those sub-materials no longer continue along with the flow, and so this step becomes the last step for these sub-materials. If other sub-materials continue to another step in the flow, this creates a permanent physical split. Because the action `Terminate` applies to all sub-materials on Step 21 (sub-materials 5 and 6), in this case, no split is created.

To represent this in the graph drawing it was necessary to parse this information, just like for the previous actions, but this time instead of adding more components to the graph, some components were hidden in order for the last step of the flow to correspond to the last step with sub-materials.

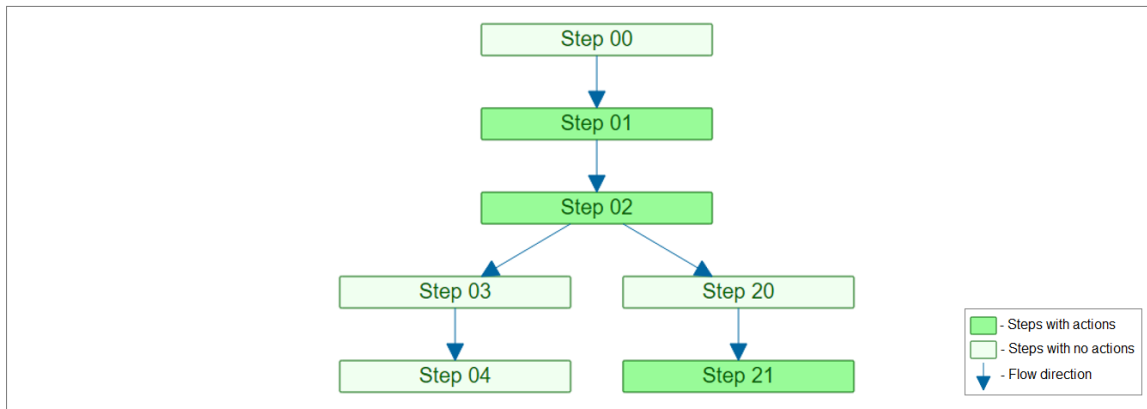


Figure 5.9: Graph drawing of the Version 5 of the experiment (addition of an action Terminate)

In Version 6 of the experiment, it was added an action TemporaryOffFlow on Step 04 to Step 30 with sub-materials 3 and 4 (see Figure 5.10). This action, at first sight, might seem to create a loop in the experiment, because in the drawing, there will be a link from the step that has the action (Step 04) to another step in another flow (Step 30), and in the last step of that other flow (Step 31), there will be a link back to the step that had the action (Step 04). According to the color scheme, Step 04 is filled in a darker green, because it has the action TemporaryOffFlow. Step 30 and Step 31 are colored in light yellow because they represent a flow that was added due to a TemporaryOffFlow. In the drawing itself, this is indeed a cycle, but the CMF EM module already ensures that this action does not create any type of logical problem during the execution of the experiment. To represent this action in the drawing the same procedure of parsing information from the database was followed.

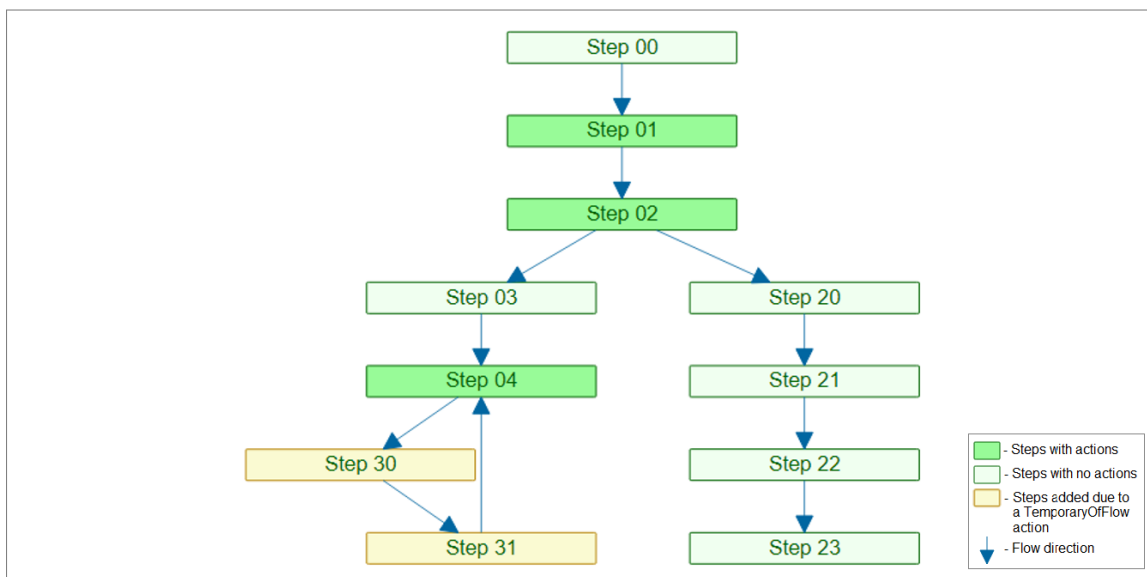


Figure 5.10: Graph drawing of the Version 6 of the experiment (addition of an action Temporary-OffFlow)



### 5.3 Logical Verification

To analyse if the experiment is logically valid, it was required to save in every node information regarding the sub-materials that have at least one action in that step and the sub-materials that pass through that step. This information is stored in the variables described in Table 5.1.

Table 5.1: Node Data Structure

| NAME               | DESCRIPTION  |
|--------------------|--|
| stepId             | Id of the step   |
| stepName           | Name of the step   |
| stepDescription    | Description of the step  |
| mainLink           | Next link in the flow, if it exists                            |
| changeLinks        | Set of other links (links due to the action ChangeFlowAndStep) |
| tempLinks          | Set of other links(links due to the action TemporaryOffFlow)   |
| subMaterials       | Set of sub-materials that pass through                         |
| subMaterialsAction | Set of sub-materials with actions                              |
| actions            | Set of actions   |
| events             | Set of events  |

The information about the sub-materials that have at least one action in a step was already stored in their respective nodes because it was necessary for the graph drawing. For the sub-materials that pass through a step, the process was more complex and required some graph properties to be used. To achieve this, an algorithm to traverse the graph depth-first was implemented.

The depth-first algorithm implemented is a recursive algorithm that accepts a node and a set of sub-materials as parameters. When called for the first time, the parameters should be the root node (first step of the main flow) and a set of every sub-material in that experiment. This algorithm does the following operations:

- Checks if the sub-materials trying to be saved in a node  $x$  are not already there; if they are not, it saves the sub-materials in the node  $x$ ;
- Checks if there are tempLinks in the node  $x$ ; if there are, an auxiliary function is called, because this case needs to be addressed differently;
- Checks if there is any Terminate action in the node  $x$ , if there is, the terminated sub-materials  $T$  are removed from the set of sub-materials that are going to next node  $y$  in the flow;
- Checks if there are changeLinks in the node  $x$ ; if there are, the sub-materials  $C$ , going to a step  $a$  in another flow, are removed from the set of sub-materials that are going to next node  $y$  in the flow, and recursively call this function with that step  $a$  and sub-materials  $C$ ;

- Recursively call this function with the next node in the flow, if it exists, and with the remaining sub-materials, if any.

After having all the information needed in the nodes, it is only necessary to do a few checks to test if the experiment is logically valid or not. There are two types of logical errors that can occur, a circular reference or a missing sub-material.

- A missing sub-material occurs when an action is added to a node, for a certain sub-material, and that sub-material does not pass through that node. So, missing sub-material type of errors only occur in nodes that have actions.
- A circular reference occurs when a sub-material is sent, via `ChangeFlowAndStep` action, to a node where it already passed through, creating a cycle in the graph. So, circular reference types of errors can occur in nodes with or without actions.

To check for circular references, it was added a verification in the previous algorithm during the first condition; if there is a sub-material trying to be saved in a node where it was already saved, this indicates that there is a cycle, which generates an error. To check for a missing sub-material, after the conclusion of the previous algorithm, it is checked, for every node, if the `subMaterialsAction` set is contained in the `subMaterials` set; if not, then there is a missing sub-material in that node. When one of these errors is identified, that node turns red and an error message is sent to the error tab with information regarding the step, the sub-material, and the type of the error so that the user can fix it easily.

Every node has stored every link that has that node as a source, and every link stores its target node. There are two types of links, that vary due to their origin, one type of link is those that are apart of a flow and connect their steps in a sequence, the other type is the one that originates from the action `ChangeFlowAndStep`, which connect steps from different flows. So while traversing the graph, for each node, first the set of sub-materials is saved in that node, after that, it will be determined which sub-materials go to each node, so it is checked if there is any `ChangeFlowAndStep`, then check if there is an action `Terminate`, and the remaining sub-materials are sent to the next node of the main flow if any.

## 5.4 Improvement of User Features

Some complementary features were added to this module to enable the user to have a better experience while using it.

- The user can filter the graph by a specific sub-material to see the graph of only that one sub-material, to ensure that it is as it was supposed to be. The filter can also be used for more than one sub-material, in the same group or not, showing the graph of all of them combined (filtering for everything yields the same result as filtering for nothing). This feature is achieved by traversing every node and checking if at least one of the sub-materials being filtered is contained in either the `subMaterials` set or `subMaterialsExp` set.

- JointJS already comes with a feature that allows users to be able to grab and hold nodes, to change their position on the paper in real-time, which is useful for graph drawings of DoE.
- It was added the feature to drag the outside of the graph to create the effect of panning. To implement this, it was used the already exposed events, `blank:pointerdown`, `blank:pointerup` and `cell:pointerup`. The first is used to set the start position and the last two are used to clean it. Also, it was used a regular javascript event listener, that takes into consideration the start position, the current position, and the current scale value and applies a translate to the graph accordingly.
- It was also implemented a zoom function which helps the user to better visualize experiments with a great number of nodes. As mentioned on the pan feature, JointJS also exposes the event for the `blank:mousewheel` and `cell:mousewheel`, so a scale is applied to everything on the paper according to the rotation of the wheel. It is also done a translate that takes into consideration the position of the mouse so that it stays in the same position of the graph.
- Nodes and links are clickable, and when clicked, information about them is shown in a side tab (this component was adapted from an existing CMF MES component to keep the same look and feel). The shown information is already stored in the elements clicked, so it is only necessary to write it in the designated sections of the side tab.
- When there is a logical error, information appears in a bar showing what is causing that error, including the type of error, the sub-material, and the node. By clicking on it, it will automatically filter by that sub-material to streamline the process of detecting what might be originating the error. The implementation of this was already described in [5.3](#).
- Unused nodes are hidden to lighten up the complexity of the drawing. This can be toggled in case the user wants to check something in those unused nodes. This is achieved by the same methods used in the filter.

## 5.5 Testing and Validation

To test the code developed, the missing sub-material type of errors and the circular reference type of errors were generated in the Version 3 of the experiment tested in graph visualization (see graph drawing of Version 3 in [Figure 5.5](#)). The generation of these errors was based on the fact that, in Version 3, sub-materials 5 and 6 are sent to step 20, due to a `ChangeFlowAndStep` on Step 02. The purpose of creating these errors was to check if the code was able to detect and catch them and show the appropriate feedback. For validation, the results were analysed by in-house experts, which is the method that CMF uses in its projects.

For generating a missing sub-material error in Version 3 of the tested experiment in graph visualization, a `SetMeasureAll` action was added for sub-materials 5 and 6 on step 03. Because of the split on step 02, these sub-materials did not pass through Steps 03 and 04. This generates

a logical error, step 03 turns red (see Figure 5.11) and two error messages saying "Trying to use missing sub-material 5 on Step 03" and "Trying to use missing sub-material 5 on Step 03" are displayed in the errors tab. By filtering by either sub-material 5 or 6 or both, step 03 appears without any link. With this information, the user should quickly realize what may be causing this error (see Figure 5.12).

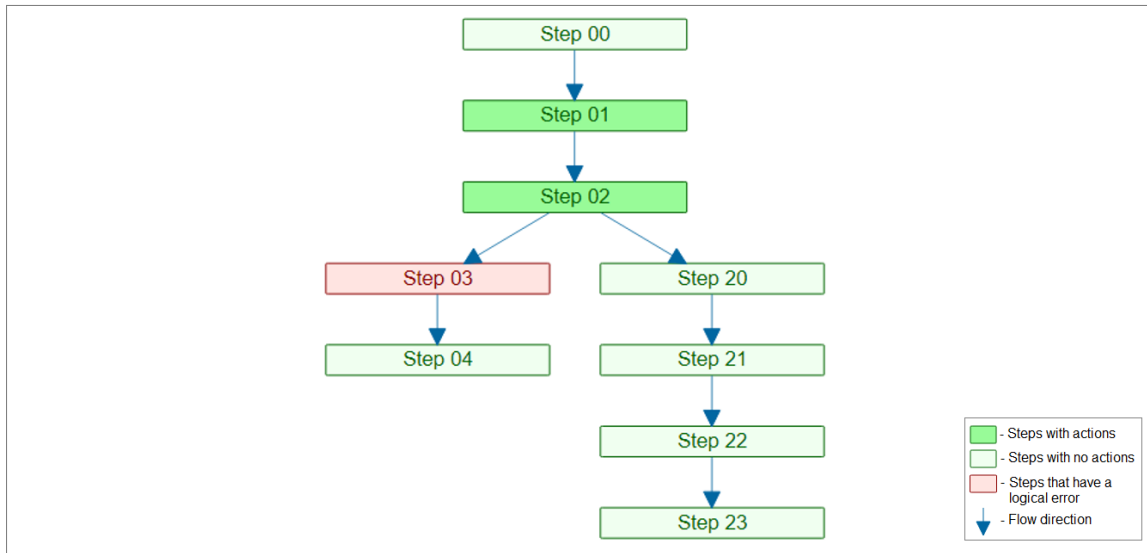


Figure 5.11: Graph drawing of the experiment with a missing sub-material error

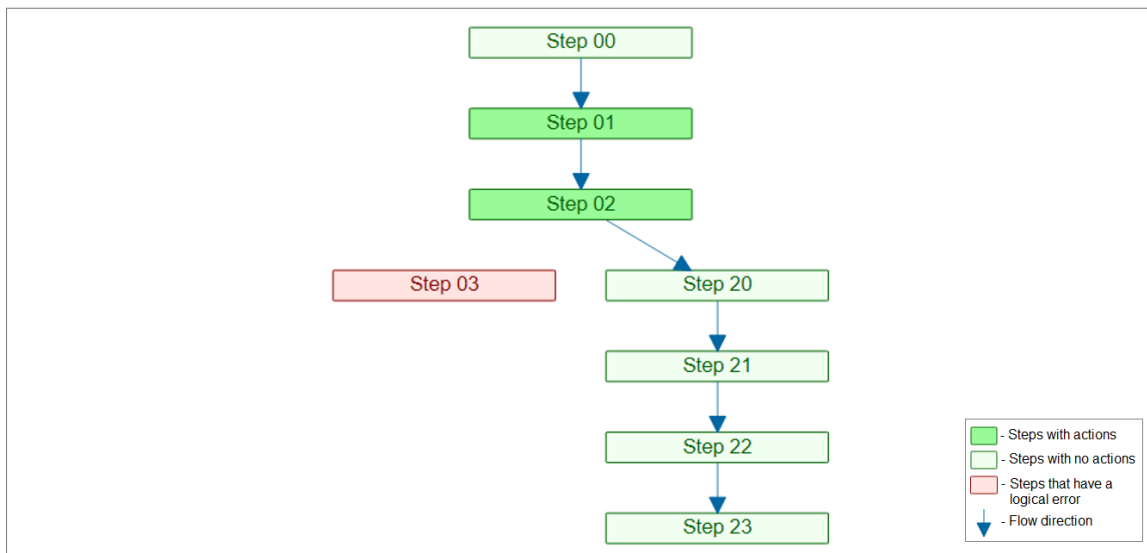


Figure 5.12: Graph drawing of the experiment with a missing sub-material error filtered by the by the sub-material causing the error

For generating a circular reference error in Version 3 of the tested experiment in graph visualization, a ChangeFlowAndStep action on step 21 to step 00 was added for sub-materials 5 and 6.

Because these sub-materials have already passed through step 00 before, this generates a logical error. Step 00 turns red (see Figure 5.13) and two error messages saying "Circular reference of sub-material 5 on Step 00" and "Circular reference of sub-material 5 on Step 00" are displayed in the errors tab. By filtering by either sub-material 5 or 6 or both, step 03 appears without any link. With this information, the user should quickly realize what may be causing this error (see Figure 5.14). The in-house experts found these results in accordance with what was intended.

