ÉCOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTÈMES

Projet de Fin d'Études

En Vue de l'Obtention du Titre

Ingénieur d'État en Informatique

Filière : Génie Logiciel

Performance Benchmarking of Executable Domain Specific Languages

---

Réalisé par :

AJABRI Hiba

Encadré par

Pr.NASSAR Mahmoud

Et par

Dr.MOTTU Jean-Marie

Dr.BOUSSE Erwan

Dr.TISI Massimo

Membres de jurys

Pr.BOUNABAT Bouchaib(President)

Pr.EL HAMLAOUI Mahmoud (Examiner)

Année universitaire 2021-2022

Dédicaces

À ma chère mère *SADIKI Soumia*

Qui m'a toujours comblé d'amour et d'affection. Son dévouement sans pareil, je n'oublierai jamais son sacrifice et son soutien qui ont conduit à mon succès aujourd'hui.

À mon adorable père *AJABRI Ameur*

À l'homme, à qui je dois tout, pour toute la compréhension, l'aide et la confiance qu'il m'a toujours accordées, pour ce que je deviens aujourd'hui. En témoignage de ma profonde gratitude et respect.

Merci d'être mes parents.

À mes deux Sœurs, À mon Frère

Je vous aime. Que Dieu vous apporte le bonheur, vous aide à réaliser tous vos vœux et vous offre un avenir plein de succès.

À ma famille et mes chers amis

Pour tout le soutien que vous m'avez offert, et pour les bons moments que nous avons passés ensemble, je vous dis MERCI.

À tous ceux que j' aime et qui m'aiment,

je dédie ce travail.

Remerciements

« وَمَا رَمَيْتَ إِذْ رَمَيْتَ وَلَكِنَّ ٱللَّهَ رَمَىٰ »

Avant tout et après tout, ma réussite n'est due qu'à **Allah**. Le Tout Puissant et Miséricordieux, qui m'a donné la force et la patience d'accomplir ce travail.

Ce projet est le fruit des conseils et critiques bienveillantes d'un grand nombre de personnes. Je tiens à les remercier ici et leur faire part de toutes mes gratitudes, pour avoir été à l'écoute et toujours d'une aide précieuse.

J'adresse à travers ces courtes lignes mes remerciements les plus sincères et ma profonde gratitude envers mes encadrants de stage ***Dr. Mottu Jean-Marie***, ***Dr. Bousse Erwan*** et ***Dr. Tisi Massimo***, qui ont eu le soin de me guider, d'examiner et encadrer ce travail et ainsi m'offrir une véritable opportunité d'apprentissage. Leurs compétences, leurs rigueurs et l'esprit humain que j'ai ressenti, ont suscité en moi une grande admiration et un profond respect.

Je voudrais également profiter de cette occasion pour remercier ***Faezeh Khorram*** pour son soutien et sa présence, ainsi que toute l'équipe ***NaoMod*** pour l'accueil chaleureux. Je voudrais également témoigner ma gratitude au laboratoire ***LS2N*** pour l'opportunité du stage , et merci à tous les contributeurs et les surveillants du projet RODIC.

Je tiens à remercier vivement mon encadrant à l'Ecole National Supérieur d'Informatique et d'Analyse du Systèmes, ***Mr. Pr. NASSAR Mahmoud*** pour l'intérêt qu'il a porté à mon projet, ses directives précieuses et sa contribution au succès de mon travail. Je remercie vivement les membres de jurys qui ont acceptés d'évaluer mon travail ***Pr. BOUNABAT Bouchaib*** et ***Pr. EL HAMLAOUI Mahmoud*** et je tiens à exprimer ma reconnaissance à l'ensemble du corps professoral de l'***ENSIAS***, et je suis très fière d'appartenir à cette grande école.

Un merci, encore une fois, à toute personne ayant contribué de près ou de loin à la réalisation de ce projet.

Abstract

This document is the result of my work as part of the PFE internship of the engineering studies - Software Engineering.

The subject of the internship project is ***«Performance Benchmarking of Executable Domain Specific Languages »***. Where the main idea is to help the industry 4.0 in accelerating the RMS reconfiguration process in both the technical and usage aspects, using a multidisciplinary and model-based software engineering (MBSE) approach.

My mission is to build a groundwork for the thesis subject that consists of evaluating the RMS reconfiguration with non-functional testing, by taking into account the KPIs defined in the xDSL (eXecutable Domain Specific Language). Therefore, the first step consist of adding the performance-based concepts to the xDSL and then evaluate the established configuration based on different test scenarios. Ultimately, consider how to represent and store the results obtained from performance benchmarks simulations.

This internship heavily relies on software skills and prior knowledge of the Model driven Engineering.

Key words: Benchmarking, Domain Specific Languages, RMS, Model Driven Engineering (MDE), xDSL, KPI, Meta-Model.

Résumé

Le présent document est le fruit de mon travail dans le cadre du stage de fin d'étude en cycle ingénieur - filière Génie Logiciel.

Le projet du stage avait pour sujet ***«Performance Benchmarking of Executable Domain Specific Languages »***. L'idée principale est d'aider l'industrie 4.0 à accélérer le processus de reconfiguration du RMS dans les aspects techniques, en s'appuyant sur une approche multidisciplinaire et sur l'ingénierie dirigée par les modèles (MDE).

La mission de ce stage est de construire une base pour un sujet de thèse qui consiste à évaluer la reconfiguration des RMSs avec des tests non-fonctionnels, en tenant compte des KPIs définis dans le xDSL (eXecutable Domain Specific Language). Ainsi, la première étape consiste à annoter les méta-modèles par des concepts de performance, puis la configuration considérée est évaluée en effectuant des scénarios de test et enfin les résultats obtenus sont stockés pour pouvoir mener une étude d'optimisation.

Ce stage s'appuie fortement sur des compétences logicielles et les connaissances préalables de l'ingénierie pilotée par les modèles.

Mot clés: Benchmarking, Domain Specific Languages, RMS, Model Driven Engineering (MDE), xDSL, KPI, Méta-Modèle.

Abbreviations list

Abbreviation	Designation
API	Application Programming Interface
AQL	Acceleo Query Language
ATL	ATLAS Transformation Language
CIM	Computational Independent Model
DMS	Dedicated Manufacturing Systemes
DSL	Domain-Specific Languages
EEL	Energy Estimation Language
EEM	Energy Estimation Models
EMF	Eclipse Modeling Framework
FSM	Flexibles manufacturing systems
GMF	Graphical Modeling Framework
IMT	Institut Mines-Télécom
IUT	Institut Universitaire de Technologie
K3	Kermeta 3
KPI	Key-Performance Indicators
LAB-STICC	Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance
LS2N	Laboratoire de Sciences et du Numériques de Nantes
MBE	Model-based Engineering
MBSE	Model-Based Software Engineering
MDA	Model Driven Architecture
MDE	Model Driven Engineering
MSE	Model Specific Event

Abbreviation	Designation
MUT	Model Under Test
OCL	Object Constraint Language
OMG	Object Management Group
OP1	Operator 1
PEL	Performance Estimation Language
PIM	Platform Independent Model
PSM	Platform Specific Model
RMS	Reconfigurable Manufacturing Systems
RODIC	Rapid recOnfiguration of manufacturing systems : a model-baseD software engineering and human Interaction Coupled approach
SLS	Science du Logiciel et des Systèmes Distribués
SMM	Structured Metric Metamode
TDL	Test Definition Language
UML	Unified Modeling Language
WBS	Work Breakdown Structure
xDSL	eXecutable Domain Specific Language

Contents

List of Figures

General Introduction

Industry 4.0 is facing more challenges than ever before, and this was clearly demonstrated when the Corona crisis erupted. Besides all unprecedented transformations and the tremendous influence that has spread in all fields and life mechanisms, the industry undergoes a radical change, the installation of new concepts and paradigms to overcome the hurdles and follow the cadence of development at the large scale.