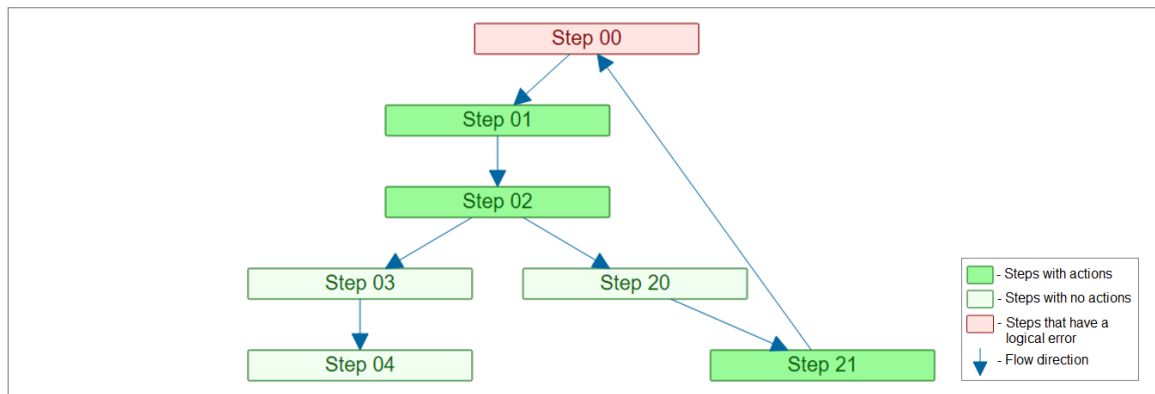


Figure 5.13: Graph drawing of the experiment with a circular reference error

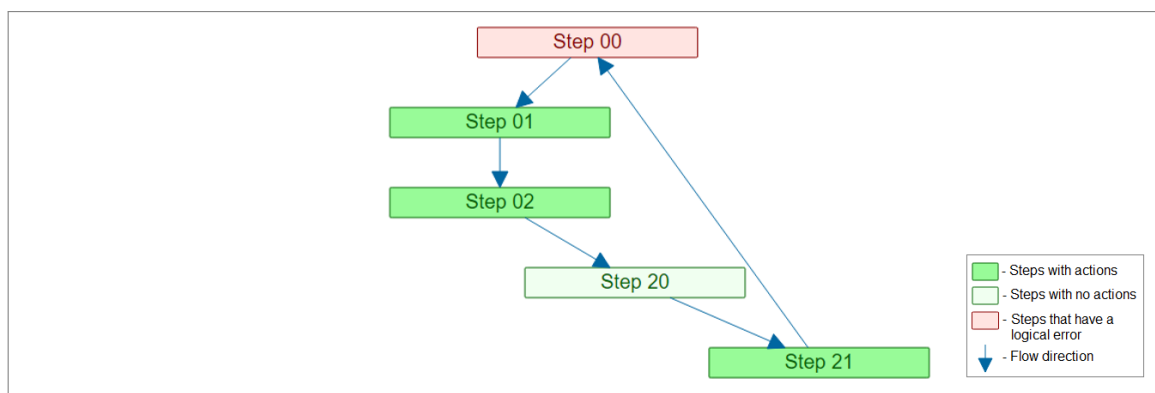


Figure 5.14: Graph drawing of the experiment with a circular reference error filtered by the sub-material causing the error

## 5.6 Summary

This chapter was dedicated to the description of the methodology used for the development of the solution. Of the presented results, it is important to highlight that:

- Users stories were taken into consideration for the definition of the objectives of this dissertation.

- A set of tasks that allowed the accomplishment of the objectives were defined and organized in stages.
- For the development of the solution, an agile software process model, more specifically the SCRUM framework was chosen.
- The development of the solution regarding the graph visualization, the logical verification, the improvement of user features and the testing and validation were described and illustrated resorting to a simple experiment with six versions.

## Chapter 6

# Validation and results analysis

This chapter presents the validation and analysis of the results of this dissertation. Section 6.1 presents the graph drawing of a complex experiment. Section 6.2 illustrates the use of the implemented user features with the graph drawing of the complex experiment. Section 6.3 includes graph drawings with different errors, illustrating their detection in a complex experiment. Section 6.4 summarizes the highlights of this chapter.

### 6.1 Graph drawing of a complex experiment

The complex experiment that illustrates the implemented solution for drawing graphs was defined and created in the CMF EM module for this dissertation.

In this experiment, there are six sub-materials (1, 2, 3, 4, 5, 6) organized in three groups of two sub-materials each (1 and 2, 3 and 4, 5, and 6). There is a main flow with five steps (Step 0 to Step 4), but also other flows (Step 5 to Step 8, Step 18 to Step 19 and Step 20 to Step 23) and a Flow of flows (Step 9 to Step 17). As for the actions, there are six actions ChangeFlowAndStep, two actions SetMeasureAll, one action TemporaryOffFlow, and one action Terminate. The steps that have these actions are Step 1, Step 4, Step 6, Step 8, Step 11, Step 13, Step 14, and Step 19. Step 1 has two actions ChangeFlowAndStep, one for sub-materials 3 and 4, and other for sub-materials 5 and 6. Step 4 has one action ChangeFlowAndStep for sub-materials 1 and 2. Step 6 has one action SetMeasureAll for sub-materials 5 and 6. Step 8 has one action ChangeFlowAndStep for the sub-materials 5 and 6. Step 11 has one action TemporaryOffFlow for the sub-materials 3 and 4. Step 13 has one action Terminate for the sub-material 3. Step 14 has one action SetMeasureAll for the sub-material 4. Step 19 has one action ChangeFlowAndStep for sub-materials 1, 2, 5, and 6.

In Table 6.1, the information of steps, sub-materials, and actions is summarized.

Table 6.1: Characteristics of the complex experiment

| STEPS | SUB-MATERIALS    | ACTIONS (SUB-MATERIALS)                            |
|-------|------------------|--|
| 0     | 1, 2, 3, 4, 5, 6 | —  |
| 1     | 1, 2, 3, 4, 5, 6 | ChangeFlowAndStep (3, 4); ChangeFlowAndStep (5, 6) |
| 2     | 1, 2             | —  |
| 3     | 1, 2             | —  |
| 4     | 1, 2             | ChangeFlowAndStep (1, 2)                           |
| 5     | 5, 6             | —  |
| 6     | 5, 6             | SetMeasureAll (5, 6)                               |
| 7     | 5, 6             | —  |
| 8     | 5, 6             | ChangeFlowAndStep (5, 6)                           |
| 9     | —                | —  |
| 10    | 3, 4             | —  |
| 11    | 3, 4             | TemporaryOffFlow (3, 4)                            |
| 12    | 3, 4             | —  |
| 13    | 3, 4             | Terminate (3)                                      |
| 14    | 4                | SetMeasureAll (4)                                  |
| 15    | 4                | —  |
| 16    | 1, 2, 4, 5, 6    | —  |
| 17    | 1, 2, 4, 5, 6    | —  |
| 18    | 1, 2, 5, 6       | —  |
| 19    | 1, 2, 5, 6       | ChangeFlowAndStep (1, 2, 5, 6)                     |
| 20    | —                | —  |
| 21    | —                | —  |
| 22    | 3, 4             | —  |
| 23    | 3, 4             | —  |

The graph drawing of the complex experiment characterized in Table 6.1 is presented in Figure 6.1. When visualizing it, the user has also available information about the sub-materials and actions at each node by clicking on it. As mentioned previously, for a certain sub-material, nodes with actions are colored in a darker green, and nodes that were added due to a TemporaryOffFlow action are colored in light yellow. The two actions ChangeFlowAndStep in Step 1 split the sub-materials by three flows, the main flow and two other flows. Two of the three groups of sub-materials are merged back by actions ChangeFlowAndStep in Step 8 and Step 4. The other group has a TemporaryOffFlow in Step 11, thus diverting to Step 22 and Step 23, and only then proceeding to Step 12. In this group, sub-material 3 has an action Terminate in Step 13 and do not pass by the following nodes. The other two groups of sub-materials are merged back with the remaining sub-material from this one, by an action ChangeFlowAndStep in Step 19.



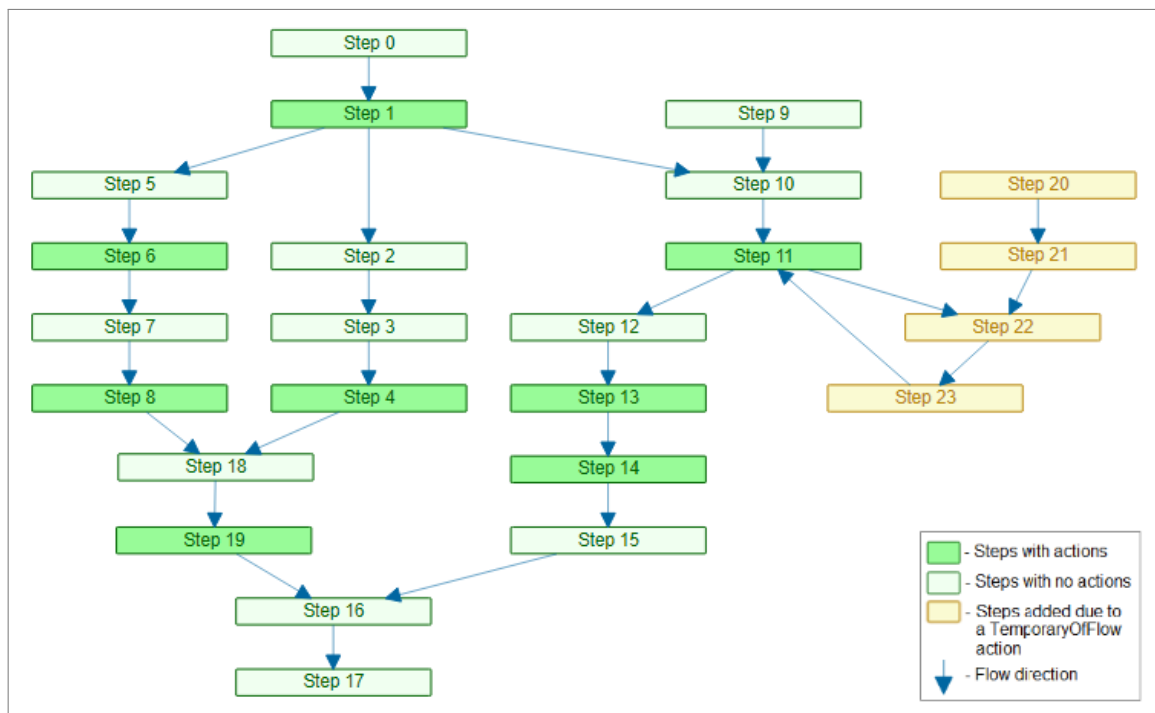


Figure 6.1: Graph drawing of the complex experiment

The graph drawing in Figure 6.1 shows that the implemented solution produces adequate graph drawings. In this case, the graph drawing is a hierarchical drawing of a directed acyclic graph. As recommended, upward links are avoided (the exception is a link due to the action TemporaryOffFlow), nodes are uniformly distributed, there are no crossings and links are straight lines.

## 6.2 Application of user features

Although the graph drawing in Figure 6.1 is an adequate visualization of the complex experiment described in Table 6.1, some additional features can improve its effectiveness, making it more useful for users. According to the information in Table 6.1, some nodes do not have sub-materials (Step 9, Step 20, and Step 21). This means that they are part of a flow that is being used in this experiment, even though these particular steps are not. A feature was implemented to allow users to hide the nodes that are not being used, *i.e.*, those that do not have sub-materials nor actions. Figure 6.2 illustrates the use of this feature by hiding the nodes corresponding to Step 9, Step 20, and Step 21.

Another feature available to users is the possibility to grab and hold the nodes, changing manually the layout of the graph drawing. For example, in the automatically generated graph drawing of Figure 6.2, the nodes of Steps 10 to 17 are not vertically aligned, despite being part of the same Flow of flows. The user may be interested in changing this, positioning the nodes in a vertical straight line and thus facilitating the identification of the Flow of flows. Figure 6.3 shows the same graph drawing of Figure 6.2 but with some nodes positioned by the user.

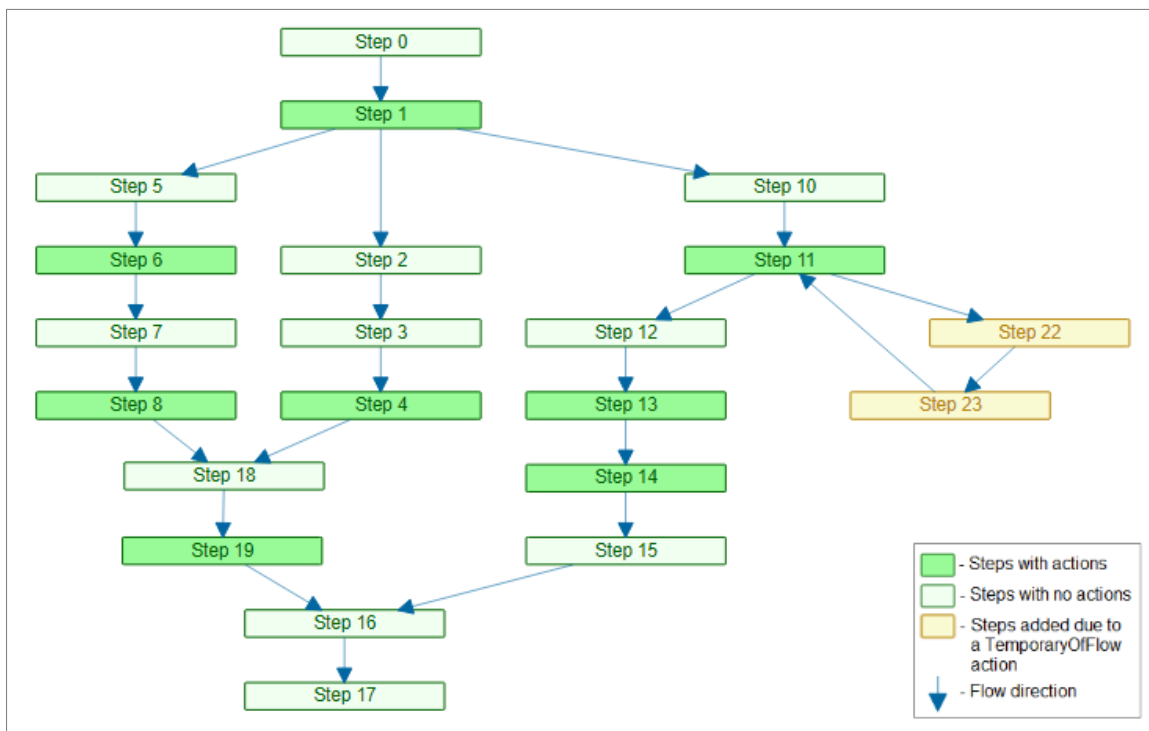


Figure 6.2: Graph drawing of the complex experiment with the unused nodes hidden

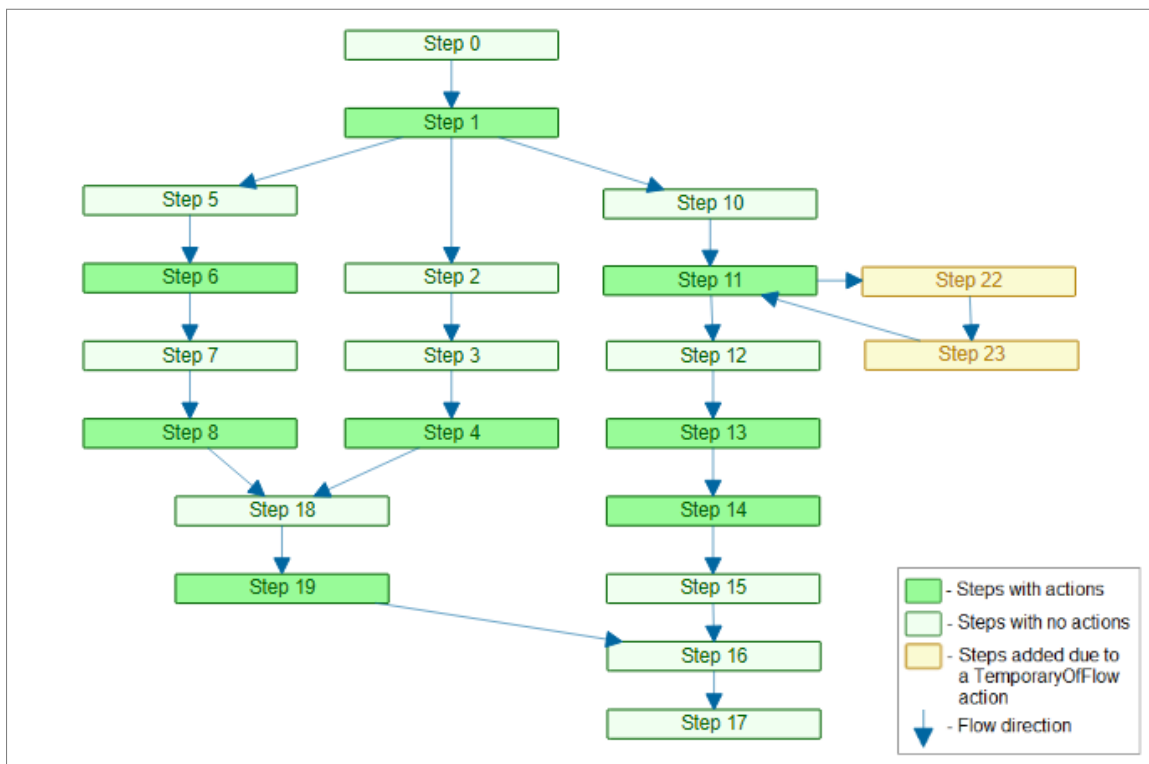


Figure 6.3: Graph drawing of the complex experiment with nodes position changed by the user

The complexity of the graph drawing can be decreased by not visualizing the complete drawing. This is accomplished with the possibility of filtering the graph drawing by sub-materials, showing the nodes and links of a specific sub-material or set of sub-materials, and recoloring the nodes accordingly. Figures 6.4, 6.5, 6.6, 6.7 and 6.8 illustrate the use of this feature by only showing the nodes and links of sub-materials 1, 3, 4, 3 and 4, and 5, respectively. The graph drawing filtered by sub-material 2 is identical to the one of sub-material 1, the same happening with sub-material 6 and 5. Because of that, they were not included here.

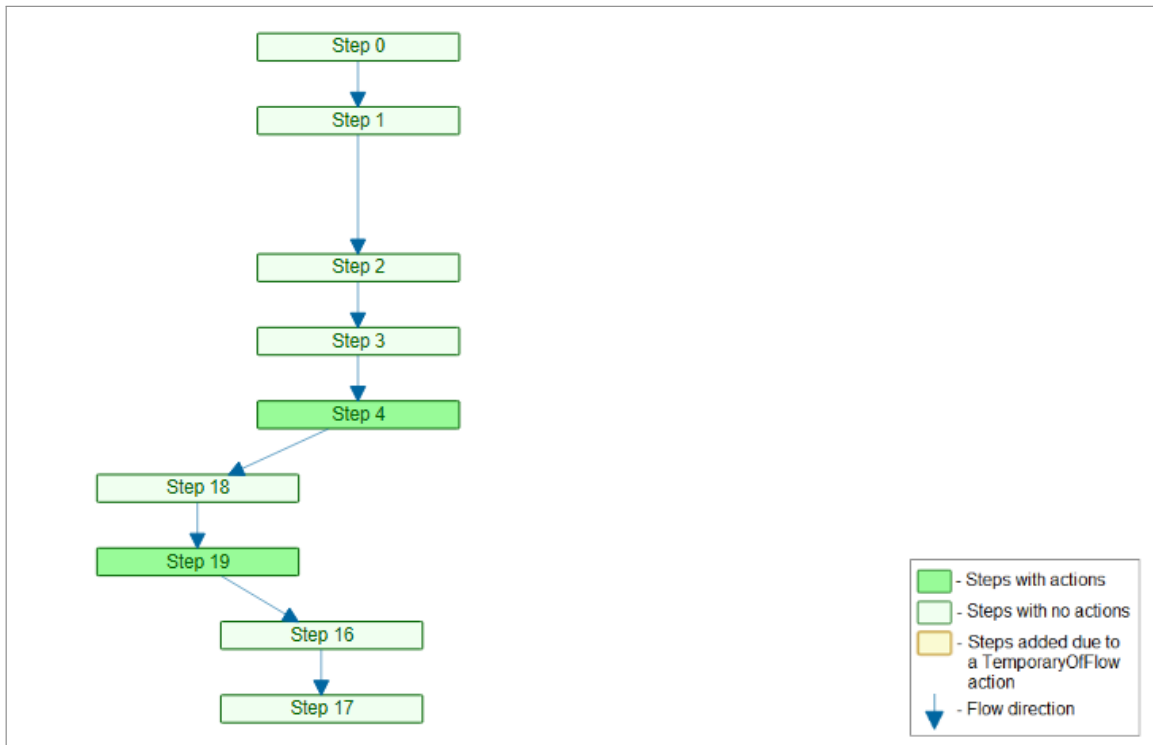


Figure 6.4: Graph drawing of the complex experiment filtered by sub-material 1

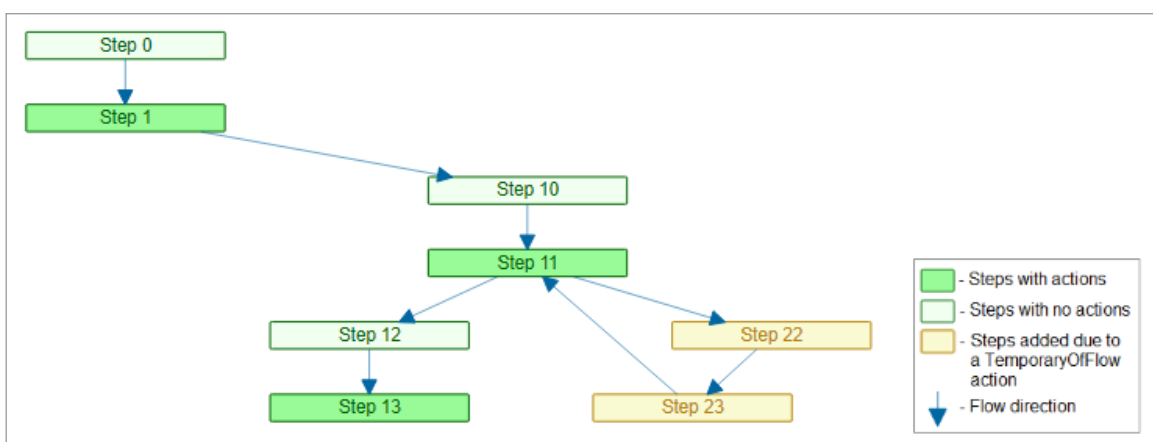


Figure 6.5: Graph drawing of the complex experiment filtered by sub-material 3

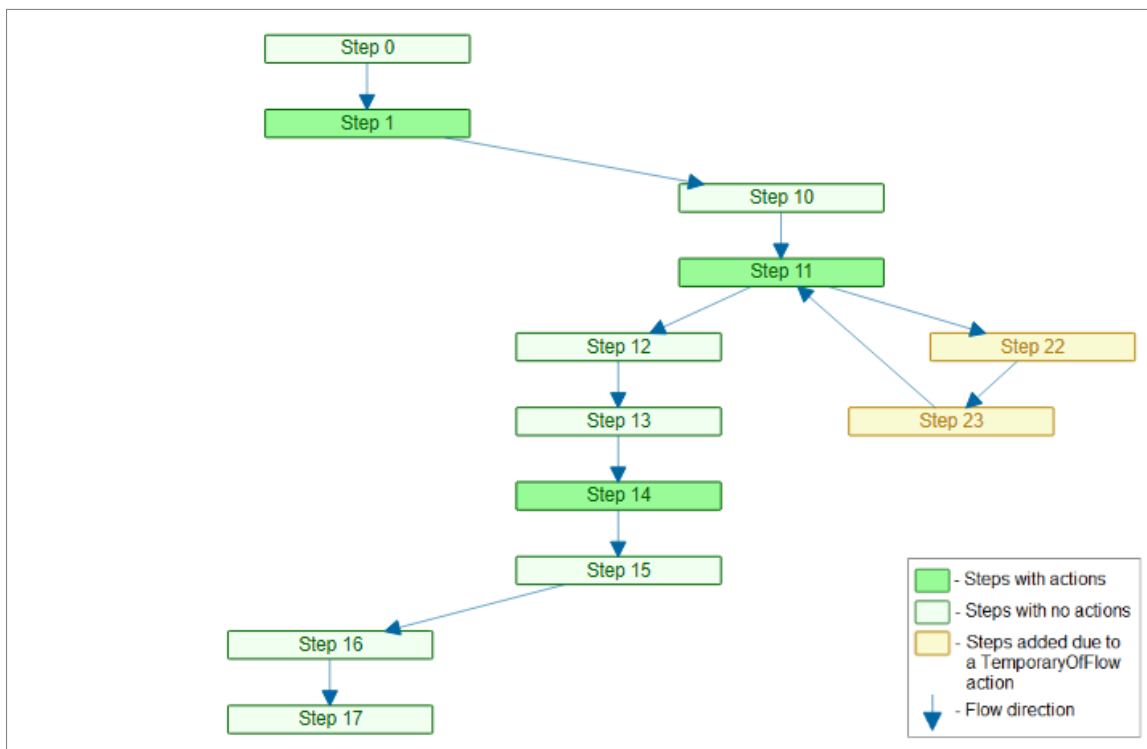


Figure 6.6: Graph drawing of the complex experiment filtered by sub-material 4

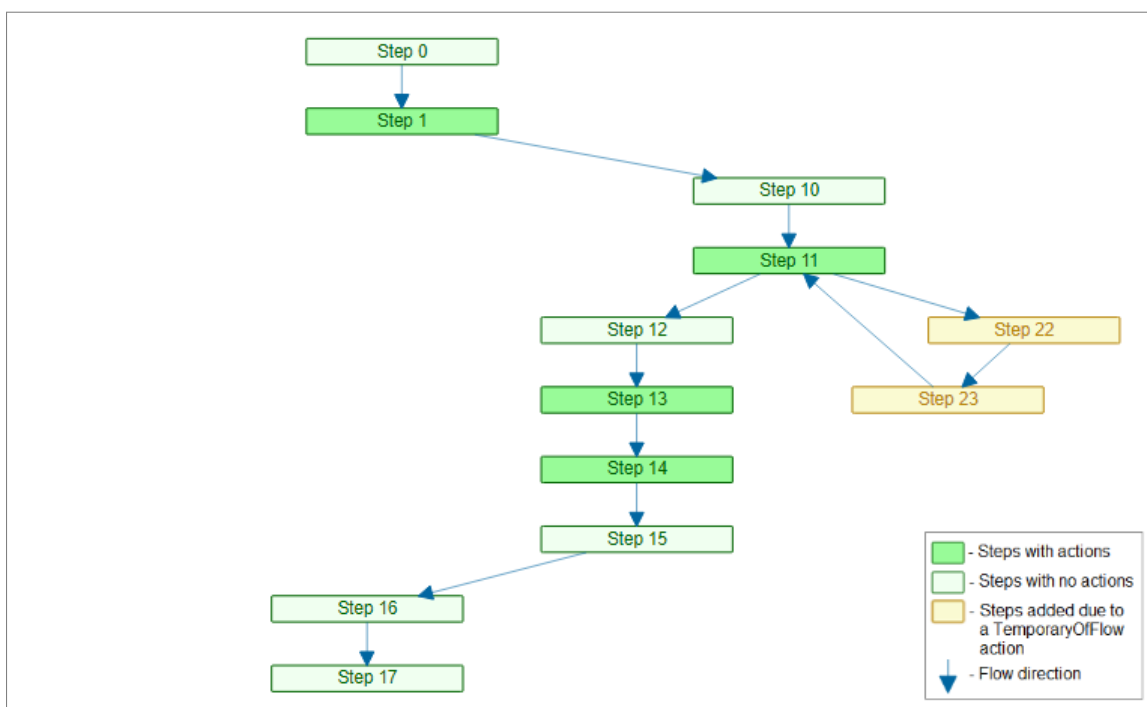


Figure 6.7: Graph drawing of the complex experiment filtered by a group of sub-materials (3 and 4)

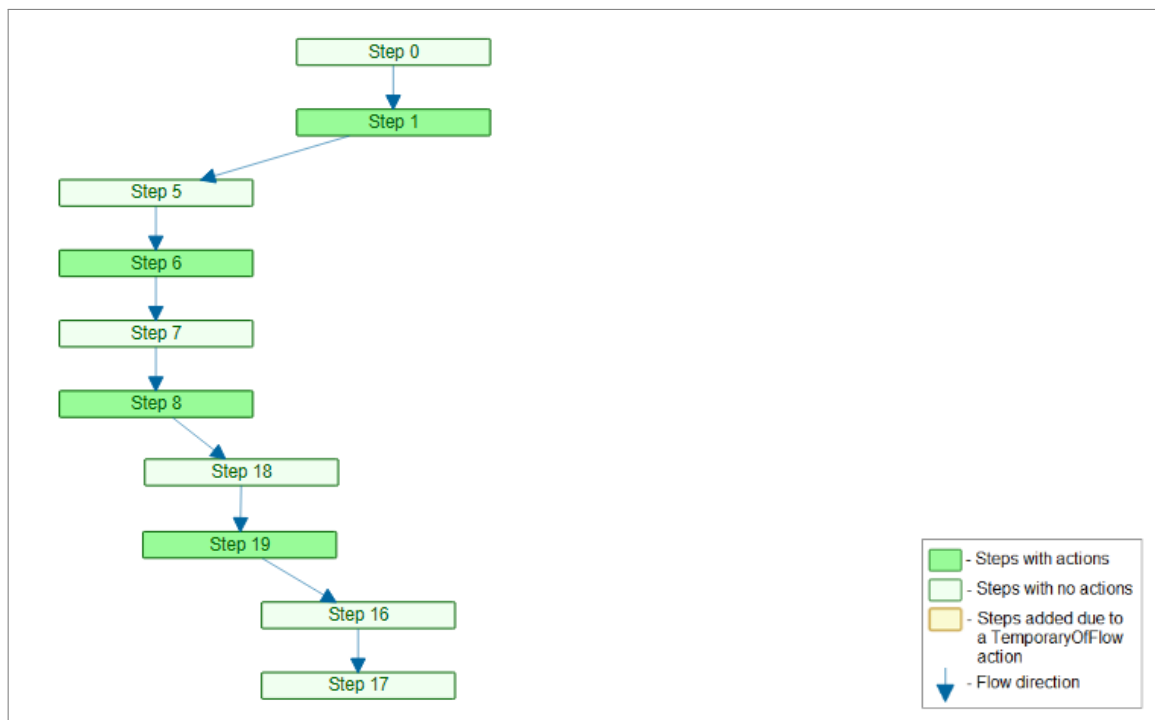


Figure 6.8: Graph drawing of the complex experiment filtered by sub-material 5

The observation of the graph drawings in this section shows that the implemented solution allows users, not only to visualize all the nodes, links, and actions involved in a complex experiment but also visualize more specific information regarding a particular sub-material or set of sub-materials. These sub-materials can be from the same group or not, being available the possibility to chose sub-materials individually or to chose a group of sub-materials. It also shows that the colors of the nodes change according to the sub-material being filtered. Figure 6.4 is the only one where Step 1 is light green because it is filtered by the sub-material 1 and this sub-material does not have an action in this node. In all graph drawings, filtered or not, the user has available information about the sub-materials and actions at each node by clicking on it.

### 6.3 Detection and visualization of errors

To illustrate the capability of the implemented solution in the detection and visualization of errors, changes were made to the complex experiment to create different errors of the two possible types (missing sub-material and circular reference).

For the missing sub-material type of error, different actions for sub-material 1, were added to five nodes (Step 06, Step 08, Step 11, Step 13, and Step 15). Four of these nodes (Step 06, Step 08, Step 11, and Step 13) already had actions for other sub-materials. Step 15 had no other action than the one added to create this error. An action `SetMeasureAll` was added to Step 6 and Step 15 and an action `ChangeFlowAndStep` was added to Step 8, Step 11, and Step 13. Hence, Step 6 had two actions `SetMeasureAll`, one for sub-materials 5 and 6 and other for sub-material 1, Step

8 had two actions ChangeFlowAndStep, one for sub-materials 5 and 6 and other for sub-material 1, Step 11 had one action TemporaryOffFlow for sub-materials 3 and 4 and one action ChangeFlowAndStep for sub-material 1, Step 13 had one action Terminate for sub-material 3 and one action ChangeFlowAndStep for sub-material 1 and Step 15 had one action ChangeFlowAndStep for sub-material 1. All the additional actions for sub-material 1 were added to nodes by which the sub-material 1 did not pass through, thus causing an error as it was intended.

In Table 6.2, the information about the additional actions and respective sub-materials causing the missing sub-material type of error is presented with the information about the steps, sub-materials, and actions of the complex experiment.

Table 6.2: Complex experiment with additional actions causing missing sub-material errors

| STEPS | SUB-MATERIALS    | ACTIONS (SUB-MATERIALS)                              | ADDITIONAL ACTIONS    |
|-------|------------------|--|-----------------------|
| 0     | 1, 2, 3, 4, 5, 6 | —  | —                     |
| 1     | 1, 2, 3, 4, 5, 6 | ChangeFlowAndStep (3, 4)<br>ChangeFlowAndStep (5, 6) | —<br>—                |
| 2     | 1, 2             | —  | —                     |
| 3     | 1, 2             | —  | —                     |
| 4     | 1, 2             | ChangeFlowAndStep (1, 2)                             | —                     |
| 5     | 5, 6             | —  | —                     |
| 6     | 5, 6             | SetMeasureAll (5, 6)                                 | SetMeasureAll (1)     |
| 7     | 5, 6             | —  | —                     |
| 8     | 5, 6             | ChangeFlowAndStep (5, 6)                             | ChangeFlowAndStep (1) |
| 9     | —                | —  | —                     |
| 10    | 3, 4             | —  | —                     |
| 11    | 3, 4             | TemporaryOffFlow (3, 4)                              | TemporaryOffFlow (1)  |
| 12    | 3, 4             | —  | —                     |
| 13    | 3, 4             | Terminate (3)  | Terminate (1)         |
| 14    | 4                | SetMeasureAll (4)                                    | —                     |
| 15    | 4                | —  | SetMeasureAll (1)     |
| 16    | 1, 2, 4, 5, 6    | —  | —                     |
| 17    | 1, 2, 4, 5, 6    | —  | —                     |
| 18    | 1, 2, 5, 6       | —  | —                     |
| 19    | 1, 2, 5, 6       | ChangeFlowAndStep (1, 2, 5, 6)                       | —                     |
| 20    | —                | —  | —                     |
| 21    | —                | —  | —                     |
| 22    | 3, 4             | —  | —                     |
| 23    | 3, 4             | —  | —                     |

Figure 6.9 illustrates how these errors are visualized in the graph drawing of the complex experiment. As expected, the color of five nodes turned red and five error messages were displayed in the errors bar: "Trying to use missing sub-material 1 on Step 6", "Trying to use missing sub-material 1 on Step 8", "Trying to use missing sub-material 1 on Step 11", "Trying to use missing sub-material 1 on Step 13", and "Trying to use missing sub-material 1 on Step 15". Clicking on any of the error messages automatically filters the graph drawing by the sub-material causing the error indicated in the message. The resulting graph drawing is shown in Figure 6.10. The same graph drawing could be obtained by applying a filter to the graph drawing shown in Figure 6.9 and selecting only the sub-material 1. As previously stated, the color of the nodes changed according to the actions that are being applied to the filtered sub-material. The color red always takes precedent over the other colors. Since the missing sub-material type of error occurs by applying actions to certain sub-materials, in nodes in which these sub-materials do not pass through, this means that these nodes were not a part of the path of that sub-material. Thus, this type of error in a graph drawing filtered by the sub-material that caused it, can be identified not only by its red color but also because it corresponds to isolated nodes, without any links to the remaining ones. This can be seen by comparing figure 6.4, without errors, with figure 6.10, with errors, both from the complex experiment and both filtered by sub-material 1.

The in-house experts found these results in accordance with what was intended.