COVID-19 had two opposite sides at the same time. It was threatening the economies of societies and changing the balance of power in the world, as much as it was mutation in the industry. When all public services ceased and sanitary confinement became mandatory, the industry was working harder than ever. Assembly lines automobiles turn into a respirator manufacturing line, which is how General Motors, Ford and Tesla proceeded in order to tackle the pandemic. They were transforming their factories to manufacture various medical and health components, but it was not that easy. A highly automated assembly line used to produce vehicles will not be transformed overnight. The same goes for textile company «Les Tissages de Charlieu», which have converted her production lines to make protective masks, specifying that they have the capacity to produce 30,000 to 50,000 masks per day. Like other industries, the cosmetics industry modifies production lines to supply hydroalcoholic gel to hospitals and healthcare establishments. Moreover, all these transformations of industrial systems are supposed to meet a large number of requests that the pandemic sparked, estimated to be in thousands around the world.

Figure 1: Industry

Not all these transformations were easy for all industries, lack of a reconfiguration strategy and existing methodologies, many of the companies closed. Others succeed to keep up with the sudden spike in demand despite all the constraints, but still cost them a lot. The secrets behind the capacity to survive in the face of the largest economic crisis in recent history, refers to the previous adoption of digital

mechanisms in the production systems in the industry. Those companies that had already scaled digital technologies found themselves better positioned to respond to the crisis, as the digital infrastructure had been crucial and essential.

Confronted with recent challenges, old structures are crumbling. There are a bunch more aspects and examples that stimulate Industry 4.0 to move forward, and accelerate its ability to install patterns to deal with unpredictable situations, but also to promote better production and to evolve the company's basic concepts, by refining the strategy and developing the business process. The Covid-19 crisis has shown us the weaknesses of industrial digitization, but also given us the opportunity to locate the problem, so that we can pick the suitable tools, herein is the necessity to adopt production systems capable of being reconfigured according to a particular situation.

Therefore, in this report, I will present the work carried out within this internship in six chapters:

- The first chapter presents the general context of the project.
- The second chapter will outline the scientific basic-knowledge to build a consistent view for the other sections.
- The third chapter introduces the documentation section for the internship subject.
- Fourth chapter will disclose the needs through a whole functional study.
- The fifth chapter will focus on the approaches and solutions that have been established to tackle the issue.
- The sixth chapter will implement the notions previously seen in other sections.

Chapter 1

General context of the internship

This chapter serves as a presentation of the host organization, the problematic, the objectives and management.

1.1 The context of the PFE internship

This internship position was opened in the context of the RODIC project – Rapid recOnfiguration of manufacturing systems: a model-baseD software engineering and human Interaction Coupled approach RODIC, for a Master student with the perspective to be recruited as a PhD student for the next 3 years. The research domain is Software Engineering, with the goal to help Industry 4.0 when reconfiguring production chains by the use of techniques and platforms based on Model Driven Engineering, Domain-Specific Languages (DSL), Software Testing, Human-Machine Interfaces. This internship subject focuses on the design and the execution of benchmarks to evaluate the performance of chain configurations designed with executable DSLs.

1.2 The academic context

This internship is attached to the NaoMod team and it takes place in the laboratory LS2N (Laboratoire de Sciences et du Numériques de Nantes), collaboration with LAB-STICC (Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance), at IMT atlantique bretagne-pays de la loire, Nantes university and IUT des Pays de la Loire.

1.3 Industrial context

Industry is the main sector of RODIC, and subsequently the internship subject is based on industrial concepts and tools. Thus, the ongoing development of the project requires the involvement of industry experts to define the domain-specific concepts, which can help to correctly understand the business process and identify the underlying problem. To assert this, the RODIC relies on the complementary expertise of French industrial teams belonging to CNRS UMR laboratories: PSI team affiliated with the LS2N laboratory specializing in industrial engineering, this team will bring its expertise in manufacturing control modeling and industrial contextualization. In addition to the CSIP team of ICUBE laboratory.

The work performed as part of this internship is not only academically targeted but also related to real work cases. Partnerships have been established with industrial companies, in particular with EUROPE TECHNOLOGIES, an expert in the industrialization and manufacture of composite, metal parts and assemblies. There's also the case study of LivingPackets, the Nantes-based startup that invented the «La box», a reusable, connected and secure e-commerce.

1.4 The business context

The evolution of industrial systems to so-called «Industry 4.0», «Factory of Future» or «CyberPhysical Production Systems», is mainly based on the development of highly connected resources throughout the whole production process and beyond. The RODIC project focuses on the reconfiguration phase: when switching from one configuration to another, a Reconfigurable Manufacturing Systems [RMS] needs to reconfigure both hardware and software components. A model of the production

chain can then be modeled, the scenario can be verified, and DSL can be used to simulate the execution. The approach of RODIC is to consider most of the reconfiguration using the same models for the definition, verification, evaluation, validation, and code generation. During this internship, we focus on the evaluation: when a new configuration is being designed, the operator needs to evaluate its performance before the (costly, time consuming) reconfiguration of the production chain in the physical world. This evaluation must rely on the existing models used for designing the new configuration and verifying its correctness.

1.5 Presentation of the host organization

The Nantes Digital Sciences Laboratory (LS2N) was created in January 2017 with the intention of significantly improving the visibility of digital science research in Nantes. The LS2N is essentially the merger of two institutions, the ex-UMR IRCCyN (Institute for Research in Communications and Cybernetics of Nantes) and LINA (Computer Science Laboratory of Nantes Atlantique). This Unit of scientific research federates the research strengths in the digital field of three higher education establishments (University of Nantes, Centrale Nantes, IMT Atlantique/Nantes campus), CNRS and its partner Inria. The LS2N is the largest public research unit on the Nantes site and in the «Pays de la Loire» region. The workforce of the LS2N amounts to more than 480 people, half of whom are permanent, spread over five sites: Faculty of Science and Technology, Centrale Nantes, Polytech Nantes, IMT Atlantique and IUT of Nantes. LS2N research is composed of 24 research teams, structured around five major scientific poles: Signals, Images, Ergonomics and Languages; Data and Decision Sciences; Science of Software and Distributed Systems; Design and Operation of Systems and Robotics, Processes, Calculation. In addition to these areas, there are application themes: Company of the future, Energy management and control of environmental impacts, Vehicles and mobility and Creation, digital culture and society.

The LS2N depends on 4 supervisory authorities: the CNRS (more precisely through its main association with the Institute of Information Sciences and their Interactions (INS2I), as well as the Institute of Engineering Sciences and Systems (INSIS) and the Institute of Biological Sciences (INSB), characterized by interdisciplinary activity, and the supervisions of the site, which are Nantes University, Centrale Nantes, and the IMT Atlantique. Inria is a partner of the LS2N, and they are united by several teams-project.

1.6 Organizational chart

As mentioned before, LS2N research is carried out within 24 research teams, structured around five poles. This training was hosted by the Naomod Group who is part of the Science of Software and Distributed Systems pole, headed by Dalila Tamzalit.