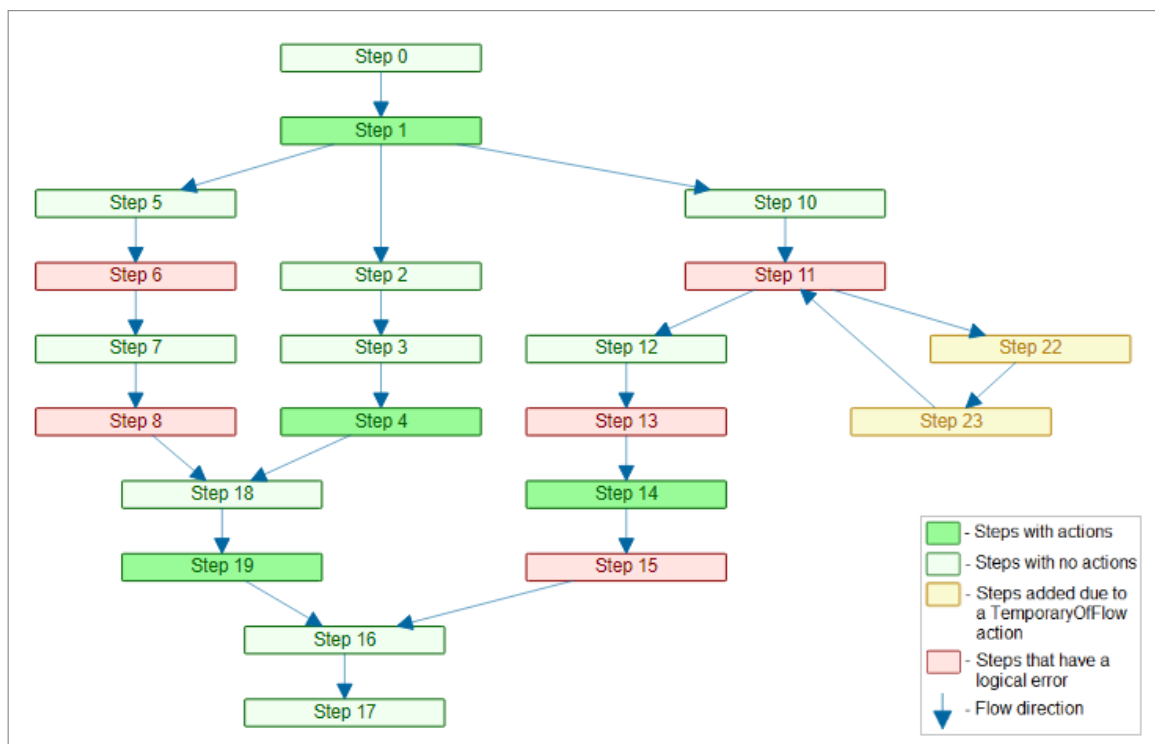


Figure 6.9: Graph drawing of the complex experiment with missing sub-material errors

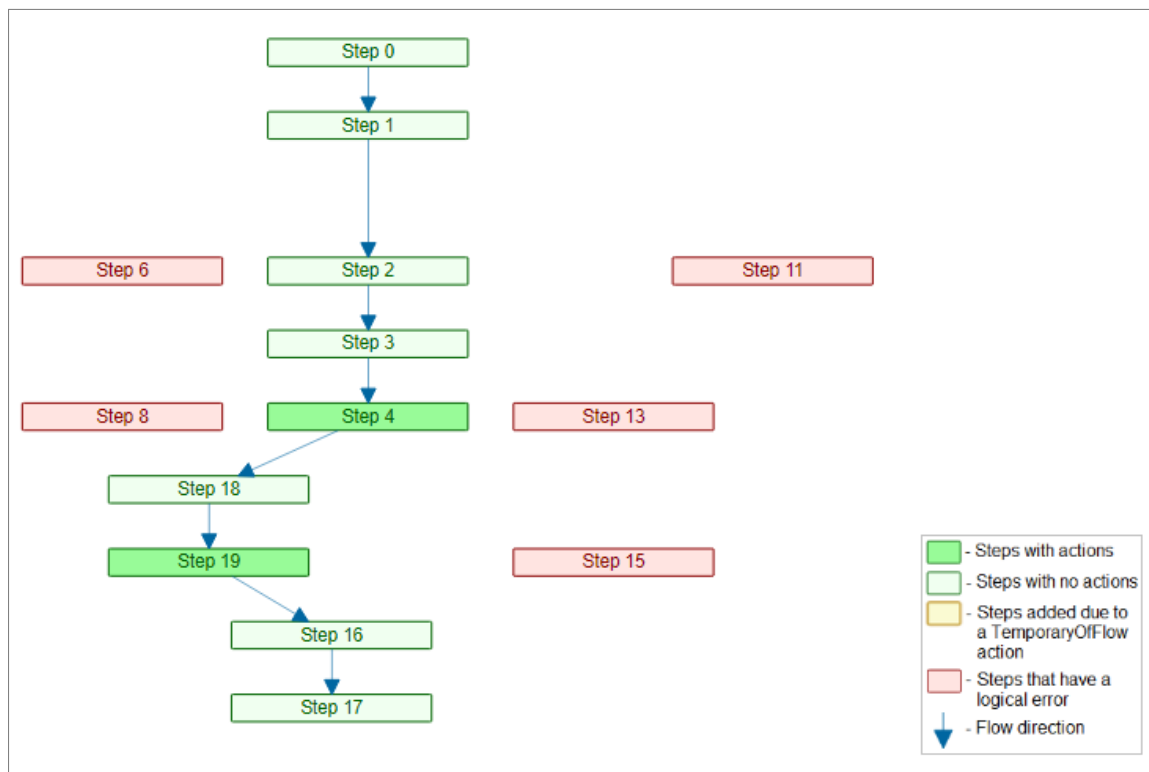


Figure 6.10: Graph drawing of the complex experiment with missing sub-material errors being filtered by the sub-material causing the errors (sub-material 1)

For the circular reference type of error, two actions `ChangeFlowAndStep` were applied to different sub-materials (2 and 4) passing through different nodes (Step 17 and Step 10). These steps had no actions in the complex experiment. Sub-material 2 passed through Step 17 and sub-material 4 passed through Step 10, thus making it possible for these actions to be applied to these steps. One action `ChangeFlowAndStep` from Step 17 to Step 19 was applied to sub-material 2, and one action `ChangeFlowAndStep` from Step 10 to Step 0 was applied to sub-material 4. The added actions `ChangeFlowAndStep` made sub-materials 2 and 4 pass through the selected nodes twice, thus causing errors due to circular reference type errors, as it was intended. Sub-material 4, Step 10, and Step 0 were intentionally chosen to create a circular reference type of error, but also missing sub-material errors due to the change of the respective path. Because of this path alteration, sub-material 4 will not pass through Step 11, thus returning an error due to the an action in that step. Similarly, sub-material 4 will also not pass through Step 14, thus returning an error due to an action in that step.

In Table 6.3, the information about the additional actions and respective sub-materials causing the circular reference and respective missing sub-materials type of errors is presented with the information about the steps, sub-materials, and actions of the complex experiment.



Table 6.3: Complex experiment with additional actions causing circular reference and respective missing sub-materials errors

| STEPS | SUB-MATERIALS    | ACTIONS (SUB-MATERIALS)                              | ADDITIONAL ACTIONS    |
|-------|------------------|--|-----------------------|
| 0     | 1, 2, 3, 4, 5, 6 | —  | —                     |
| 1     | 1, 2, 3, 4, 5, 6 | ChangeFlowAndStep (3, 4)<br>ChangeFlowAndStep (5, 6) | —<br>—                |
| 2     | 1, 2             | —  | —                     |
| 3     | 1, 2             | —  | —                     |
| 4     | 1, 2             | ChangeFlowAndStep (1, 2)                             | —                     |
| 5     | 5, 6             | —  | —                     |
| 6     | 5, 6             | SetMeasureAll (5, 6)                                 | —                     |
| 7     | 5, 6             | —  | —                     |
| 8     | 5, 6             | ChangeFlowAndStep (5, 6)                             | —                     |
| 9     | —                | —  | —                     |
| 10    | 3, 4             | —  | ChangeFlowAndStep (4) |
| 11    | 3, 4             | TemporaryOffFlow (3, 4)                              | —                     |
| 12    | 3, 4             | —  | —                     |
| 13    | 3, 4             | Terminate (3)  | —                     |
| 14    | 4                | SetMeasureAll (4)                                    | —                     |
| 15    | 4                | —  | —                     |
| 16    | 1, 2, 4, 5, 6    | —  | —                     |
| 17    | 1, 2, 4, 5, 6    | —  | ChangeFlowAndStep (2) |
| 18    | 1, 2, 5, 6       | —  | —                     |
| 19    | 1, 2, 5, 6       | ChangeFlowAndStep (1, 2, 5, 6)                       | —                     |
| 20    | —                | —  | —                     |
| 21    | —                | —  | —                     |
| 22    | 3, 4             | —  | —                     |
| 23    | 3, 4             | —  | —                     |

Figure 6.11 illustrates how these errors are visualized in the graph drawing of the complex experiment. As expected, the color of four nodes turned red and four error messages were displayed in the errors tab: "Circular reference of sub-material 2 on Step 19", "Circular reference of sub-material 4 on Step 0", "Trying to use missing sub-material 4 on Step 11" and "Trying to use missing sub-material 4 on Step 14". As previously stated, clicking in any of the error messages automatically filters the graph drawing by the sub-material causing the error indicated in the message. The resulting graph drawings are shown in Figure 6.12 and in Figure 6.13. The same graph drawings could be obtained by applying the respective filters to the graph drawing shown in Figure 6.11.

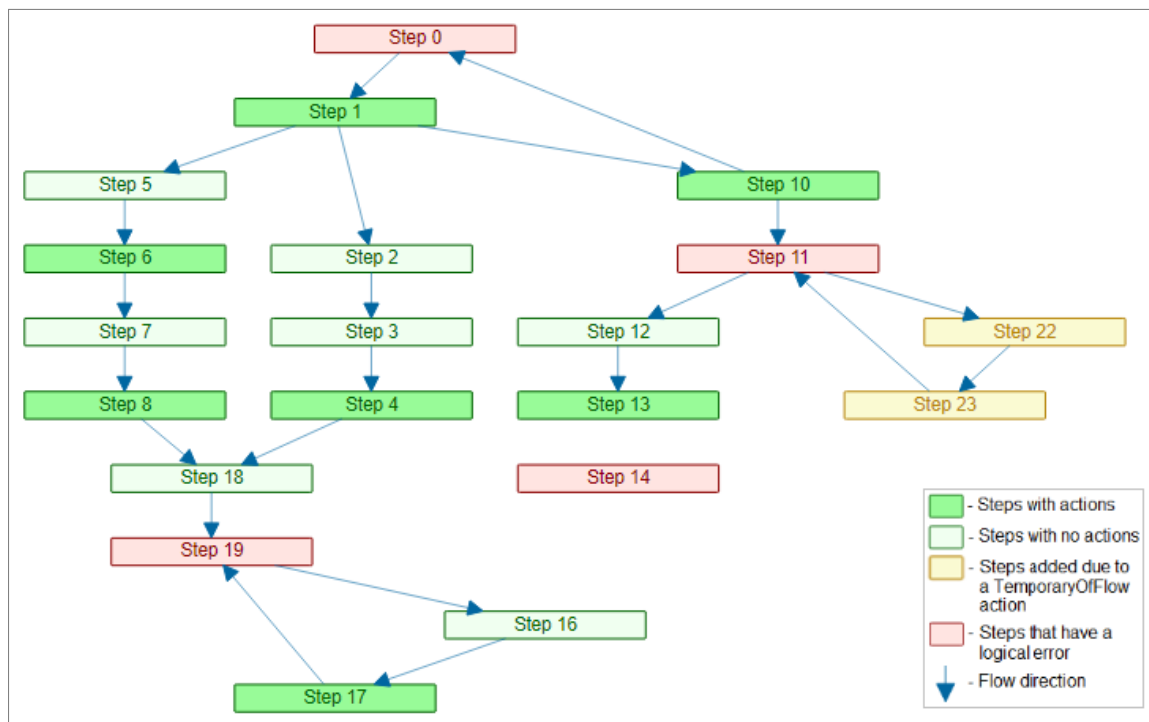


Figure 6.11: Graph drawing of the complex experiment with circular reference errors and respective missing sub-material errors

The observation of Figure 6.11, namely the red nodes, shows an isolated node (Step 14), which means that there is a missing sub-material (sub-material 4) in the respective action. Another red node (Step 11) due to a missing sub-material type of error is not presented as an isolated node in Figure 6.11 because Step 11 is also a part of the path of the sub-material 3. For visualizing this node as an isolated node, it is necessary to filter the graph drawing by the sub-material 4 (see Figure 6.12), thus hiding all the elements that are not traversed by sub-material 4 or that have an action for sub-material 4. Because Step 11 has the color red, and also is a part of what seems to be a cycle, one might think that this indicates a circular reference type of error. As mentioned before, this is not the case, because the other nodes involved have the color yellow, which indicates an action `TemporaryOffFlow`. The color red in Step 11 is due to the already mentioned missing sub-material type of error and not to a circular reference. To confirm this, the user can check the error messages, that show that the error in Step 11 is due to a missing sub-material. If this is not enough, clicking on that error message, filters the graph drawing by the missing sub-material (see Figure 6.12), which, for a missing sub-material error due to that same sub-material, will always show that node as an isolated node. The other two red nodes in Figure 6.11 (Step 0 and Step 19) are each a part of a cycle with upward and downward links, which indicates a circular reference type of error. That information can also be confirmed by clicking on the respective error messages and filtering the graph drawing by the respective sub-material (see Figure 6.12 and Figure 6.13 for sub-material 4 and 2, respectively).

The in-house experts found these results in accordance with what was intended.

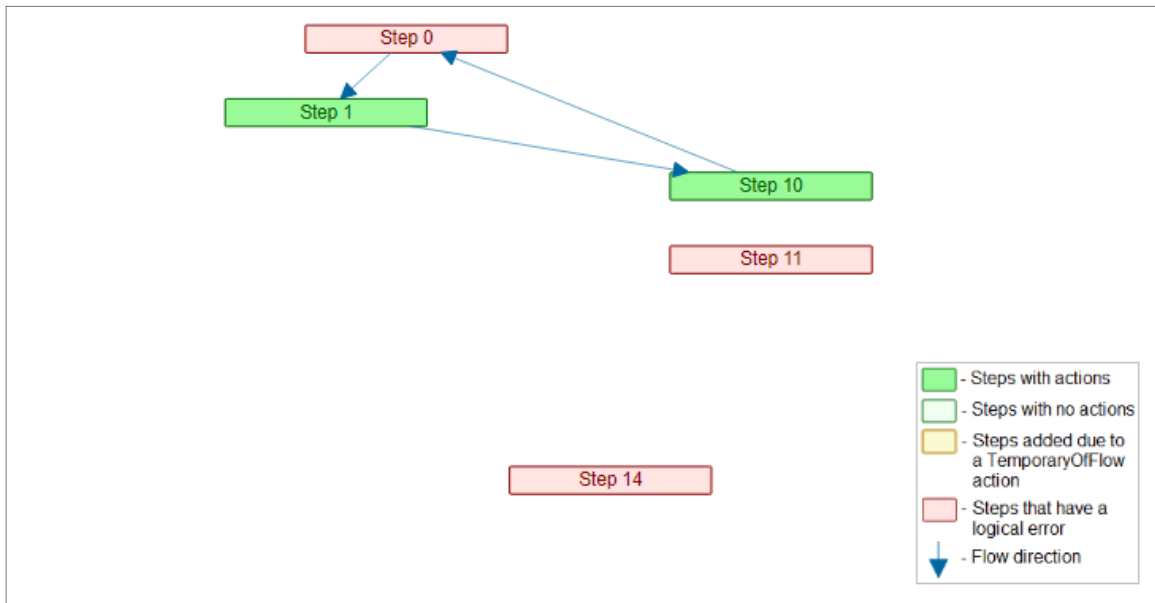


Figure 6.12: Graph drawing of the complex experiment with circular reference errors and respective missing sub-material error filtered by the sub-materials causing the errors (sub-material 4)

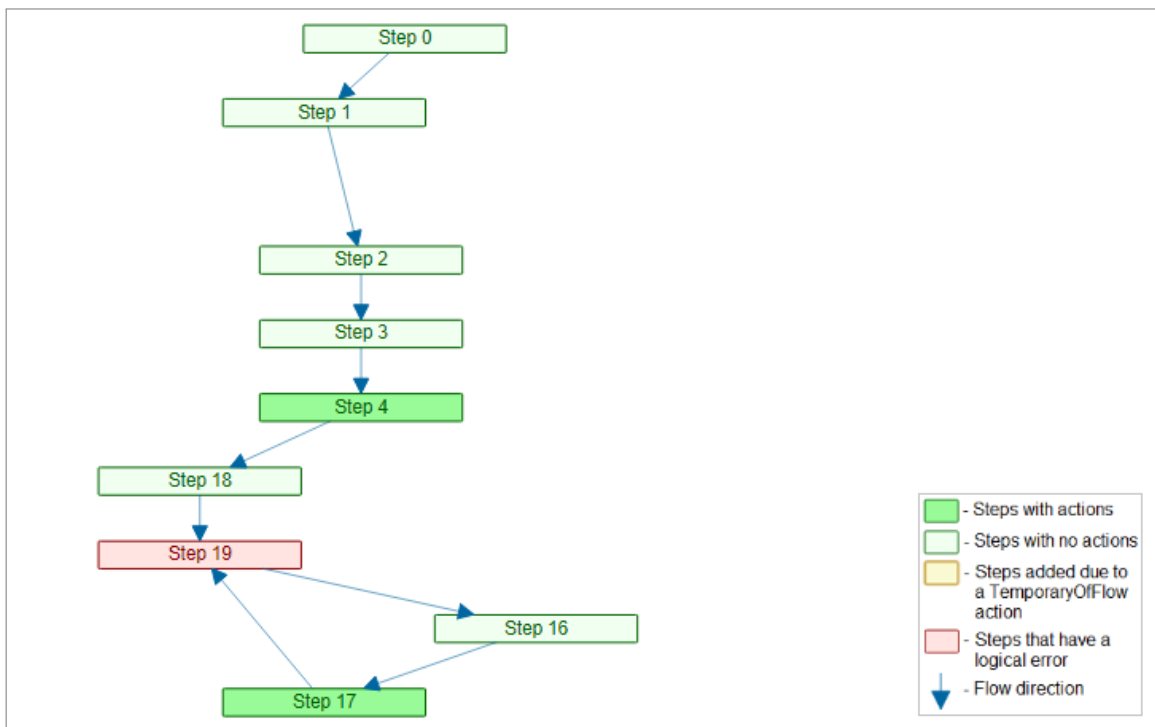


Figure 6.13: Graph drawing of the complex experiment with circular reference errors filtered by the sub-material causing the errors (sub-material 2)

## **6.4 Summary**

This chapter was dedicated to the validation and analyses of the results. Of the presented results, it is important to highlight that:

- The implemented solution was illustrated resorting to a complex experiment that was designed for this purpose. The graph drawing of this complex experiment was presented.
- The user features developed for the implemented solution were also illustrated resorting to the graph drawing of the complex experiment, focusing on how they facilitate the user interaction with the graph drawing.
- To show the capability of the implemented solution in the detection and visualization of errors, changes were made to the complex experiment to create different errors of the two possible types (missing sub-material and circular reference). The resulting graph drawings were included and the usefulness of the user features was shown in the analyses of these drawings in the process of error detection.

## Chapter 7

# Conclusions

The problem that motivated this dissertation is grounded on the idea that Industry 4.0 brings difficulties and opportunities, like the IIoT, and that, even though old MES are not adequate, the new MES solutions can optimize the production environment. The global relevance of CMF and recognized expertise as a MES solution provider was argued. The improvements recently made in CMF MES were highlighted, namely the EM, a breakthrough module in the achievement of optimum process performance. The module allows workers to perform DoE with multiple input variables, and execute and monitor experiments seamlessly in a single system, which results in an easier, more efficient, and effective process. The advantages of DoE were presented, and its relevance for the semiconductor manufacturing industry was pointed out. Reasons for the difficulty in implementing DoE, regarding high manufacturing costs (time and money), the little time available in the production line to conduct experiments, and the manufacturing process itself is very complex, are described. It was argued that although the CMF MES EM module is a valuable addition in performing DoE for the semiconductor manufacturing, because of its complexity, human errors occur and cause waste of valuable production time and resources. As these errors are due to logically invalid experiments and incorrect experiments, a claim was made that graph and graph drawings would be an appropriate way to address this problem, making CMF MES even a more valuable tool for companies towards Industry 4.0.

With this problem and framework in mind, the main goal of this dissertation was stated as the implementation of graph capabilities into the CMF MES EM module, to prevent logically invalid and or incorrectly designed experiments to be executed.

To achieve the general objective of this dissertation, an agile software process model, more specifically the SCRUM framework was used. A set of tasks were defined and organized in five stages: preparation, graph visualization, logical verification, improvement of user features, and testing and validation. The tasks were analysed to evaluate the priority of each one, and, in the begin of each sprint, moved to the sprint backlog. It was the choice of the SCRUM framework that justified resorting to a simple experiment with several versions in the graph visualization and

testing and validation stages. This simple experiment was progressively evolving, thus being well adjusted to the duration of the sprints. Also, this allowed for a development of the implemented solution that was based on a set of decisions and intermediate tests that were validating it.

## 7.1 Main contributions

The main contributions of this dissertation focused on the implementation of the logical verification and of the feedback for users, and in the implementation of features that facilitate users' identification of errors' cause and related information. For this implementation, a complex experiment was used, designed for this specific purpose, to demonstrate that the solution developed meets the intended objectives. The use of this complex experiment, which could be seen as a limitation, because it is not an experiment obtained in the real context of Wafer Fabrication DoE, allowed the combination, in a single experiment, of a wide range of errors, with various consequences, allowing to simplify the mentioned demonstration.

Taking into consideration the results obtained, one can conclude that the use of DAG and hierarchical graph drawings were an appropriate option for the visualization of Wafer Fabrication DoE since the graph drawings obtained follow the recommendations and aesthetic criteria that facilitate the understanding and interpretation of these drawings. All the intentionally added logical errors were correctly identified and signaled with the appropriate color and corresponding error messages, thus making it possible to conclude of the adequacy of the code developed to do it and of the graph structure implemented.

Also in the results, one tried to demonstrate the need and advantage of the implemented user features, emphasizing that, although the signaling of errors is important, the use of these features is indispensable to the visualization and identification of the cause of the errors, thus preventing not only the execution of logically invalid experiments but also of incorrect ones. It is, therefore, possible to conclude that taking these aspects into account was an appropriate choice and that the general objective has been fully achieved.

## 7.2 Future work

Although the general objective has been fully achieved, this dissertation works as a proof of concept, and for it to be ready for integration in the final product it needs some changes in the GUI. These changes include improving the node representation to be able to show more information to the user in the node itself. Some of this information could be in collapsible sections in the node so that the user can choose between a more heavy information view and a lighter version more similar to the one presented in this dissertation. There should also be a visual representation of the sub-material groups in the nodes, for example with a color scheme system, that would allow the user to check which group contains the sub-materials passing through each node.

Moving on from the graph drawing itself, some adjustments could be made to the rest of the page, mainly to the details tab. Even though this tab was implemented, its features are limited and

rudimentary, presenting to the user information regarding each node clicked, but just text-based. In the future, some more information could be added and there should be links wherever they are useful. For example, when clicking a node there should be information about the flow of the corresponding step, including a link to the flow page. All these improvements mentioned so far should be implemented while keeping and improving the look and feel of the CMF MES and the EM module.

In addition to the improvements to the GUI, some further developments that could be done to add more features to the graph. The main improvement would be allowing the user to define, or at least edit, the experiment in the graph view, without needing to use the wizards in the matrix view. Initially, this should start as the user being able to click in a button on a node to add/edit/remove an action. Then, there should be an option to allow to add or remove flows, which implies also the need to have a way of add/edit/remove links. Furthermore, this allows the user to tackle any mistakes that he or she might make, straight away.

These suggestions for future work highlight the potential yet to be explored of the use of graphs and graph drawings in streamlining wafer fabrication DoE, thus being of paramount importance for the semiconductor industry, and in making the integrated CMF MES an even more valuable tool for companies towards Industry 4.0.





# Appendix A

## Source code

In this appendix, it is presented most of the source code of the software developed in this dissertation. This only includes the typescript code since the html and less (Leaner Style Sheets) files are very simple and standard. Some sections of this file were also omitted to keep the appendix as concise and relevant as possible.

```
1
2 @Component ({
3     moduleId: __moduleName,
4     selector: 'icf-graph-custom-graphView',
5     inputs: [],
6     outputs: [],
7     templateUrl: './graphView.html',
8     styleUrls: ['./graphView.css',
9         '../../../../../jointjs/dist/joint.css'],
10    encapsulation: ng.ViewEncapsulation.None,
11    assign: { i18n: i18n }
12 })
13 export class GraphView extends CoreComponent implements ng.OnChanges, ng.OnInit, ng
    .AfterViewInit {
14
15    constructor(private _elementRef: ng.ElementRef, viewContainerRef: ng.
        ViewContainerRef) {
16        super(_elementRef, viewContainerRef);
17
18        this.errorCount = 0;
19
20        this.headerItem1 = {
21            id: "0",
22            iconClass: "",
23            text: "Errors",
24            disabled: false
25        };
26        this.bodyItems1 = [
```

```
27     ];
28     this.bodyItems1SubMaterials = [
29     ];
30     this.menuModel1 = {
31         headerItem: this.headerItem1,
32         bodyItems: this.bodyItems1
33     }
34
35     this.disabled = false;
36     this.multiSelection = true;
37     this.search = true;
38
39     this.items = [];
40
41     this.selectedItems = [];
42
43     this.leftPanelShowCollapse = true;
44     this.rightPanelShowCollapse = true;
45     this.centerContent = "Click on the left menu";
46
47     this.selectedStepName = "";
48     this.selectedStepDescription = "";
49     this.selectedStepAllMaterials = [];
50     this.selectedStepAllMaterialsWithActions = [];
51     this.selectedStepAllMaterialsWithErrors = [];
52     this.selectedStepActions = [];
53 }
54
55 ///region Private methods
56
57 ///endregion
58
59 ///region Public methods
60
61 public onSelect(value: any): void {
62     this.selectedItems = [];
63
64     if (this.multiSelection) {
65         value.map((item, index) => {
66             this.selectedItems[index] = item.item.name;
67         });
68     } else {
69         this.selectedItems.push(value.item.name);
70     }
71
72     this.filterBySubMaterial(this.selectedItems);
73 }
74
75 public onClick(item: any): void {
```

```

76     this.centerContent = "clicked: " + item.name;
77   }
78
79   public onMenuGroupClick(e: any) {
80   }
81
82   public onPanelBarItemClick(e: any) {
83     this.selectedItems = [];
84     this.selectedItems.push(this.bodyItems1SubMaterials[e.id]);
85     this.filterBySubMaterial(this.selectedItems);
86   }
87
88   // New functions
89   // Fill with subMaterials
90   public fillSubMaterials(node, subMaterials): boolean {
91
92     // TemporaryOffFlow Stuff
93     for (let i = 0; i < node.attributes.tempLinks.length; i++) {
94       const tempSubMaterials = [];
95       for (let j = 0; j < node.attributes.tempLinks[i][1].length; j++) {
96         if (subMaterials.indexOf(node.attributes.tempLinks[i][1][j]) !==
97           -1) {
98           tempSubMaterials.push(node.attributes.tempLinks[i][1][j]);
99           node.attributes.tempLinks[i][0].attributes.subMaterials.push(
100             node.attributes.tempLinks[i][1][j]);
101         }
102       }
103       this.fillSubMaterialsTemp(node.attributes.tempLinks[i][0].attributes.
104         target, tempSubMaterials);
105     }
106
107     // SubMaterials sent to next step in the flow
108     const childSubMaterials = [];
109
110     // SubMaterials splitted from the flow
111     const allSplitSubMaterials = [];
112
113     // Save subMaterials in node
114     let circRef = false;
115     for (let i = 0; i < subMaterials.length; i++) {
116       if (node.attributes.subMaterials.find(subMaterial => subMaterial ===
117         subMaterials[i]) === undefined) {
118         node.attributes.subMaterials.push(subMaterials[i]);
119       } else {
120         node.attributes.subMaterialsError.push(subMaterials[i]);
121         node.attributes.subMaterialsError.sort();
122         node.attr({
123           rect: { fill: this.inColorError, stroke: this.outColorError },
124           text: { fill: this.outColorError }
125         });
126       }
127     }
128   }

```

```

121         });
122         const message = 'Circular reference of Sub-Material ' +
            subMaterials[i]
123             + ' on step ' + node.attributes.stepName;
124         this.bodyItems1.push({
125             id: this.errorCount.toString(),
126             iconClass: "",
127             text: message,
128             disabled: false
129         });
130         this.bodyItems1SubMaterials.push(subMaterials[i]);
131         this.errorCount++;
132         console.log('Circular reference of Sub-Material ' + subMaterials[i]
            + ' on step ' + node.attributes.stepName);
133         circRef = true;
134     }
135 }
136
137 if (circRef) {
138     return false;
139 }
140
141 node.attributes.subMaterials.sort();
142
143 // Terminated subMaterials
144 for (let i = 0; i < node.attributes.actions.size; i++) {
145     if (node.attributes.actions.get(i).find(action => action === 0) === 0)
146     {
147         for (let j = 0; j < node.attributes.subMaterialsExp.get(i).length;
            j++) {
148             allSplitSubMaterials.push(node.attributes.subMaterialsExp.get(i)
                [j]);
149         }
150     }
151
152 // Splitted subMaterials
153 for (let i = 0; i < node.attributes.otherLinks.length; i++) {
154     const splitSubMaterials = [];
155     for (let j = 0; j < node.attributes.otherLinks[i][1].length; j++) {
156         if (subMaterials.indexOf(node.attributes.otherLinks[i][1][j]) !==
            -1) {
157             splitSubMaterials.push(node.attributes.otherLinks[i][1][j]);
158             allSplitSubMaterials.push(node.attributes.otherLinks[i][1][j]);
159             node.attributes.otherLinks[i][0].attributes.subMaterials.push(
                node.attributes.otherLinks[i][1][j]);
160         }
161     }
162

```

```
163         // Send splitted subMaterials to another flow
164         if (splitSubMaterials.length !== 0) {
165             this.fillSubMaterials(node.attributes.otherLinks[i][0].attributes.
166                 target, splitSubMaterials);
167         }
168     }
169     // Remaining subMaterials
170     for (let i = 0; i < subMaterials.length; i++) {
171         if (allSplitSubMaterials.indexOf(subMaterials[i]) === -1) {
172             childSubMaterials.push(subMaterials[i]);
173         }
174     }
175
176     // Send subMaterials to next step in the flow
177     if (node.attributes.mainLink !== undefined) {
178         if (childSubMaterials.length !== 0) {
179             node.attributes.mainLink.attributes.subMaterials = node.attributes.
180                 mainLink.attributes.subMaterials.concat(childSubMaterials);
181             node.attributes.mainLink.attributes.subMaterials.sort();
182             this.fillSubMaterials(node.attributes.mainLink.attributes.target,
183                 childSubMaterials);
184         }
185     }
186     return true;
187 }
188 // Fill with subMaterials ofr TempOffFlow
189 public fillSubMaterialsTemp(node, subMaterials): boolean {
190
191     node.attributes.subMaterials = subMaterials;
192
193     if (node.attributes.mainLink !== undefined) {
194         node.attributes.mainLink.attributes.subMaterials = subMaterials;
195         node.attributes.mainLink.attributes.subMaterials.sort();
196         this.fillSubMaterialsTemp(node.attributes.mainLink.attributes.target,
197             subMaterials)
198     } else {
199         node.attributes.tempLinks[0][0].attributes.subMaterials = subMaterials;
200         node.attributes.tempLinks[0][0].attributes.subMaterials.sort();
201     }
202     return true;
203 }
204
205 // Filter by subMaterial(s)
206 public filterBySubMaterial(filteredSubMaterials) {
207     const nodes = this.paper.model.getElements();
```

```
208     const links = this.paper.model.getLinks();
209
210     if (filteredSubMaterials.length === 0) {
211         for (let i = 0; i < links.length; i++) {
212             if (links[i].attributes.subMaterials.length === 0) {
213                 links[i].attr({
214                     '.connection': { display: 'none' },
215                     '.connection-wrap': { display: 'none' },
216                     '.marker-target': { display: 'none' }
217                 });
218                 if (links[i].attributes.source.attributes.subMaterialsExpSorted
219                     .length === 0 &&
220                     links[i].attributes.source.attributes.subMaterials.length
221                         === 0) {
222                     links[i].attributes.source.attr({
223                         rect: { display: 'none' },
224                         text: { display: 'none' }
225                     });
226                 } else {
227                     links[i].attributes.source.attr({
228                         rect: { display: 'initial' },
229                         text: { display: 'initial' }
230                     });
231                 } if (links[i].attributes.source.attributes.type !== 'Temp')
232                 {
233                     if (links[i].attributes.source.attributes.
234                         subMaterialsError.length > 0) {
235                         links[i].attributes.source.attr({
236                             rect: { fill: this.inColorError, stroke: this.
237                                 outColorError },
238                             text: { fill: this.outColorError }
239                         });
240                     } else if (links[i].attributes.source.attributes.
241                         subMaterialsExpSorted.length > 0) {
242                         links[i].attributes.source.attr({
243                             rect: { fill: this.actionColor, stroke: this.
244                                 outColor },
245                             text: { fill: this.outColor }
246                         });
247                     } else {
248                         links[i].attributes.source.attr({
249                             rect: { fill: this.inColor, stroke: this.
250                                 outColor },
251                             text: { fill: this.outColor }
252                         });
253                     }
254                 }
255             }
256         }
257     }
```

```
248         if (links[i].attributes.target.attributes.subMaterialsExpSorted
249             .length === 0
250             && links[i].attributes.target.attributes.subMaterials.
251                 length === 0) {
252             links[i].attributes.target.attr({
253                 rect: { display: 'none' },
254                 text: { display: 'none' }
255             });
256         } else {
257             links[i].attributes.target.attr({
258                 rect: { display: 'initial' },
259                 text: { display: 'initial' }
260             });
261         } if (links[i].attributes.target.attributes.type !== 'Temp')
262         {
263             if (links[i].attributes.target.attributes.
264                 subMaterialsError.length > 0) {
265                 links[i].attributes.target.attr({
266                     rect: { fill: this.inColorError, stroke: this.
267                         outColorError },
268                     text: { fill: this.outColorError }
269                 });
270             } else if (links[i].attributes.target.attributes.
271                 subMaterialsExpSorted.length > 0) {
272                 links[i].attributes.target.attr({
273                     rect: { fill: this.actionColor, stroke: this.
274                         outColor },
275                     text: { fill: this.outColor }
276                 });
277             }
278         }
279     } else {
280         links[i].attributes.source.attr({
281             rect: { display: 'initial' },
282             text: { display: 'initial' }
283         });
284     } if (links[i].attributes.source.attributes.type !== 'Temp') {
285         if (links[i].attributes.source.attributes.subMaterialsError
286             .length > 0) {
287             links[i].attributes.source.attr({
288                 rect: { fill: this.inColorError, stroke: this.
289                     outColorError },
```

```

287         text: { fill: this.outColorError }
288     });
289     } else if (links[i].attributes.source.attributes.
290         subMaterialsExpSorted.length > 0) {
291         links[i].attributes.source.attr({
292             rect: { fill: this.actionColor, stroke: this.
293                 outColor },
294             text: { fill: this.outColor }
295         });
296     } else {
297         links[i].attributes.source.attr({
298             rect: { fill: this.inColor, stroke: this.outColor
299                 },
300             text: { fill: this.outColor }
301         });
302     }
303     links[i].attributes.target.attr({
304         rect: { display: 'initial' },
305         text: { display: 'initial' }
306     });
307     if (links[i].attributes.target.attributes.type !== 'Temp') {
308         if (links[i].attributes.target.attributes.subMaterialsError
309             .length > 0) {
310             links[i].attributes.target.attr({
311                 rect: { fill: this.inColorError, stroke: this.
312                     outColorError },
313                 text: { fill: this.outColorError }
314             });
315         } else if (links[i].attributes.target.attributes.
316             subMaterialsExpSorted.length > 0) {
317             links[i].attributes.target.attr({
318                 rect: { fill: this.actionColor, stroke: this.
319                     outColor },
320                 text: { fill: this.outColor }
321             });
322         } else {
323             links[i].attributes.target.attr({
324                 rect: { fill: this.inColor, stroke: this.outColor
325                     },
326                 text: { fill: this.outColor }
327             });
328         }
329     }
330     links[i].attr({
331         '.connection': { display: 'initial' },
332         '.connection-wrap': { display: 'initial' },
333         '.marker-target': { display: 'initial' }
334     });