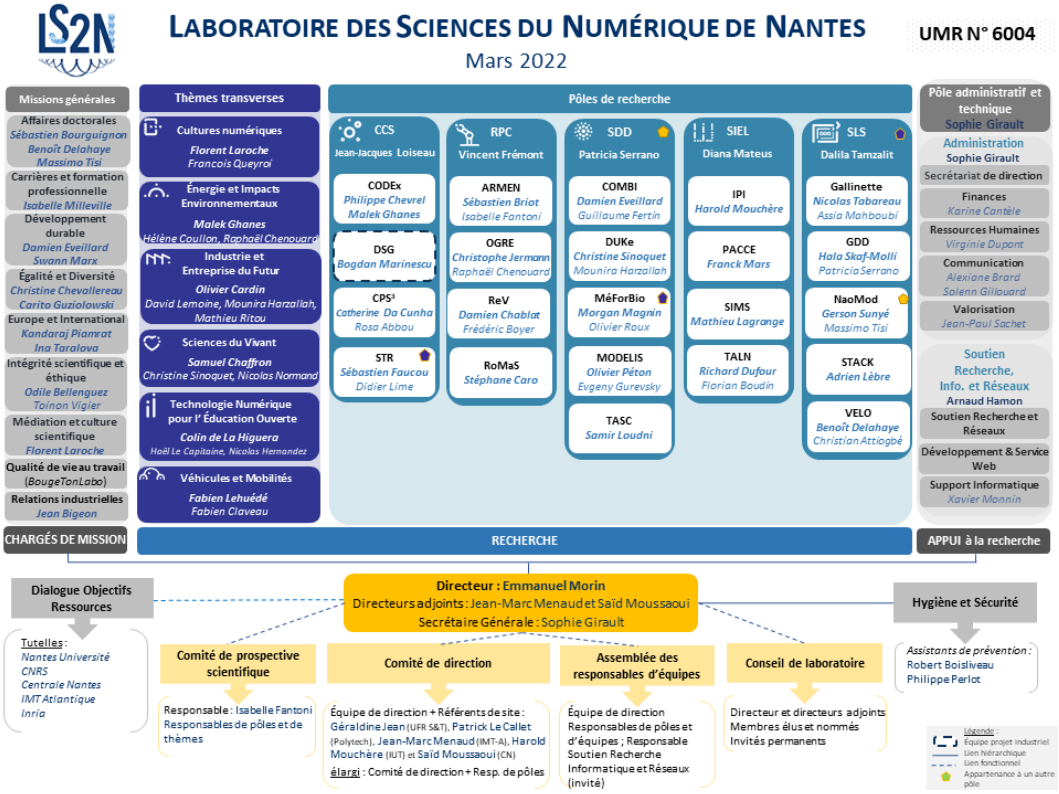

Figure 1.1: LS2N chart

Science of Software and Distributed Systems: Based on the enormous rise in the number and the complexity of the calculators (objects connected to data centers) and the migration from centralized systems to distributed systems, the SLS pole pays particular attention to the engineering of programming, software and systems, both in an embedded systems engineering and highly distributed context, by exploring the complementary facets of languages, models and systems.

1.7 Co-supervision of the NaoMod teams and LABSTICC

For this internship, two laboratories involved and collaborated to supervise and constantly conduct the ongoing development of the project.

NaoMod, the Nantes Software Modeling Group, is a joint IMT Atlantique and LS2N research team resulting from a long-time specialization and research expertise

on MDE (Model Driven Engineering) in the area of Nantes-France since 1990. The group have been working intensively on improving and promoting the base techniques, best practices and actual usages of MDE by paying equally attention to both the academic & industrial problems, and notably investigating on the benefits that can be brought by a rigorous use of models in a software engineering context. From this research effort, an open source model-based toolbox is being developed and proposed (notably via the worldwide Eclipse Community) to software engineers & architects with the objective of improving their productivity as well as increasing the quality of the applications they design and build.

The Laboratory of Sciences and Techniques of Information, Communication and Knowledge (Lab-STICC), with its dual attachment to the INS2I and INSIS institutes of the CNRS, is a research unit historically recognized in «Bretagne Océane» and in France in the field of STIC. It displays a proven ability to cover a wide scientific spectrum around digital sciences, with a particular interest in various disciplinary fields (Information Theory, Waves & Materials, Data Sciences, Communication and signal detection, Human-Machine Interfaces, etc.) subscribing into multiple themes/application sectors: communicating objects, space, health, security, robotics, etc.

1.8 Analysis of the existing situation

As stated before, the coronavirus has disclosed vulnerabilities in many fields. However, the nature of industrial evolution expresses new stakes each time. With the aim of placing the customer at the heart of the business, and not simply to satisfy consumers, but also to anticipate their needs to better meet them and to build a relationship of trust over time. The industry evolves its methods to deliver value faster, with greater quality and predictability, and greater aptitude to respond to change. The ultimate value is to personalize the product as much as possible and remain effective and efficient in a competitive capitalist market. To do so, a clear strategy is needed to establish, drive, and sustain the transition and ensure its success, and especially when responding to a sudden surge in demand, which in turn requires a specific scaling framework.

In the midst of this evolution, the production lines are being transformed, some stations may be set up for new tasks, others may be duplicated according to the expected production rate, or may be completely removed or replaced when production objectives change. All these changes affect the production system, the outcomes can increase or decrease depending on multiple variants. To demonstrate this, a first workshop was organized to reveal the importance of the study and clarify the issue. The aim was to simulate the work of a production station, where 11 people were involved, and the roles were distributed as follows: eight operators, each occupying a workstation, a manager, a client and a supplier. First, the workshop was about producing planes and cars. Thus, the industry started with two production lines and it is up to the manager to decide whether or not to produce according to demand, the price of the raw material and the stock to manage. Afterwards, a third production line was added. The orders were available for a limited time in an effort to show a representation that is more realistic. The workshop was therefore a simplified case of a real production line system, but shows real industrial constraints slowing down the transformation. In this case, the training of the operators on the production

lines, the time each operator takes to accomplish his task, and the exchange time are all key indicators in the results obtained.

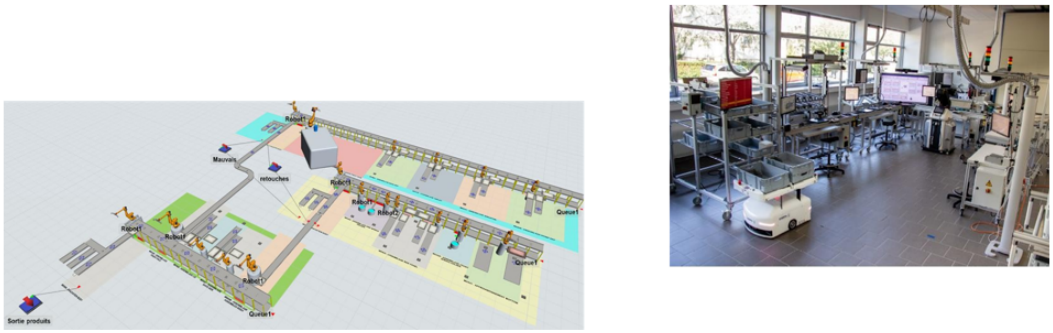

Figure 1.2: Case Study

Thereby, we can realize the importance and the necessity of existing reconfiguration models to handle the forthcoming changes and models to evaluate each configuration, taking into account strong reference standards. In the scope of this work, we will mainly rely on performance aspects to evaluate different suggested configurations for a particular industrial system.

1.9 Problematic

In this perspective, RODIC aims to provide the operators of a factory with a Model-Based Software Engineering (MBSE) framework to reconfigure RMS. Such a framework is constructed around Domain Specific Languages (DSLs) to design configuration models, and also provides methods and languages to manipulate the models and generate production code. In particular, for evaluating performance, the framework must provide a way to simulate the production system by executing the models (using a model interpreter) before the system can be deployed in the real-world. For this task, executable DSLs (xDSLs) are a kind of modelling language that allow the definition and the execution of the behavioural models of the system. The preliminary work of the internship is to realize a state of the art about xDSL and their use for manufacturing systems and RMS.

1.10 Objectives

Evaluating the performance of a simulated system requires defining benchmarks, which is a form of non-functional testing. In such a benchmark, the evaluator must identify and add Key-Performance Indicators to the configuration model. **The first goal of the internship is to consider how to add KPIs to the definition of an xDSL**. In addition, we want to use a standard language for the evaluator to be able to design a benchmark in the same way while considering different systems and their xDSLs. The Test Definition Language (TDL) [TDL] is a standard language, and an interesting candidate for benchmarking. **The second goal of the internship is to extend this existing work with non-functional testing, by taking into account the KPIs defined in the xDSL**. Finally, running a benchmark does

not provide simple verdicts but measurements that need to be stored in models to be analyzed. **The third goal of the internship is to consider how to represent and store the results obtained from performance benchmarks simulations**. The existing Structured Metric Metamodel [SMM] is a good candidate for that purpose.

1.11 RODIC Approach

The approach general of the Rodic project consists of three main components. First component has the object to build an interactive Human-Machine interface, the second component is the central work package that bridges the two other components. It relies on the third component that aims to design configuration modules. In the context of this internship, we contribute to the second work package that has the object to carry out performance evaluation by performing test scenarios.

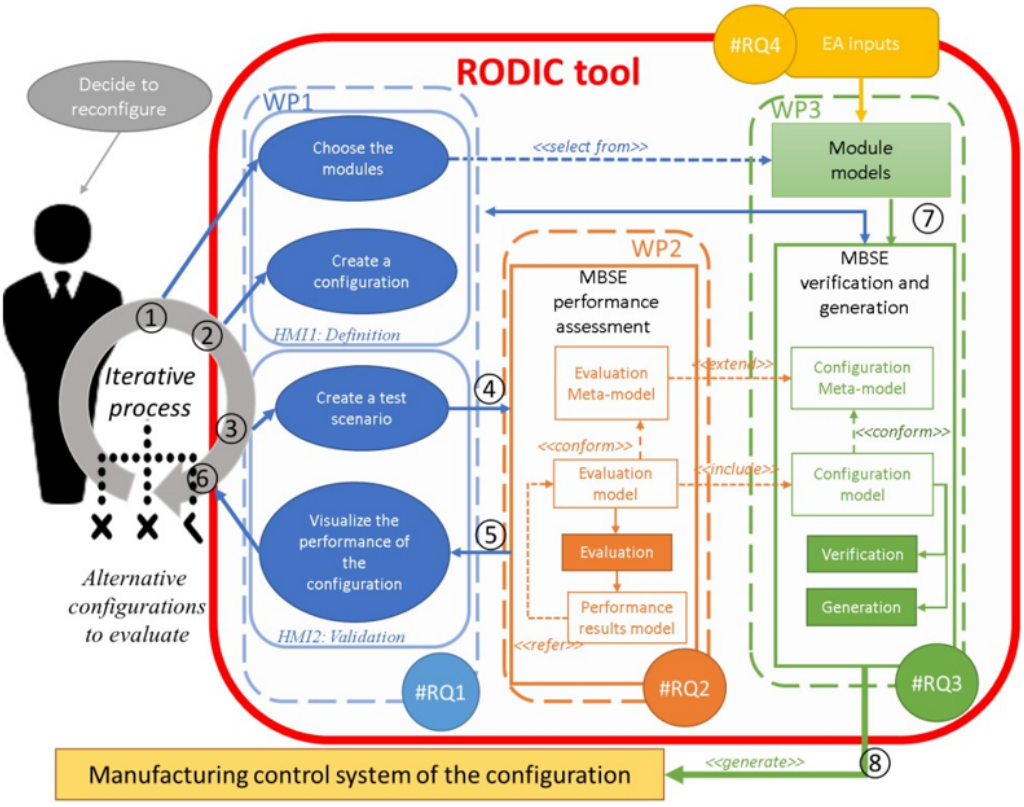

Figure 1.3: RODIC Approach

1.12 RODIC Planning

The RODIC project follows the WBS method to manage the project. Work Breakdown Structure (WBS) is a hierarchical outline of the tasks required to complete a project. The primary purpose of the WBS is to plan the schedule, where each task duration is planned in conjunction with its required predecessors and following tasks. Four work packages involved in the RODIC project, each with a specific function as shown in the figure below.

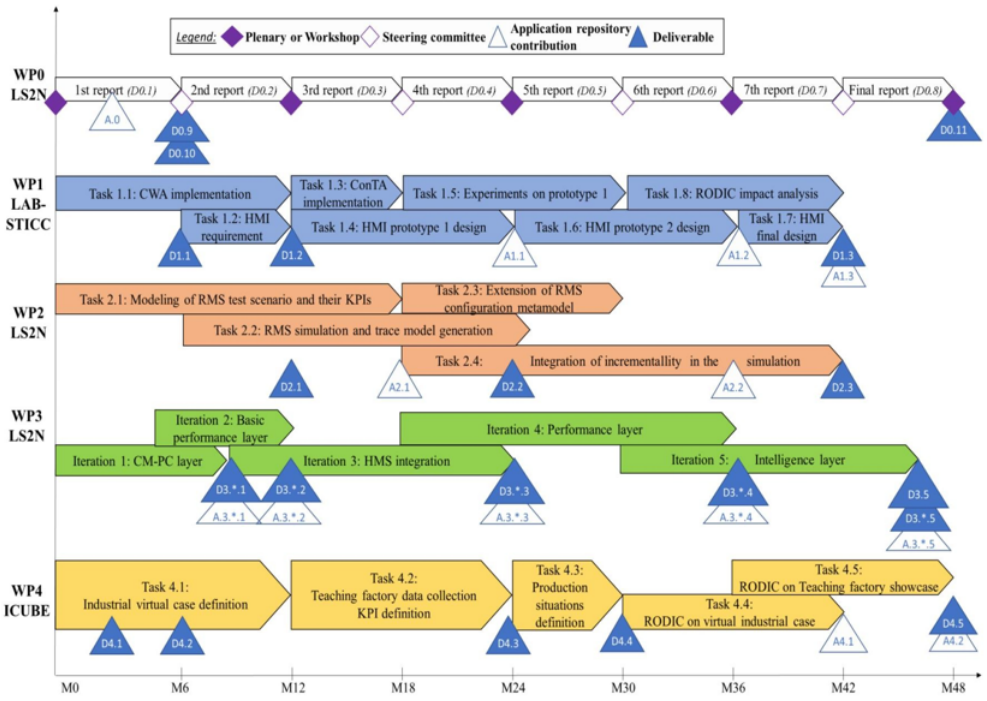

Figure 1.4: RODIC Approach

1.13 Organization and task scheduling

A Gantt chart is a project management tool assisting in the planning and scheduling projects of all sizes, although they are particularly useful for simplifying complex projects.

In this project, the work was organized based on different tasks that can be illustrated in the following Gantt chart.