```



```

328         }
329     }
330     } else {
331         for (let i = 0; i < nodes.length; i++) {
332             if (nodes[i].attributes.subMaterials.filter(subMaterial =>
333                 filteredSubMaterials.includes(subMaterial)).length === 0 &&
334                 nodes[i].attributes.subMaterialsExpSorted.filter(subMaterial =>
335                     filteredSubMaterials.includes(subMaterial)).length === 0)
336             {
337                 nodes[i].attr({
338                     rect: { display: 'none' },
339                     text: { display: 'none' }
340                 });
341             } else {
342                 nodes[i].attr({
343                     rect: { display: 'initial' },
344                     text: { display: 'initial' }
345                 });
346                 if (nodes[i].attributes.type !== 'Temp') {
347                     if (!(nodes[i].attributes.subMaterialsError.filter(
348                         subMaterial =>
349                             filteredSubMaterials.includes(subMaterial)).length ===
350                             0)) {
351                         nodes[i].attr({
352                             rect: { fill: this.inColorError, stroke: this.
353                                 outColorError },
354                             text: { fill: this.outColorError }
355                         });
356                     } else if (!(nodes[i].attributes.subMaterialsExpSorted.
357                         filter(subMaterial =>
358                             filteredSubMaterials.includes(subMaterial)).length ===
359                             0)) {
360                         nodes[i].attr({
361                             rect: { fill: this.actionColor, stroke: this.
362                                 outColor },
363                             text: { fill: this.outColor }
364                         });
365                     } else {
366                         nodes[i].attr({
367                             rect: { fill: this.inColor, stroke: this.outColor
368                                 },
369                             text: { fill: this.outColor }
370                         });
371                     }
372                 }
373             }
374         }
375     }
376     }
377     }
378     }
379     }
380     }
381     }
382     }
383     }
384     }
385     }
386     }
387     }
388     }
389     }
390     }
391     }
392     }
393     }
394     }
395     }
396     }
397     }
398     }
399     }
400     }
401     }
402     }
403     }
404     }
405     }
406     }
407     }
408     }
409     }
410     }
411     }
412     }
413     }
414     }
415     }
416     }
417     }
418     }
419     }
420     }
421     }
422     }
423     }
424     }
425     }
426     }
427     }
428     }
429     }
430     }
431     }
432     }
433     }
434     }
435     }
436     }
437     }
438     }
439     }
440     }
441     }
442     }
443     }
444     }
445     }
446     }
447     }
448     }
449     }
450     }
451     }
452     }
453     }
454     }
455     }
456     }
457     }
458     }
459     }
460     }
461     }
462     }
463     }
464     }
465     }
466     }
467     }
468     }
469     }
470     }
471     }
472     }
473     }
474     }
475     }
476     }
477     }
478     }
479     }
480     }
481     }
482     }
483     }
484     }
485     }
486     }
487     }
488     }
489     }
490     }
491     }
492     }
493     }
494     }
495     }
496     }
497     }
498     }
499     }
500     }
501     }
502     }
503     }
504     }
505     }
506     }
507     }
508     }
509     }
510     }
511     }
512     }
513     }
514     }
515     }
516     }
517     }
518     }
519     }
520     }
521     }
522     }
523     }
524     }
525     }
526     }
527     }
528     }
529     }
530     }
531     }
532     }
533     }
534     }
535     }
536     }
537     }
538     }
539     }
540     }
541     }
542     }
543     }
544     }
545     }
546     }
547     }
548     }
549     }
550     }
551     }
552     }
553     }
554     }
555     }
556     }
557     }
558     }
559     }
560     }
561     }
562     }
563     }
564     }
565     }
566     }
567     }
568     }
569     }
570     }
571     }
572     }
573     }
574     }
575     }
576     }
577     }
578     }
579     }
580     }
581     }
582     }
583     }
584     }
585     }
586     }
587     }
588     }
589     }
590     }
591     }
592     }
593     }
594     }
595     }
596     }
597     }
598     }
599     }
600     }
601     }
602     }
603     }
604     }
605     }
606     }
607     }
608     }
609     }
610     }
611     }
612     }
613     }
614     }
615     }
616     }
617     }
618     }
619     }
620     }
621     }
622     }
623     }
624     }
625     }
626     }
627     }
628     }
629     }
630     }
631     }
632     }
633     }
634     }
635     }
636     }
637     }
638     }
639     }
640     }
641     }
642     }
643     }
644     }
645     }
646     }
647     }
648     }
649     }
650     }
651     }
652     }
653     }
654     }
655     }
656     }
657     }
658     }
659     }
660     }
661     }
662     }
663     }
664     }
665     }
666     }
667     }
668     }
669     }
670     }
671     }
672     }
673     }
674     }
675     }
676     }
677     }
678     }
679     }
680     }
681     }
682     }
683     }
684     }
685     }
686     }
687     }
688     }
689     }
690     }
691     }
692     }
693     }
694     }
695     }
696     }
697     }
698     }
699     }
700     }
701     }
702     }
703     }
704     }
705     }
706     }
707     }
708     }
709     }
710     }
711     }
712     }
713     }
714     }
715     }
716     }
717     }
718     }
719     }
720     }
721     }
722     }
723     }
724     }
725     }
726     }
727     }
728     }
729     }
730     }
731     }
732     }
733     }
734     }
735     }
736     }
737     }
738     }
739     }
740     }
741     }
742     }
743     }
744     }
745     }
746     }
747     }
748     }
749     }
750     }
751     }
752     }
753     }
754     }
755     }
756     }
757     }
758     }
759     }
760     }
761     }
762     }
763     }
764     }
765     }
766     }
767     }
768     }
769     }
770     }
771     }
772     }
773     }
774     }
775     }
776     }
777     }
778     }
779     }
780     }
781     }
782     }
783     }
784     }
785     }
786     }
787     }
788     }
789     }
790     }
791     }
792     }
793     }
794     }
795     }
796     }
797     }
798     }
799     }
800     }
801     }
802     }
803     }
804     }
805     }
806     }
807     }
808     }
809     }
810     }
811     }
812     }
813     }
814     }
815     }
816     }
817     }
818     }
819     }
820     }
821     }
822     }
823     }
824     }
825     }
826     }
827     }
828     }
829     }
830     }
831     }
832     }
833     }
834     }
835     }
836     }
837     }
838     }
839     }
840     }
841     }
842     }
843     }
844     }
845     }
846     }
847     }
848     }
849     }
850     }
851     }
852     }
853     }
854     }
855     }
856     }
857     }
858     }
859     }
860     }
861     }
862     }
863     }
864     }
865     }
866     }
867     }
868     }
869     }
870     }
871     }
872     }
873     }
874     }
875     }
876     }
877     }
878     }
879     }
880     }
881     }
882     }
883     }
884     }
885     }
886     }
887     }
888     }
889     }
890     }
891     }
892     }
893     }
894     }
895     }
896     }
897     }
898     }
899     }
900     }
901     }
902     }
903     }
904     }
905     }
906     }
907     }
908     }
909     }
910     }
911     }
912     }
913     }
914     }
915     }
916     }
917     }
918     }
919     }
920     }
921     }
922     }
923     }
924     }
925     }
926     }
927     }
928     }
929     }
930     }
931     }
932     }
933     }
934     }
935     }
936     }
937     }
938     }
939     }
940     }
941     }
942     }
943     }
944     }
945     }
946     }
947     }
948     }
949     }
950     }
951     }
952     }
953     }
954     }
955     }
956     }
957     }
958     }
959     }
960     }
961     }
962     }
963     }
964     }
965     }
966     }
967     }
968     }
969     }
970     }
971     }
972     }
973     }
974     }
975     }
976     }
977     }
978     }
979     }
980     }
981     }
982     }
983     }
984     }
985     }
986     }
987     }
988     }
989     }
990     }
991     }
992     }
993     }
994     }
995     }
996     }
997     }
998     }
999     }
1000    }

```

```

367     if (links[i].attributes.subMaterials.filter(subMaterial =>
368         filteredSubMaterials.includes(subMaterial)).length === 0) {
369         links[i].attr({
370             '.connection': { display: 'none' },
371             '.connection-wrap': { display: 'none' },
372             '.marker-target': { display: 'none' }
373         });
374     } else {
375         links[i].attr({
376             '.connection': { display: 'initial' },
377             '.connection-wrap': { display: 'initial' },
378             '.marker-target': { display: 'initial' }
379         });
380     }
381 }
382 }
383
384 // Hide nodes that don't have actions or subMaterials
385 public hideUnusedNodes(graph) {
386     const links = graph.getLinks();
387     for (let i = 0; i < links.length; i++) {
388         if (links[i].attributes.subMaterials.length === 0) {
389             links[i].attr({
390                 '.connection': { display: 'none' },
391                 '.connection-wrap': { display: 'none' },
392                 '.marker-target': { display: 'none' }
393             });
394             if (links[i].attributes.source.attributes.subMaterialsExpSorted.
395                 length === 0 &&
396                 links[i].attributes.source.attributes.subMaterials.length ===
397                 0) {
398                 links[i].attributes.source.attr({
399                     rect: { display: 'none' },
400                     text: { display: 'none' }
401                 });
402             }
403             if (links[i].attributes.target.attributes.subMaterialsExpSorted.
404                 length === 0 &&
405                 links[i].attributes.target.attributes.subMaterials.length ===
406                 0) {
407                 links[i].attributes.target.attr({
408                     rect: { display: 'none' },
409                     text: { display: 'none' }
410                 });
411             }
412         }
413     }
414 }

```

```
411
412 // Details Tab
413 public setSelectedStep(attributes): boolean {
414     this.selectedStepName = attributes.stepName;
415     this.selectedStepDescription = attributes.stepDesc;
416     this.selectedStepAllMaterials = attributes.subMaterials;
417     this.selectedStepAllMaterialsWithActions = attributes.subMaterialsExpSorted
418     ;
419     this.selectedStepAllMaterialsWithErrors = attributes.subMaterialsError;
420
421     this.selectedStepActions = [];
422     let message;
423
424     for (let i = 0; i < attributes.actions.size; i++) {
425         message = 'Sub-Materials: ' + attributes.subMaterialsExp.get(i) + ' in
426         Actions: ';
427         for (let j = 0; j < attributes.actions.get(i).length; j++) {
428             message += Cmf.Navigo.BusinessObjects.
429                 ExperimentStepMaterialGroupAction[attributes.actions.get(i)[j]
430                 ];
431             message += '(' + Cmf.Navigo.BusinessObjects.
432                 ExperimentStepMaterialGroupActionEvent[attributes.events.get(i)
433                 [j]] + ')' + '\n';
434         }
435         this.selectedStepActions.push(message);
436     }
437
438     return true;
439 }
440
441 /**
442  * On changes method
443  *
444  * @param changes the changes made to the component properties
445  */
446 public ngOnChanges(changes: ng.SimpleChanges): void { }
447
448 public async ngAfterViewInit() {
449
450     const _this = this;
451
452     let outputExpDef;
453     let outputFlow;
454
455     // Get Experiment Definition (omitted)
456     // Returns the experiment in outputExpDef
457
458     // Get Main Flow (omitted)
459     // Returns the flow in outputFlow
```

```

454
455 // Save steps that have an experiment
456 const stepsExp = outputExpDef.ExperimentDefinition.Steps;
457 const numberOfSubMaterials = outputExpDef.ExperimentDefinition.
    RequiredSubMaterialsCount;
458 const allSubMaterials = [];
459 for (let i = 1; i <= numberOfSubMaterials; i++) {
460     allSubMaterials.push(i.toString());
461     this.items.push({ name: i.toString() });
462 }
463
464 // Save steps of the main flow
465 const mainFlowId = outputFlow.Flow.Id;
466 let steps = [];
467 if (outputFlow.Flow.FlowSteps === undefined) {
468     for (let i = 0; i < outputFlow.Flow.SubFlows.length; i++) {
469         steps = steps.concat(outputFlow.Flow.SubFlows[i].TargetEntity.
            FlowSteps);
470     }
471 } else {
472     steps = outputFlow.Flow.FlowSteps;
473 }
474
475 // Create Map of Flows (ID, [Step])
476 const flows = new Map();
477 flows.set(mainFlowId, []);
478 const flowsTemp = new Map();
479
480 // Create graph and paper
481 const graph = new joint.dia.Graph;
482
483 const paper = new joint.dia.Paper({
484     el: this.holder.nativeElement,
485     model: graph,
486     width: 1360,
487     height: 500,
488     interactive: function (cellView) {
489         if (cellView.model instanceof joint.dia.Link) {
490             return { vertexAdd: false };
491         }
492         return true;
493     },
494 });
495
496 this.paper = paper;
497
498 // Represent main flow
499 for (let i = 0; i < steps.length; i++) {
500     const mainStep = new joint.shapes.basic.Rect({

```

```

501         position: { x: 60, y: 20 + 100 * i },
502         size: { width: 220, height: 30 },
503         stepId: steps[i].TargetEntity.Id,
504         stepName: steps[i].TargetEntity.Name,
505         stepDesc: steps[i].TargetEntity.Description,
506         mainLink: undefined,
507         otherLinks: [],
508         tempLinks: [],
509         subMaterials: [],
510         subMaterialsExp: new Map(),
511         subMaterialsExpSorted: [],
512         subMaterialsError: [],
513         actions: new Map(),
514         events: new Map(),
515         type: 'Main'
516     }).attr({
517         rect: { fill: this.inColor, stroke: this.outColor, rx: 1, ry: 1 },
518         text: { 'font-size': 20, fill: this.outColor, text: steps[i].
                    TargetEntity.Name }
519     });
520
521     graph.addCell(mainStep);
522
523     flows.get(mainFlowId).push(mainStep);
524 }
525
526 // Add links for main flow
527 const mainSteps = flows.get(mainFlowId);
528 for (let i = 1; i < mainSteps.length; i++) {
529     const mainLink = new joint.dia.Link({
530         source: mainSteps[i - 1],
531         target: mainSteps[i],
532         subMaterials: [],
533         type: 'Main Flow'
534     })
535     graph.addCell(mainLink);
536     mainSteps[i - 1].attributes.mainLink = mainLink;
537 }
538
539 // Add experiments to nodes
540 for (let i = 0; i < stepsExp.length; i++) {
541     // If the flow isn't represented yet, represent it
542     if (!flows.has(stepsExp[i].Flow.Id)) {
543
544         // Get Other Flow (omitted)
545         // Returns the flow in outputOtherFlow
546
547         flows.set(stepsExp[i].Flow.Id, []);
548

```

```

549     let newSteps = [];
550     if (outputOtherFlow.Flow.FlowSteps === undefined) {
551         for (let i = 0; i < outputOtherFlow.Flow.SubFlows.length; i++)
552             {
553                 newSteps = newSteps.concat(outputOtherFlow.Flow.SubFlows[i
554                 ].TargetEntity.FlowSteps);
555             }
556     } else {
557         newSteps = outputOtherFlow.Flow.FlowSteps;
558     }
559
560     for (let j = 0; j < newSteps.length; j++) {
561         const otherStep = new joint.shapes.basic.Rect({
562             position: { x: 400, y: 20 + 100 * j },
563             size: { width: 220, height: 30 },
564             stepId: newSteps[j].TargetEntity.Id,
565             stepName: newSteps[j].TargetEntity.Name,
566             stepDesc: newSteps[j].TargetEntity.Description,
567             mainLink: undefined,
568             otherLinks: [],
569             tempLinks: [],
570             subMaterials: [],
571             subMaterialsExp: new Map(),
572             subMaterialsExpSorted: [],
573             subMaterialsError: [],
574             actions: new Map(),
575             events: new Map(),
576             type: 'Other'
577         }).attr({
578             rect: { fill: this.inColor, stroke: this.outColor, rx: 1,
579             ry: 1 },
580             text: { 'font-size': 20, fill: this.outColor, text:
581             newSteps[j].TargetEntity.Name }
582         });
583
584         graph.addCell(otherStep);
585
586         flows.get(stepsExp[i].Flow.Id).push(otherStep);
587     }
588
589     const otherSteps = flows.get(stepsExp[i].Flow.Id);
590     for (let j = 1; j < otherSteps.length; j++) {
591         const mainLink = new joint.dia.Link({
592             source: otherSteps[j - 1],
593             target: otherSteps[j],
594             subMaterials: [],
595             type: 'Other Flow'
596         })
597     }
598     graph.addCell(mainLink);

```

```

594         otherSteps[j - 1].attributes.mainLink = mainLink;
595     }
596 }
597 const subMaterialGroups = stepsExp[i].MaterialGroups;
598 const subMaterialsExp = new Map();
599 const actionsExp = new Map();
600 const eventsExp = new Map();
601 let hasChangeFlowAndStep = false;
602 const ChangeFlowAndStepStep = [];
603 const ChangeFlowAndStepRestrictions = [];
604 let hasTemporaryOffFlow = false;
605 const temporaryOffFlowStep = [];
606 const temporaryOffFlowRestrictions = [];
607 for (let j = 0; j < subMaterialGroups.length; j++) {
608     if (!subMaterialGroups[j].NoActions) {
609
610         subMaterialsExp.set(j, subMaterialGroups[j].TargetEntity.
611             SubMaterialNumbers.split(';'));
612         actionsExp.set(j, []);
613         eventsExp.set(j, []);
614
615         for (let k = 0; k < subMaterialGroups[j].Actions.length; k++) {
616             actionsExp.get(j).push(subMaterialGroups[j].Actions[k].
617                 Action);
618             eventsExp.get(j).push(subMaterialGroups[j].Actions[k].Event
619                 );
620
621             // ChangeFlowAndStep
622             if (subMaterialGroups[j].Actions[k].Action === 2) {
623                 if (!flows.has(subMaterialGroups[j].Actions[k].
624                     ChangeFlowAndStepFlow.Id)) {
625
626                     // Get Other Flow (omitted)
627                     // Returns the flow in outputOtherFlow
628                     flows.set(subMaterialGroups[j].Actions[k].
629                         ChangeFlowAndStepFlow.Id, []);
630
631                     let newSteps = [];
632                     if (outputOtherFlow.Flow.FlowSteps === undefined) {
633                         for (let i = 0; i < outputOtherFlow.Flow.
634                             SubFlows.length; i++) {
635                             newSteps = newSteps.concat(outputOtherFlow.
636                                 Flow.SubFlows[i].TargetEntity.FlowSteps
637                                 );
638                         }
639                     } else {
640                         newSteps = outputOtherFlow.Flow.FlowSteps;
641                     }
642                     for (let l = 0; l < newSteps.length; l++) {