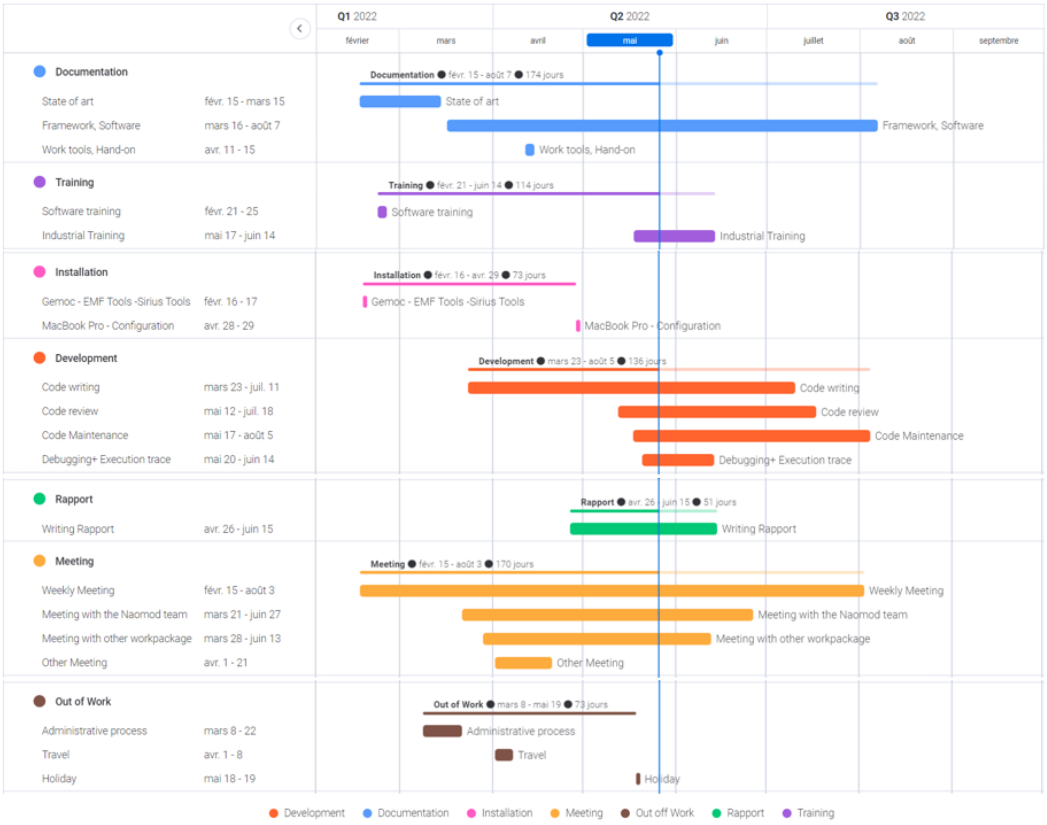

Figure 1.5: Gantt chart

As depicted in the Gantt chart, there are seven crucial groups of tasks that characterized the internship period.

- The documentation started with a state of art task, where I had read the existing scientific works related to the internship subject, and it lasted a month. In addition to software framework and working tools documentation that extends over the entire training period.

- The Training group corresponds to all training that was taking place in the internship period, and it is divided into two types, software training covers all sessions and hand-on programs as Naomod days. And the industrial training (Basics of Flexsim, Handling the Living Packets library).

- The installation concerns all the required tools and the configuration of the environment that we must be equipped with for the development needs, as it also includes the switching between different operating systems, especially from windows to mac and all the updates that had to be managed.

- The development step comprises the design and modeling task, generation and code writing, code review and fixing bugs.

- To make sure that there is no misunderstanding between the different parties involved in this internship, because in fact, we should be dealing with controversial industrial concepts expressed by business experts and we should discuss each concept, what will be the appropriate approach to implement it. All this requires a succession of meetings, besides the weekly meeting to supervise and evaluate the work, and meetings to interact and exchange the results with other related work packages. One of the significant events was the kick-off meeting that took place on March 31, 2022.

- Report group is mentioned in the Gantt diagram to represent the beginning and duration of writing the PFE report.

- Out of work group is related to all tasks that concern the internship but are not included in the work advancement, as the VISA procedure.

1.14 Work methodology

In order to structure the different phases of the project, which will be explained later, and in order to guarantee optimal organization, it is necessary to set a framework that simplifies the kickoff of the project, its progress and subsequently its success.

This internship is part of the RODIC project, started in March 2022 and scheduled to end in February 2026. As known, the beginning of every project confronts several challenges, especially a big project like RODIC, due to new concepts and ambiguity in what we are dealing with, which explains the instability in the work carried out and in the rhythm of meetings. However, we tried as much as possible to constantly maintain the cadence of the work process, which comprises holding regular meetings and accumulating results.

The development methodology is initially defined by the type of systems development life cycle adopted, which in our case is the agile pattern as a process for planning, creating, testing, and deploying software products. This first step is essential because it gives an overview of the progress of the project. So in the first place, we had assemblies with the client, who in our case was the industrial expert. These meetings intended to clarify the requirements and to agree on the concepts and the keywords that will be used during the whole project. This step was the hardest part in the entire internship. It was about understanding what the customer wanted to say and translate it into software language (Meta-models, models, functions...). Afterward, we discuss the approach to adopt and we validate it with the client in order to be able to apply it. From there, we achieve the product backlog in which we plan the next tasks to work on. And when I finish the implementation of the sprint(set of « user stories ») , we hold sprint review meetings to examine what has been accepted, what should be reimplemented and to recognize and remedy bugs. Thus, we had a weekly meeting and quick meeting to ensure the work is on the right track as it is depicted by the figure below. The frequency of the quick meeting depends on the complexity of the current tasks.

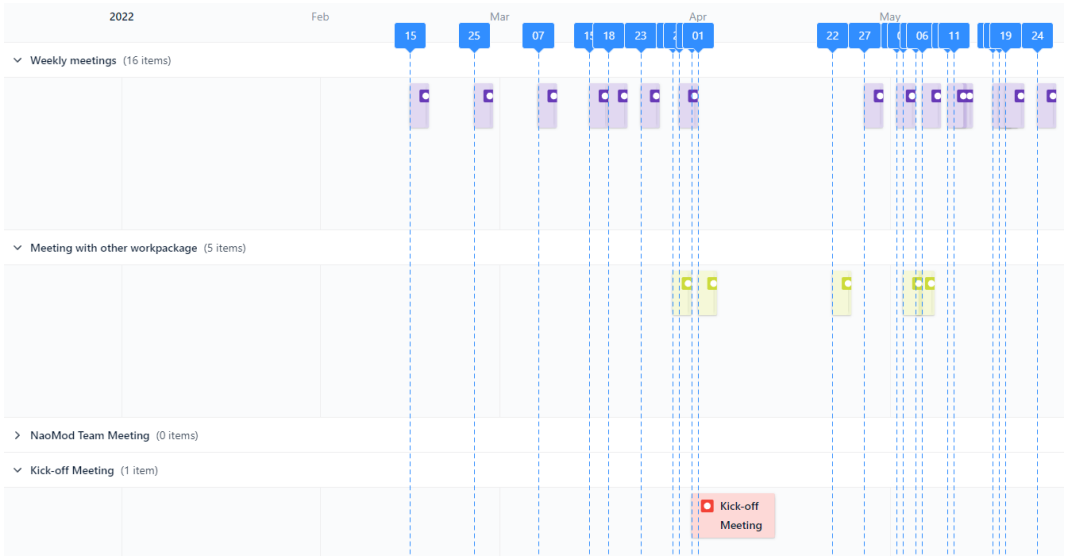

Figure 1.6: Excerpt from TimeLine

Note that the Naomod Team meeting takes place every Monday.

Therefore, to work across multiple teams and organizations, it is necessary to choose strong tools to ensure the flexibility in internal communication and more effective collaborations. This communication toolkit not only empowers the continuity of work to achieve success, but it is also a strategic way to save the traces and evaluate the performance of the job. Here I present the communication tools used during the training.

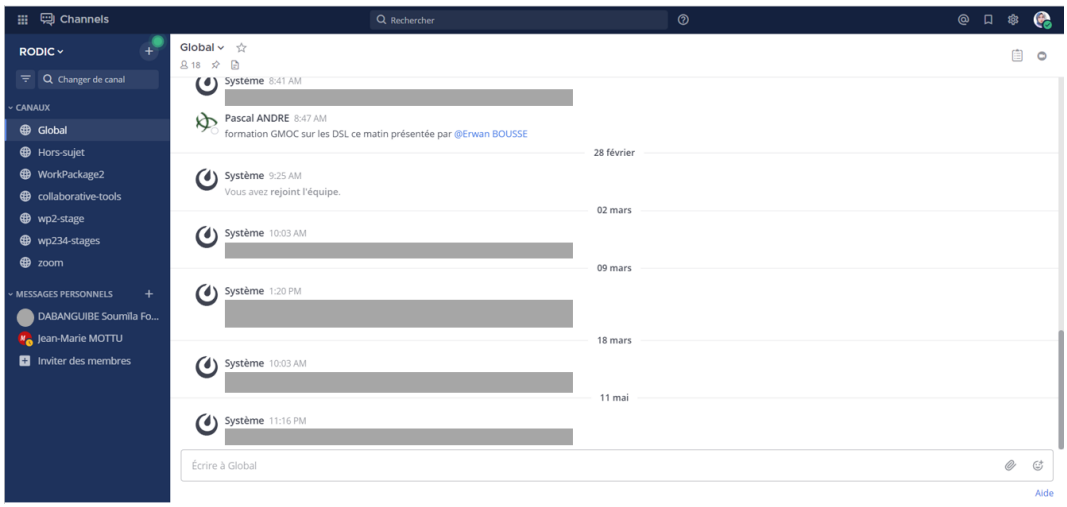

Figure 1.7: Mattermost

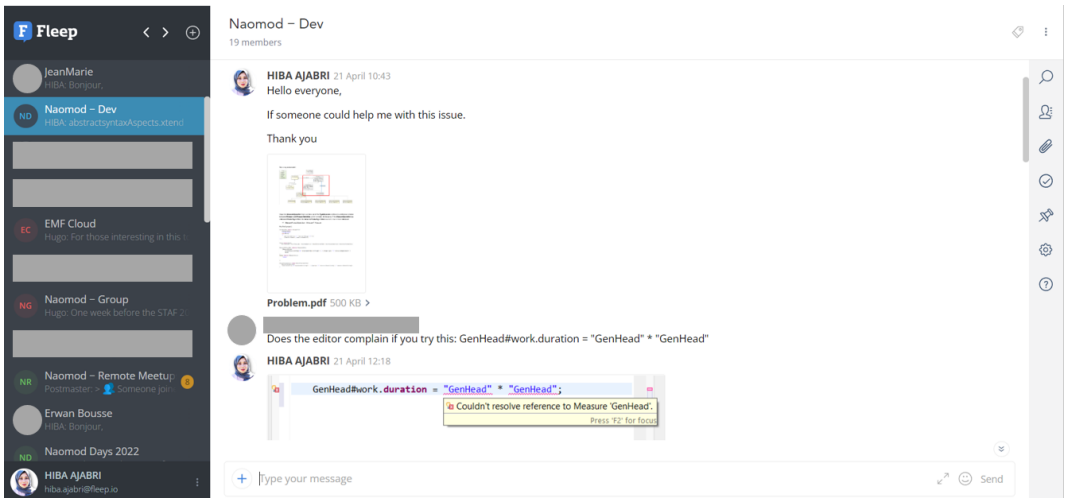

Figure 1.8: Fleep

Mattermost and Fleep were the main tools that facilitate communication between the work packages and with the supervisors. Zoom hosted the remote meetings.

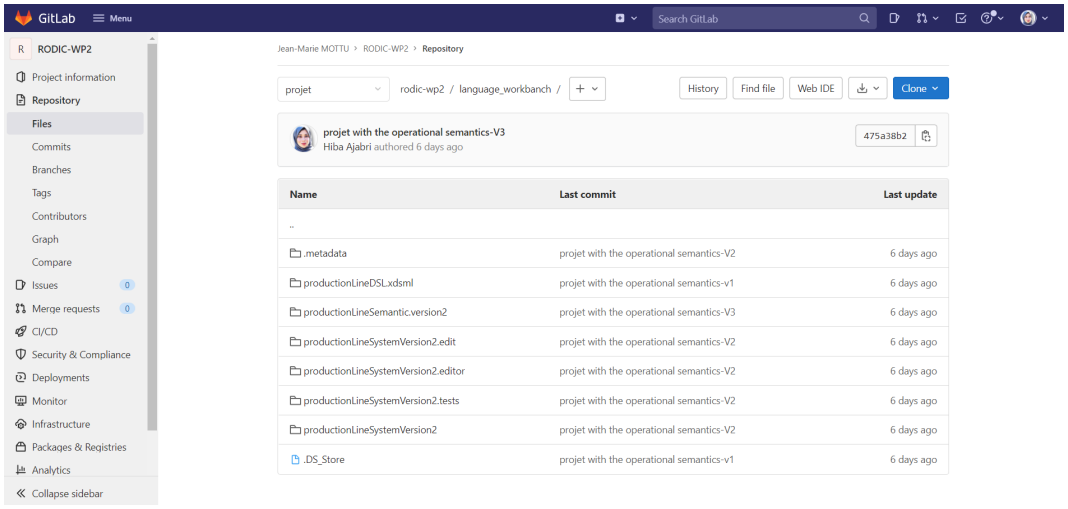

Figure 1.9: GitLab

Gitlab was the platform we rely on to manage working repositories and simplify administration tasks.

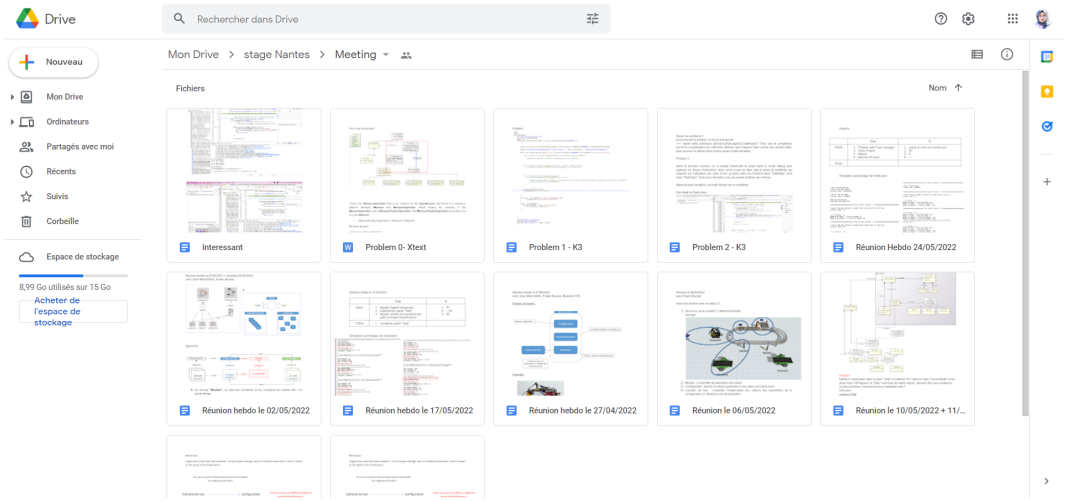

Figure 1.10: Drive

It is imperative to capture the trace of each meeting, as well as it is highly recommended to summarize the output of large discussions. To conduct that, I wrote after each meeting a document that covers the relevant points to take into account to fulfill the next step.

1.15 Life cycle

Every project will go through a development process that can be considered as an iterative process with multiple steps. The systems development life cycle is tailored to perform a rigid structure to lead a successful project, from an initial feasibility study to maintenance of the completed product. Therefore, in this internship, we follow the basic phases.