```

```

635         const otherStep = new joint.shapes.basic.Rect({
636             position: { x: 400, y: 20 + 100 * l },
637             size: { width: 220, height: 30 },
638             stepId: newSteps[l].TargetEntity.Id,
639             stepName: newSteps[l].TargetEntity.Name,
640             stepDesc: newSteps[l].TargetEntity.
                Description,
641             mainLink: undefined,
642             otherLinks: [],
643             tempLinks: [],
644             subMaterials: [],
645             subMaterialsExp: new Map(),
646             subMaterialsExpSorted: [],
647             subMaterialsError: [],
648             actions: new Map(),
649             events: new Map(),
650             type: 'Other'
651         }).attr({
652             rect: { fill: this.inColor, stroke: this.
                outColor, rx: 1, ry: 1 },
653             text: { 'font-size': 20, fill: this.
                outColor, text: newSteps[l].
                TargetEntity.Name }
654         });
655
656         graph.addCell(otherStep);
657
658         flows.get(subMaterialGroups[j].Actions[k].
                ChangeFlowAndStepFlow.Id).push(otherStep);
659     }
660
661     const otherSteps = flows.get(subMaterialGroups[j].
        Actions[k].ChangeFlowAndStepFlow.Id);
662     for (let l = 1; l < otherSteps.length; l++) {
663         const mainLink = new joint.dia.Link({
664             source: otherSteps[l - 1],
665             target: otherSteps[l],
666             subMaterials: [],
667             type: 'Other Flow'
668         })
669         graph.addCell(mainLink);
670         otherSteps[l - 1].attributes.mainLink =
                mainLink;
671     }
672 }
673
674 hasChangeFlowAndStep = true;
675

```



```

676     const changeFlow = flows.get(subMaterialGroups[j].
        Actions[k].ChangeFlowAndStepFlow.Id);
677     for (let l = 0; l < changeFlow.length; l++) {
678         if (subMaterialGroups[j].Actions[k].
            ChangeFlowAndStepStep.Id === changeFlow[l].
            attributes.stepId) {
679             ChangeFlowAndStepStep.push(changeFlow[l]);
680             ChangeFlowAndStepRestrictions.push(
                subMaterialsExp.get(j));
681             break;
682         }
683     }
684 }
685
686 // TempOffFlow
687 if (subMaterialGroups[j].Actions[k].Action === 3) {
688     const otherFlow: Cmf.Navigo.BusinessObjects.Flow = new
        Cmf.Navigo.BusinessObjects.Flow();
689     otherFlow.Id = subMaterialGroups[j].Actions[k].
        TemporaryOffFlowFlow.Id;
690
691     // Get Other Flow (omitted)
692     // Returns the flow in outputOtherFlow
693     flowsTemp.set(subMaterialGroups[j].Actions[k].
        TemporaryOffFlowFlow.Id, []);
694
695     let newSteps = [];
696     if (outputOtherFlow.Flow.FlowSteps === undefined) {
697         for (let i = 0; i < outputOtherFlow.Flow.SubFlows.
            length; i++) {
698             newSteps = newSteps.concat(outputOtherFlow.Flow
                .SubFlows[i].TargetEntity.FlowSteps);
699         }
700     } else {
701         newSteps = outputOtherFlow.Flow.FlowSteps;
702     }
703     for (let l = 0; l < newSteps.length; l++) {
704         const otherStep = new joint.shapes.basic.Rect({
705             position: { x: 400, y: 20 + 100 * l },
706             size: { width: 220, height: 30 },
707             stepId: newSteps[l].TargetEntity.Id,
708             stepName: newSteps[l].TargetEntity.Name,
709             stepDesc: newSteps[l].TargetEntity.Description,
710             mainLink: undefined,
711             otherLinks: [],
712             tempLinks: [],
713             subMaterials: [],
714             subMaterialsExp: new Map(),
715             subMaterialsExpSorted: [],

```

```

716         subMaterialsError: [],
717         actions: new Map(),
718         events: new Map(),
719         type: 'Temp'
720     }).attr({
721         rect: { fill: this.inColorTemp, stroke: this.
722             outColorTemp, rx: 1, ry: 1 },
723         text: { 'font-size': 20, fill: this.
724             outColorTemp, text: newSteps[l].
725             TargetEntity.Name }
726     });
727
728     graph.addCell(otherStep);
729
730     flowsTemp.get(subMaterialGroups[j].Actions[k].
731         TemporaryOffFlowFlow.Id).push(otherStep);
732 }
733
734 const otherSteps = flowsTemp.get(subMaterialGroups[j].
735     Actions[k].TemporaryOffFlowFlow.Id);
736
737 for (let l = 1; l < otherSteps.length; l++) {
738     const mainLink = new joint.dia.Link({
739         source: otherSteps[l - 1],
740         target: otherSteps[l],
741         subMaterials: [],
742         type: 'Temporary Off Flow'
743     });
744     graph.addCell(mainLink);
745     otherSteps[l - 1].attributes.mainLink = mainLink;
746 }
747
748 hasTemporaryOffFlow = true;
749
750 const tempFlow = flowsTemp.get(subMaterialGroups[j].
751     Actions[k].TemporaryOffFlowFlow.Id);
752 for (let l = 0; l < tempFlow.length; l++) {
753     if (subMaterialGroups[j].Actions[k].
754         TemporaryOffFlowStep.Id === tempFlow[l].
755         attributes.stepId) {
756         temporaryOffFlowStep.push([tempFlow[l],
757             tempFlow[tempFlow.length - 1]]);
758         temporaryOffFlowRestrictions.push(
759             subMaterialsExp.get(j));
760         break;
761     }
762 }
763 }
764 }

```

```

755     }
756   }
757   const nodes = flows.get(stepsExp[i].Flow.Id);
758   for (let j = 0; j < nodes.length; j++) {
759     if (nodes[j].attributes.stepId === stepsExp[i].Step.Id) {
760       nodes[j].attributes.subMaterialsExp = subMaterialsExp;
761       nodes[j].attributes.actions = actionsExp;
762       nodes[j].attributes.events = eventsExp;
763       let subMaterialsExpSorted = [];
764       for (let k = 0; k < subMaterialsExp.size; k++) {
765         subMaterialsExpSorted = subMaterialsExpSorted.concat(
766           subMaterialsExp.get(k));
767       }
768       subMaterialsExpSorted.sort();
769       nodes[j].attributes.subMaterialsExpSorted =
770         subMaterialsExpSorted;
771       nodes[j].attr({
772         rect: { fill: this.actionColor }
773       });
774       if (hasChangeFlowAndStep) {
775         for (let k = 0; k < ChangeFlowAndStepStep.length; k++) {
776           const changeFlowAndStepLink = new joint.dia.Link({
777             source: nodes[j],
778             target: ChangeFlowAndStepStep[k],
779             subMaterials: [],
780             type: 'Change Flow and Step'
781           });
782           graph.addCell(changeFlowAndStepLink);
783           nodes[j].attributes.otherLinks.push([
784             changeFlowAndStepLink,
785             ChangeFlowAndStepRestrictions[k]]);
786         }
787       } else if (hasTemporaryOffFlow) {
788         for (let k = 0; k < temporaryOffFlowStep.length; k++) {
789           const temporaryOffFlowLinkStart = new joint.dia.Link({
790             source: nodes[j],
791             target: temporaryOffFlowStep[k][0],
792             subMaterials: [],
793             type: 'Start Temporary Off Flow'
794           });
795           const temporaryOffFlowLinkEnd = new joint.dia.Link({
796             source: temporaryOffFlowStep[k][1],
797             target: nodes[j],
798             subMaterials: [],
799             type: 'End Temporary Off Flow'
800           });
801           graph.addCell(temporaryOffFlowLinkStart);
802           graph.addCell(temporaryOffFlowLinkEnd);

```

```

799         nodes[j].attributes.tempLinks.push([
            temporaryOffFlowLinkStart,
            temporaryOffFlowRestrictions[k]]);
800         temporaryOffFlowStep[k][1].attributes.tempLinks.push([
            temporaryOffFlowLinkEnd,
            temporaryOffFlowRestrictions[k]]);
801     }
802 }
803 }
804 }
805 }
806
807 // Fill with actual subMaterials and check if valid
808 this.fillSubMaterials(mainSteps[0], allSubMaterials);
809
810 // Check islands
811 const nodes = graph.getElements();
812 for (let i = 0; i < nodes.length; i++) {
813     for (let j = 0; j < nodes[i].attributes.subMaterialsExpSorted.length; j
            ++) {
814         if (nodes[i].attributes.subMaterials.find(subMaterial =>
            subMaterial === nodes[i].attributes.subMaterialsExpSorted[j])
            === undefined) {
815             nodes[i].attributes.subMaterialsError.push(nodes[i].attributes.
                subMaterialsExpSorted[j]);
816             nodes[i].attributes.subMaterialsError.sort();
817             nodes[i].attr({
818                 rect: { fill: this.inColorError, stroke: this.outColorError
                    },
819                 text: { fill: this.outColorError }
            });
820         });
821         const message = 'Trying to use missing Sub-Material ' + nodes[i]
            .attributes.subMaterialsExpSorted[j]
822             + ' on step ' + nodes[i].attributes.stepName;
823         this.bodyItems1.push({
824             id: this.errorCount.toString(),
825             iconClass: "",
826             text: message,
827             disabled: false
828         });
829         this.bodyItems1SubMaterials.push(nodes[i].attributes.
            subMaterialsExpSorted[j]);
830         this.errorCount++;
831         console.log('Trying to use missing Sub-Material ' + nodes[i].
            attributes.subMaterialsExpSorted[j]
832             + ' on step ' + nodes[i].attributes.stepName);
833     }
834 }
835 }

```

```

836
837 // Check future merge
838 for (let i = 0; i < stepsExp.length; i++) {
839     for (let j = 0; j < stepsExp[i].MaterialGroups.length; j++) {
840         if (!stepsExp[i].MaterialGroups[j].NoActions) {
841             if (stepsExp[i].MaterialGroups[j].MergeStep) {
842                 for (let k = 0; k < flows.get(stepsExp[i].MaterialGroups[j]
843 ]MergeFlow.Id).length; k++) {
844                     if (flows.get(stepsExp[i].MaterialGroups[j].MergeFlow.
845 Id)[k].attributes.stepId === stepsExp[i].
846 MaterialGroups[j].MergeStep.Id) {
847                         for (let l = 0; l < stepsExp[i].MaterialGroups[j].
848 TargetEntity.SubMaterialNumbers.split(';').
849 length; l++) {
850                             if (flows.get(stepsExp[i].MaterialGroups[j].
851 MergeFlow.Id)[k].attributes.subMaterials.
852 find(
853 subMaterial => subMaterial ===
854 stepsExp[i].MaterialGroups[j].
855 TargetEntity.SubMaterialNumbers.
856 split(';')[l]) === undefined) {
857                                 console.log(
858                                     'Missing merge of subMaterial ' +
859 stepsExp[i].MaterialGroups[j].
860 TargetEntity.SubMaterialNumbers.
861 split(';')[l]
862 + ' on step ' + flows.get(stepsExp[i].
863 MaterialGroups[j].MergeFlow.Id)[k].
864 attributes.stepName)
865
866                             }
867                         }
868                     }
869                 }
870             }
871         }
872     }
873 }
874
875 // Remove unused nodes
876 const hide = true;
877 if (hide) {
878     this.hideUnusedNodes(graph);
879 }
880
881 // Organize graph
882 joint.layout.DirectedGraph.layout(graph, {
883     nodeSep: 50,
884     edgeSep: 80,
885     rankDir: "TB"
886 }

```

```
871     });
872
873     // Change link aspect
874     paper.model.getLinks().forEach(link => {
875         link.attr({
876             '.marker-target': {
877                 d: 'M 16 0 L 0 8 L 16 16 z',
878                 fill: this._defaultLinkColor,
879                 stroke: this._defaultLinkColor
880             },
881             '.connection': {
882                 stroke: this._defaultLinkColor
883             }
884         });
885     });
886
887     // Highlight node and link when hover
888     paper.on('cell:mouseover', function (cellView) {
889         if (cellView.model.isElement()) {
890             cellView.highlight(null, {
891                 highlighter: {
892                     name: 'stroke',
893                     options: {
894                         padding: 1,
895                         rx: 2,
896                         ry: 2,
897                         attrs: {
898                             'stroke-width': 3,
899                             stroke: cellView.model.attributes.attrs.rect.stroke
900                         }
901                     }
902                 }
903             });
904         } else
905         if (cellView.model.isLink()) {
906             const source = paper.findViewByModel(cellView.model.attributes.
907                 source);
908             const target = paper.findViewByModel(cellView.model.attributes.
909                 target);
910
911             source.highlight(null, {
912                 highlighter: {
913                     name: 'stroke',
914                     options: {
915                         padding: 1,
916                         rx: 2,
917                         ry: 2,
918                         attrs: {
919                             'stroke-width': 3,
```

```
918         stroke: source.model.attributes.attrs.rect.  
919             stroke  
920     }  
921 }  
922 });  
923  
924 target.highlight(null, {  
925     highlighter: {  
926         name: 'stroke',  
927         options: {  
928             padding: 1,  
929             rx: 2,  
930             ry: 2,  
931             attrs: {  
932                 'stroke-width': 3,  
933                 stroke: target.model.attributes.attrs.rect.  
934                     stroke  
935             }  
936         }  
937     });  
938 }  
939 });  
940  
941 // Return to normal when stop hover  
942 paper.on('cell:mouseout', function (cellView) {  
943     if (cellView.model.isElement()) {  
944         cellView.unhighlight(null, {  
945             highlighter: {  
946                 name: 'stroke',  
947                 options: {  
948                     padding: 1,  
949                     rx: 2,  
950                     ry: 2,  
951                     attrs: {  
952                         'stroke-width': 3,  
953                         stroke: cellView.model.attributes.attrs.rect.stroke  
954                     }  
955                 }  
956             }  
957         });  
958     } else  
959     if (cellView.model.isLink()) {  
960         const source = paper.findViewByModel(cellView.model.attributes.  
961             source);  
962         const target = paper.findViewByModel(cellView.model.attributes.  
963             target);
```

```

963         source.unhighlight(null, {
964             highlighter: {
965                 name: 'stroke',
966                 options: {
967                     padding: 1,
968                     rx: 2,
969                     ry: 2,
970                     attrs: {
971                         'stroke-width': 3,
972                         stroke: source.model.attributes.attrs.rect.
                             stroke
973                     }
974                 }
975             }
976         });
977
978         target.unhighlight(null, {
979             highlighter: {
980                 name: 'stroke',
981                 options: {
982                     padding: 1,
983                     rx: 2,
984                     ry: 2,
985                     attrs: {
986                         'stroke-width': 3,
987                         stroke: target.model.attributes.attrs.rect.
                             stroke
988                     }
989                 }
990             }
991         });
992     }
993 });
994
995 // Update details tab
996 paper.on('cell:pointerclick', function (cellView) {
997     if (cellView.model.isElement()) {
998         _this.setSelectedStep(cellView.model.attributes);
999         console.log('Name: ' + cellView.model.attributes.stepName);
1000        console.log('Description: ' + cellView.model.attributes.stepDesc);
1001        console.log('All sub-materials: ' + cellView.model.attributes.
            subMaterials);
1002        console.log('All sub-materials with actions: ' + cellView.model.
            attributes.subMaterialsExpSorted);
1003        for (let i = 0; i < cellView.model.attributes.actions.size; i++) {
1004            console.log('Sub-Materials: ' + cellView.model.attributes.
                subMaterialsExp.get(i) + ' in Actions:');
1005            for (let j = 0; j < cellView.model.attributes.actions.get(i).
                length; j++) {

```



```
1006         console.log(Cmf.Navigo.BusinessObjects.  
1007             ExperimentStepMaterialGroupAction[cellView.model.  
1008                 attributes.actions.get(i)[j]]  
1009                 + ', when: ' +  
1010                 Cmf.Navigo.BusinessObjects.  
1011                     ExperimentStepMaterialGroupActionEvent[cellView.  
1012                         model.attributes.events.get(i)[j]]);  
1013     }  
1014 }  
1015 console.log('All sub-materials with errors: ' + cellView.model.  
1016     attributes.subMaterialsError);  
1017 }  
1018  
1019 if (cellView.model.isLink()) {  
1020     console.log(cellView.model.attributes.type + ' link from step ' +  
1021         cellView.model.attributes.source.attributes.stepName +  
1022         ' to step ' + cellView.model.attributes.target.attributes.  
1023             stepName);  
1024     console.log('SubMaterials: ' + cellView.model.attributes.  
1025         subMaterials);  
1026 }  
1027 });  
1028  
1029 // Zoom parameters  
1030 let scaleValue = 1;  
1031 let transValue = 0;  
1032 const maxZoom = 4;  
1033  
1034 // Zoom  
1035 paper.on('blank:mousewheel', function (_evt, x, y, delta) {  
1036     if (delta === 1 && scaleValue < maxZoom) {  
1037         scaleValue = scaleValue * 1.2;  
1038         transValue = -1 / 6;  
1039         paper.scale(scaleValue, scaleValue);  
1040         graph.translate(x * transValue, y * transValue);  
1041     }  
1042     if (delta === -1 && scaleValue > 1 / maxZoom) {  
1043         scaleValue = scaleValue / 1.2;  
1044         transValue = 1 / 5;  
1045         paper.scale(scaleValue, scaleValue);  
1046         graph.translate(x * transValue, y * transValue);  
1047     }  
1048 }  
1049 });  
1050  
1051 paper.on('cell:mousewheel', function (_cellView, _evt, x, y, delta) {  
1052     if (delta === 1 && scaleValue < maxZoom) {  
1053         scaleValue = scaleValue * 1.2;  
1054         transValue = -1 / 6;  
1055         paper.scale(scaleValue, scaleValue);
```

```
1047         graph.translate(x * transValue, y * transValue);
1048     }
1049     if (delta === -1 && scaleValue > 1 / maxZoom) {
1050         scaleValue = scaleValue / 1.2;
1051         transValue = 1 / 5;
1052         paper.scale(scaleValue, scaleValue);
1053         graph.translate(x * transValue, y * transValue);
1054     }
1055 });
1056
1057 let dragStartPosition;
1058
1059 // Pan
1060 paper.on('blank:pointerdown', function (_evt, x, y) {
1061     dragStartPosition = { x: x * scaleValue, y: y * scaleValue };
1062 });
1063
1064 paper.on('cell:pointerup blank:pointerup', function () {
1065     dragStartPosition = undefined;
1066 });
1067
1068 this.holder.nativeElement.addEventListener('mousemove', event => {
1069     if (dragStartPosition) {
1070         graph.translate(
1071             (event.offsetX - dragStartPosition.x) / scaleValue,
1072             (event.offsetY - dragStartPosition.y) / scaleValue);
1073         dragStartPosition.x = event.offsetX;
1074         dragStartPosition.y = event.offsetY;
1075     }
1076 });
1077 }
1078
1079 public ngOnInit(): void {
1080 }
1081
1082 //endregion
1083 }
1084
1085 @Module({
1086     imports: [
1087         DropdownModule,
1088         PageSplitterModule,
1089         CollapsiblePanelsMenuModule,
1090         CollapsiblePanelsMenuPanelBarModule,
1091         PanelBarModule,
1092         PropertyEditorModule
1093     ],
1094     declarations: [GraphView],
1095     defaultRoute: GraphView,
```

```
1096     exports: [GraphView]  
1097   })  
1098   export class GraphViewModule { }
```