- **Analysis:** is to evaluate the actual system. This was done by examining the existing software Flexim, that was dedicated to simulate the production lines system of the industrial world.
- **Specification:** is to identify the goals to achieve, and define the corresponding functionalities.
- **Design:** is to develop a functional and software architecture to remedy the problem developed during the specification phase.
- **Development:** is to apply the solution, where new components and programs are installed and composed to fit the specified requirements.
- **Evaluation:** to assess the effectiveness and efficiency of the designed approach, as well as to validate the output with the customer.

1.16 Conclusion

Here we have clarified the context of this internship and highlighted the problematic conduct to open this internship position. We put the issue in the general context and outlined the main objective to attain, and subsequently we gave an overview on the progress of the project by showing the Gantt chart and describing the methodology adopted.

Chapter 2

Scientific and technical background

In this chapter, we'll outline the scientific basic-knowledge to build a consistent view for the forthcoming sections. This software ecosystem will heavily contribute to comprehend the selection approaches.

2.1 Model driven engineering

Models have been used in various fields of engineering to help manage the complexity and represent the information at different levels of abstraction, according to specific notations and requirements expressed by the stakeholders.

Model Driven Engineering [7]as a new paradigm shift from classical programming consists in coding everything to a higher and more abstract level. It is generative engineering characterized by impressive and significant improvements in the development of complex systems, thanks to the code generation of part or all the application through the model. MDE allows transformation into both ways, model to code or code to a generic model, for the reason that when we mount to a very abstract level many implementations become possible. Therefore, this engineering has increased the transparency , interoperability and compatibility between systems, which has a huge involvement in the software development approach since programs are the principal output.

2.1.1 MDE Application

MDE is an approach that holds several impressive techniques applied in software engineering, it gives the principals rules to represent a real system with models, through analyzing the existing situation and suggesting the solution designed by a significant model representation. Not only it allows a static representation but it includes paradigms to annotate the models with dynamic concepts through query languages(Acceleo, OCL...) , and also to write the models semantics (kermeta 3, Melange project...), then we obtain executable models simulating the real system. Moverover, MDE enables multiple operations between models, and to switch from model to another with high qualified tools (ATL). This combination of distinct perspectives that covers the main layers(physic, logic, simulation) enhance the software world and facilitate the “development life” for developers and for business experts as well.

MDE has become widely used technologies by well-known organizations and has diverse application fields such as the low code - no code, one of the most popular technologies that reveals the future of enterprises and crushes business complexity. The low code platforms rely on code generation, the basic features of model-driven engineering approaches.

RODIC project is among the MDE use cases, that is attempting to provide the operator of a factory with a simulation of the production chain in case we change its configuration before the system can be deployed in the physical world to reduce time and cost. The simulation will basically rely on existing reconfiguration models and (x)DSL for the execution.

2.1.2 Scientific context

2.1.2.1 Model & Meta-Model

The Object Management Group (OMG) is an international group developing technology standards that provide real-world value for thousands of vertical industries. OMG standardizes the model object, so before diving into the OMG's modeling standards, it is quite important to discern the underlying concept behind the "Model" word. The scientific consensus is oriented to define the term "Model" as a representation of reality. All theories and norms subsequently aim to improve the model design to represent the real system in the best possible way. Thereby, a new concept has emerged and has been entangled with the model term, which is the meta-model.

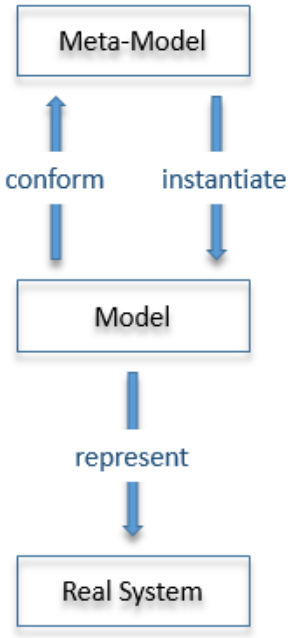

Figure 2.1: Model & Meta-Model

Meta-Model is a high level of abstraction where we can define the idea to be represented, and then we can design models that conform to it, in which we instantiate the idea. It is important to note that a meta-model can also successively have a superMeta-model, the level of granularity is being defined according to the complexity level of the system under study.

2.1.2.2 Modeling approaches

OMG came with different modeling approaches, to organize the development process.

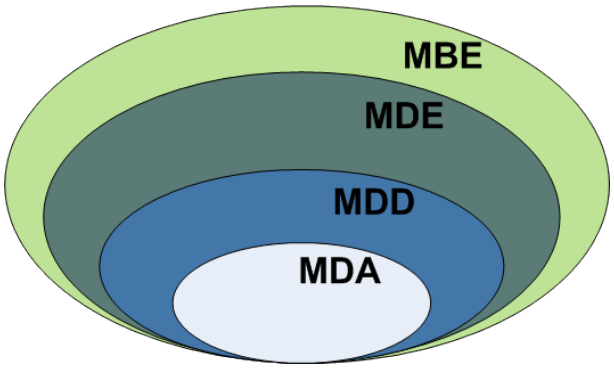

Figure 2.2: Modeling approaches

First, the Model-based Engineering (MBE) that relies on models to provide software programs but these models don't contribute to the code implementation. Second, Model Driven Engineering (MDE) is narrower in scope than MBE, where the models drive the development. Third, the Model Driven Architecture (MDA) [11] standard is MDE incarnation ,yet it contains many OMG standards, and its implementation technologies are frequently used by enterprises.

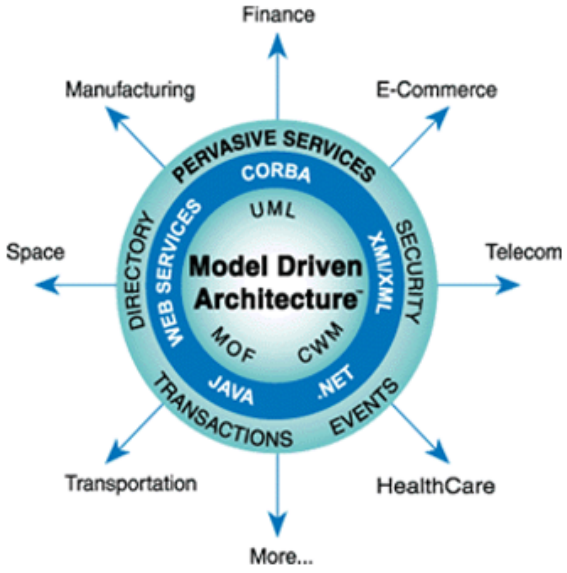

Figure 2.3: MDA Diagram

2.1.2.3 Reverse Engineering

Model transformation is an important step in the MDA process, thus MDA consists of three stages : the CIM, PIM and PSM. We thereby start by modeling a business domain that helps to understand the problem domain and make it available to all persons involved, and we move to PIM through transformation model to model, and then PSM that is a result of a model transformation and contains all the details necessary to generate code. With this logic chain we are in forward engineering, otherwise we can from the code generate the source model what is called reverse engineering[3].