---



# References

- [1] A. Abreu, J. Requeijo, J. M. F. Calado, and A. Dias. Definition of strategies based on simulation and design of experiments. In John R. Wagner, Eldridge M. Mount, and Harold F. Giles, editors, *ISEL - Eng. Mecan. - Comunicações*. Instituto Superior de Engenharia de Lisboa, 2019.
- [2] Agidens. MES, optimization of your production environment. <https://www.agidens.com/en/automation/Our-expertise/Manufacturing-execution-systems>.
- [3] Oleg Alexandrov. Siligon ingot at Intel Museum. <https://commons.wikimedia.org/wiki/File:Siligon{ }ingot{ }at{ }Intel{ }Museum.JPG>, 2013.
- [4] J. Antony, S. Coleman, D. C. Montgomery, M. J. Anderson, and R. T. Silvestrini. Design of Experiments for non-manufacturing processes: Benefits, challenges and some examples. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 225(11):2078–2087, 2011.
- [5] June Young Bang, Jae Hun Kang, Bong Kyun Kim, and Yeong Dae Kim. Multi-product lot merging/splitting algorithms for a semiconductor wafer fabrication. In *Proceedings of the Winter Simulation Conference*, pages 2209–2215, Miami, dec 2008. IEEE.
- [6] Oliver Bastert and Christian Matuszewski. Layered Drawings of Digraphs. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, chapter 5, pages 87–120. Springer, Berlin, 2001.
- [7] Giuseppe Di Battista, Ioannis G. Tollis, Peter Eades, and Roberto Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, New Jersey, 1999.
- [8] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A Taxonomy and Survey of Dynamic Graph Visualization. *Computer Graphics Forum*, 36(1):133–159, jan 2017.
- [9] Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, apr 2014.
- [10] Ulrik Brandes. Drawing on Physical Analogies. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, chapter 4, pages 71–86. Springer, Berlin, 2001.
- [11] Jurgen Branke. Dynamic Graph Drawing. In Michael Kaufmann, editor, *Drawing Graphs: Methods and Models*, chapter 9, pages 228–246. Springer, Berlin, 2001.
- [12] Joy Chao. Graph Search Algorithms: Depth-first and Breadth-first. <https://neo4j.com/blog/graph-search-algorithm-basics/>.

- [13] R. Chao, M. Breton, B. L'Herron, B. Mendoza, R. Muthinti, F. Nelson, A. De La Pena, F. L. Le, E. Miller, S. Sieg, J. Demarest, P. Gin, M. Wormington, A. Cepler, C. Bozdog, M. Sendelbach, S. Wolfling, T. Cardinal, S. Kanakasabapathy, J. Gaudiello, and N. Felix. Advanced in-line metrology strategy for self-aligned quadruple patterning. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 9778, 2016.
- [14] Chen Fu Chien, Kuo Hao Chang, and Wen Chih Wang. An empirical study of design-of-experiment data mining for yield-loss diagnosis for semiconductor manufacturing. *Journal of Intelligent Manufacturing*, 25(5):961–972, 2014.
- [15] client IO s.r.o. JointJS - JavaScript diagramming library - Getting started. <https://resources.jointjs.com/tutorial>.
- [16] client IO s.r.o. JointJS: Visualize and interact with diagrams and graphs. <https://www.jointjs.com/opensource>.
- [17] Critical Manufacturing. Critical Manufacturing - Complete Modular Solution. <https://www.criticalmanufacturing.com/en/critical-manufacturing-mes/complete-modular-solution>.
- [18] Critical Manufacturing. Critical Manufacturing - Experiments Management. <https://www.criticalmanufacturing.com/en/critical-manufacturing-mes/industry-4-0/experiments-management>.
- [19] Critical Manufacturing. Critical Manufacturing - Manufacturing Technology Solution Provider | Company Overview. <https://www.criticalmanufacturing.com/en/company/overview>.
- [20] Critical Manufacturing. Critical Manufacturing - Other Industries. <https://www.criticalmanufacturing.com/en/industries/other-industries>.
- [21] Critical Manufacturing. Critical Manufacturing V7 Brochure. Technical report.
- [22] Critical Manufacturing. Critical Manufacturing - Blog about MES, Industry 4.0, Quality, Manufacturing, Automation & much more - Genealogy. <https://www.criticalmanufacturing.com/en/newsroom/blog/posts/blog/genealogy{#}.XtufhmkjMV>, 2014.
- [23] Critical Manufacturing and Iyno Advidores. IIoT Has a ‘Thing’ for MES Why IoT Platforms Won’t Replace MES. Technical report, 2018.
- [24] Sanjoy Dasgupta. Learning Polytrees. In Kathryn Blackmond Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 134–141, Stockholm, 1999. Morgan Kaufmann Publishers.
- [25] B. Saenz De Ugarte, A. Artiba, and R. Pellerin. Manufacturing execution system - A literature review. *Production Planning and Control*, 20(6):525–539, 2009.
- [26] Christian A. Duncan and Michael T. Goodrich. Planar Orthogonal and Polyline Drawing Algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 7, pages 223–246. CRC Press, Boca Raton, 2014.

- [27] Markus Eiglsperger, Sándor P Fekete, and Gunnar W Klau. Orthogonal Graph Drawing. In Michael Kaufmann, editor, *Drawing Graphs: Methods and Models*, chapter 6, pages 121–171. Springer, Berlin, 2001.
- [28] Ernesto Estrada. Graph and Network Theory. In Michael Grinfeld, editor, *Mathematical Tools for Physicists*, pages 111–158. Wiley, 2013.
- [29] Shimon Even. *Graph algorithms*. Cambridge University Press, New York, 2nd edition, 2012.
- [30] Muhammad Arsalan Farooq, Henriqueta Nóvoa, António Araújo, and Sergio M.O. Tavares. An innovative approach for planning and execution of pre-experimental runs for Design of Experiments. *European Research on Management and Business Economics*, 22(3):155–161, 2016.
- [31] Victor Ferreira. Governo relançou a Indústria 4.0, a indústria pediu “um Estado 4.0”. <https://www.publico.pt/2019/04/09/economia/noticia/governo-falou-industria-40-industria-pediou-estado-40-1868683>, apr 2019.
- [32] Rudolf Fleischer and Colin Hirsch. Graph Drawing and Its Applications. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, chapter 1, pages 1–22. Springer, Berlin, 2001.
- [33] Jean-Claude Fournier. *Graphs Theory and Applications: With Exercises and Problems*. Wiley, 2011.
- [34] Harold N. Gabow. Searching. In Jonathan L Gross and Jay Yellen, editors, *Handbook of Graph Theory*, chapter 10-1, pages 953–984. CRC Press, 2004.
- [35] Gallagher Fluid Seals Inc. Basic Semiconductor Manufacturing Process. <https://www.gallagherseals.com/blog/semiconductor-manufacturing-process/>.
- [36] Gartner Peer Insights. Manufacturing Execution Systems (MES) Software Reviews. <https://www.gartner.com/reviews/market/manufacturing-execution-systems>.
- [37] Harold F. Giles Jr, Eldridge M. Mount III, and John R. Wagner Jr. *Extrusion: The Definitive Processing Guide and Handbook*. Elsevier, Waltham, second edition, 2014.
- [38] Jonathan L. Gross and Jay Yellen. Fundamentals of graph theory. In Jonathan L. Gross and Jay Yellen, editors, *Handbook of Graph Theory*. CRC Press, 2004.
- [39] J. N. D. Gupta, R. Ruiz, J. W. Fowler, and S. J. Mason. Operational planning and control of semiconductor wafer production. *Production Planning and Control*, 17(7):639–647, oct 2006.
- [40] Patrick Healy and Nikola S Nikolov. Hierarchical Drawing Algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 13, pages 409–453. CRC Press, Boca Raton, 2014.
- [41] Hexacta. 7 tips to know when to use SCRUM in your project. <https://www.hexacta.com/7-tips-to-know-when-to-use-scrum-in-your-project/>, 2018.

- [42] Hitachi High-Tech GLOBAL. Semiconductor Glossary. <https://www.hitachi-hightech.com/global/products/device/semiconductor/words.html#{#}Semiconductor>.
- [43] Hitachi High-Tech GLOBAL. Semiconductor manufacturing process. <https://www.hitachi-hightech.com/global/products/device/semiconductor/process.html>.
- [44] Weidong Huang, Mao Lin Huang, and Chun Cheng Lin. Evaluating overall quality of graph visualizations based on aesthetics aggregation. *Information Sciences*, 330:444–454, feb 2016.
- [45] Konstantinos G. Kakoulis and Ioannis G. Tollis. Labeling Algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 15, pages 489–515. CRC Press, Boca Raton, 2014.
- [46] T. S. Kim. Chip speed prediction model for optimization of semiconductor manufacturing process using neural networks and statistical methods. In *Lecture Notes in Computer Science*, volume 3498, pages 845–850, 2005.
- [47] Moritz Klammler, Tamara McHedlidze, and Alexey Pak. Aesthetic discrimination of graph layouts. In Biedl T. and Kerren A., editors, *Graph Drawing and Network Visualization. GD 2018. Lecture Notes in Computer Science*, volume 11282, pages 169–184. Springer, sep 2018.
- [48] Stephen G. Kobourov, Sergey Pupyrev, and Bahador Saket. Are Crossings Important for Drawing Large Graphs? In Duncan C. and Symvonis A., editors, *Graph Drawing. GD 2014. Lecture Notes in Computer Science*, volume 8871, pages 234–245. Springer, Berlin, 2014.
- [49] KPMG Portugal. Indústria 4.0- Fase II. Technical report, República Portuguesa, COTEC Portugal, IAPMEI, 2019.
- [50] Ashok Kumar. Are MES systems dead? How IoT and AI are transforming the shop floor. <https://www.ibm.com/blogs/internet-of-things/iot-mes-is-dead/>, 2018.
- [51] J. Kuo and C. Kuo. *Optimal parameter design in flip chip micro-machining process for solder residue by using design of experiments approach*, volume 126-128 of *Advanced Materials Research*. 2010.
- [52] Yongxin Liao, Fernando Deschamps, Eduardo de Freitas Rocha Loures, and Luiz Felipe Pierin Ramos. Past, present and future of Industry 4.0 - a systematic literature review and research agenda proposal. *International Journal of Production Research*, 55(12):3609–3629, 2017.
- [53] Giuseppe Liotta and Roberto Tamassia. Drawings of graphs. In Jonathan L Gross and Jay Yellen, editors, *Handbook of Graph Theory*, chapter 10-3, pages 1015–1045. CRC Press, 2004.
- [54] Yang Liu and Jingshan Li. Modelling and analysis of split and merge production systems with bernoulli reliability machines. *International Journal of Production Research*, 47(16):4373–4397, jan 2009.



- [55] Ser Lee Loh, Chin Kim Gan, Tau Han Cheong, Shaharuddin Salleh, and Nor Haniza Sarmin. An overview on network diagrams: Graph-based representation. *Journal of Telecommunication, Electronic and Computer Engineering*, 8(2):83–86, 2016.
- [56] C. J. Mahandran, A. Y. A. Fatah, N. A. Bani, H. M. Kaidi, M. N. B. Muhtazaruddin, and M. E. Amran. Thermal oxidation improvement in semiconductor wafer fabrication. *International Journal of Power Electronics and Drive Systems*, 10(3):1141–1147, 2019.
- [57] Soujanya Mantravadi and Charles Møller. An overview of next-generation manufacturing execution systems: How important is MES for industry 4.0? *Procedia Manufacturing*, 30:588–595, 2019.
- [58] Critical Manufacturing. Critical Manufacturing - Overview. *Critical Manufacturing*. <https://www.criticalmanufacturing.com/en/company/overview>. (accessed Jun. 2, 2020).
- [59] Critical Manufacturing. Critical Manufacturing - Sobre nós. *Linkedin*. <https://pt.linkedin.com/company/critical-manufacturing>. (accessed Jun. 2, 2020).
- [60] Manufacturing Technology Insights. Top 10 MES Solution Companies - 2019. <https://manufacturing-execution-system.manufacturingtechnologyinsights.com/vendors/top-mes-solution-companies.html>.
- [61] Stephen B. Maurer. Directed acyclic graphs. In Jonathan L Gross and Jay Yellen, editors, *Handbook of Graph Theory*, chapter 3-2, pages 142–155. CRC Press, 2004.
- [62] Gary S. May and Costas J. Spanos. *Fundamentals of Semiconductor Manufacturing and Process Control*. Wiley, New Jersey, 2006.
- [63] Julian Marius Müller, Daniel Kiel, and Kai Ingo Voigt. What drives the implementation of Industry 4.0? The role of opportunities and challenges in the context of sustainability. *Sustainability (Switzerland)*, 10(1):247, 2018.
- [64] Nachosan. Wafers at the National Museum of Scotland. <https://commons.wikimedia.org/wiki/File:Wafers{ }National{ }Museum{ }of{ }Scotland{ }20.JPG>.
- [65] Gabriele Neyer. Map Labeling with Application to Graph Drawing. In Michael Kaufmann, editor, *Drawing Graphs: Methods and Models*, chapter 10, pages 247–273. Springer, Berlin, 2001.
- [66] Boštjan Pajntar. Overview of Algorithms for Graph Drawing. In *Conference on Data Mining and Data Warehouses*, 2006.
- [67] S. Pampuri, G. A. Susto, J. Wan, A. Johnston, P. O’Hara, and S. McLoone. Insight extraction for semiconductor manufacturing processes. In *IEEE International Conference on Automation Science and Engineering*, volume 2014-January, pages 786–791, 2014.
- [68] Chris Parsons. Critical Manufacturing - Blog about MES, Industry 4.0, Quality, Manufacturing, Automation & much more - Effective, Speedy Experiments. <https://www.criticalmanufacturing.com/en/newsroom/blog/posts/blog/effective-speedy-experiments{ }.Xu0uIkVKiHs>, jul 2019.

- [69] Peellden. 12-inch silicon wafer. <https://commons.wikimedia.org/wiki/File:12-inch{ }silicon{ }wafer.jpg>, 2011.
- [70] C. Qi, A. I. Sivakumar, and S. B. Gershwin. Impact of production control and system factors in semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 21(3):376–389, 2008.
- [71] Jonathan Vasconcelos Rodrigues. *Indústria 4.0-Desenvolvimento de um Manufacturing Execution System [Dissertação de mestrado]*. PhD thesis, Coimbra, 2018.
- [72] Andreja Rojko. Industry 4.0 concept: Background and overview. *International Journal of Interactive Mobile Technologies*, 11(5):77–90, 2017.
- [73] Bruno Scibilia. A DOE in a Manufacturing Environment (Part 1). <https://blog.minitab.com/blog/applying-statistics-in-quality-projects/a-doe-in-a-manufacturing-environment-part-1>, mar 2016.
- [74] Scrum.org. What is SCRUM. <https://www.scrum.org/resources/what-is-scrum>.
- [75] Martín Tanco, Elisabeth Viles, Laura Ilzarbe, and María Jesús Álvarez. Is design of experiments really used ? A survey of Basque industries Basque industries. *Journal of Engineering Design*, 19(5):447–460, 2008.
- [76] ASM Pacific Technology. ASMPPT To Invest In SMART Factory Software Solutions. *ASM Pacific Technology*, 2018. <https://www.asmpacific.com/en/media-release/81-asmppt-to-invest-in-smart-factory-software-solutions>.
- [77] K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms*. Wiley, 1992.
- [78] Yu-Ping Wang. Methods of manufacturing semiconductor devices and transistors. <https://patents.google.com/patent/US8377779B1/en>, jan 2013.
- [79] René Weiskircher. Drawing Planar Graphs. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, chapter 2, pages 23–45. Springer, Berlin, 2001.
- [80] Hong Xiao. *Introduction to Semiconductor Manufacturing Technology*. SPIE, Washington, 2nd edition, nov 2012.
- [81] Yong Yin, Kathryn E. Stecke, and Dongni Li. The evolution of production systems from Industry 2.0 through Industry 4.0. *International Journal of Production Research*, 56(1-2):848–861, jan 2018.
- [82] H. Younan, L. Y. Ping, N. R. Rao, and T. Qinghua. Studies on galvanic corrosion on floating and grounded bondpads in wafer fabrication. volume 25, pages 219–223, 2009.
- [83] yWorks GmbH. Developer’s Guide: Automatic Label Placement. <https://docs.yworks.com/yfiles-html/dguide/layout/label{ }placement.html>, 2020.
- [84] Keliang Zhou, Taigang Liu, and Lifeng Zhou. Industry 4.0: Towards future industrial opportunities and challenges. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015*, pages 2147–2152. IEEE, 2016.