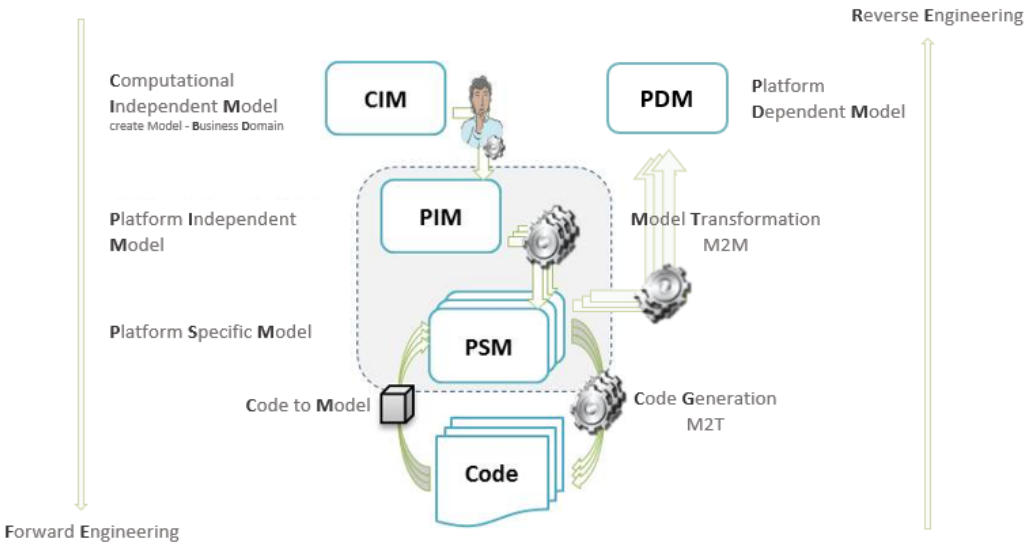

Figure 2.4: Reverse Engineering

2.2 WorkSpace

This internship was based on software tools to apply the approaches and test its realization.

2.2.1 Gemoc Studio Environment

Figure 2.5: Gemoc Studio

The Eclipse GEMOC Studio provides generic components through Eclipse technologies for the development, integration, and use of heterogeneous executable modeling languages. This includes, among others.

- Metaprogramming approaches and associated execution engines to design and execute the behavioral semantics of executable modeling languages.
- Efficient and domain-specific execution trace management services.
- Model animation services.
- Advanced debugging facilities such as forward and backward debugging (i.e. omniscient debugging), timeline, etc.
- Coordination facilities to support concurrent and coordinated execution of heterogeneous models.
- An extensible framework for easily adding new execution engines and runtime services.

The studio is also provided as an Eclipse product, which eases the diffusion of the various technologies aforementioned through an integrated studio that also includes documentation, examples, and tutorials.

The Eclipse GEMOC Studio offers two workbenches (accessible with specific perspectives).

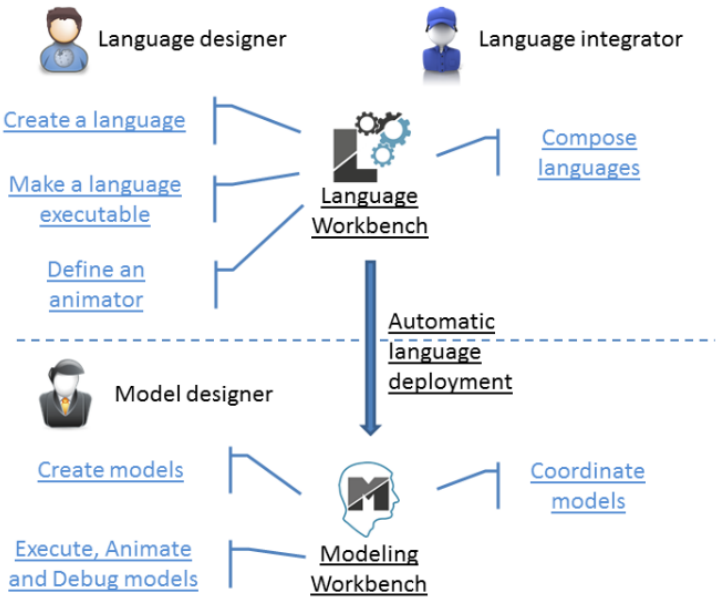

Figure 2.6: Gemoc perspectives

- A Language Workbench to be used by language designers to build and compose new executable modeling languages.
- A Modeling Workbench to be used by domain designers to create, execute, and coordinate models conforming to executable modeling languages.

2.2.2 Eclipse Modeling Framework

Figure 2.7: Eclipse Modeling Framework

The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor.

2.2.3 Kermeta 3 - Executable Meta-Modeling

Figure 2.8: Kermeta 3

Kermeta 3 (K3) is an action language built on top of the Xtend programming language and mainly used to implement the execution semantics of Ecore metamodels. Concretely, K3 allows to "re-open" the classes generated from an Ecore metamodel using simple annotations in order to weave new features and operations.

Main features of K3 include.

- Executable metamodeling: Using K3, one can insert new methods in existing Ecore meta-classes, with their implementation. These methods define the execution semantics of the corresponding metamodel in the form of an interpreter.
- Metamodel extension: The very same mechanism can be used to extend existing Ecore metamodels and insert new features in a non-intrusive way.
- Full Java compatibility: K3 files are plain Xtend files. As such, K3 files are ultimately compiled as plain Java code. This means that Java code and API can be used in K3 files and vice versa.

2.2.4 Sirius - Graphical modeling

Figure 2.9: Sirius

Sirius is an Eclipse project that allows users to easily create graphical modeling workbench by leveraging the Eclipse Modeling technologies, including EMF and GMF. Sirius has been created by Obeo and Thales to provide a generic workbench for model-based architecture engineering that could be easily tailored to fit specific needs.

Based on a viewpoint approach, Sirius makes it possible to equip teams who have to deal with complex architectures on specific domains.

Figure 2.10: Xtext

2.2.5 Xtext

Xtext is a framework for development of programming languages and domain-specific languages. It allows users to define a language using a powerful grammar language. As a result it gives a full infrastructure, including parser, linker, typechecker, compiler as well as editing support for Eclipse.

2.2.6 Acceleo

Figure 2.11: Acceleo

The Acceleo Query Language (AQL) is a language used to navigate and query an EMF model. AQL as a query engine is small, simple, fast and extensible and it brings a richer validation.

2.2.7 Flexim

Figure 2.12: Flexim

Flexim is a software tool that enables the operators of a factory and industrial companies to simulate their systems in a very accurate way, with a 3D flow. And it also gives analysis and statistics for management targets. In our work, Flexim was mainly useful to understand the business domain and to analyze real use cases.

2.3 Conclusion

In the above section, we have illustrated the scientific background where we have highlighted the MDE approach proposed by the OMG, and we have presented the technical tools used to carry out the work. The next step will be devoted to presenting the state of the art of the internship subject.

Chapter 3

State of art

As is well known, each doctoral study program relies on a scientific strategy, and one of the essential steps is to establish the state of art, which describes the current knowledge about the study through the analysis of similar or related published work. Thus, this part introduces the documentation section for the internship. It covers a presentation of basic concepts in the scientific literature, as a scientific background of the subject, in particular, it will present the work of two theses and an article, and how they are linked to our work.

3.1 Doctoral Thesis of Erica CAPAWA FOTSOH

« Contribution à la reconfiguration des lignes de production : définition et démarche de choix de configurations alternatives »

This thesis introduces the basic industrial definitions, in particular, it identifies the crucial assumptions on which the RMS are based, and it presents the suitables steps to drive an effective configuration choice.

3.1.1 Definitions

3.1.1.1 Reconfiguration Definition

«Reconfigurability is defined by [10] as being the capacity of the system to dispose or replace system modules in order to respond to changes in the market.»

«[6], [13] and [9] consider reconfigurability as the ability to add, remove or rearrange elements of the production system in a cost-effective manner.»

From these two relevant definitions, we conclude that a system is a collection of elements that are organized in a specific way (Configuration), and any change of this element's organization is considered as a reconfiguration as long as the system keeps performing its work.

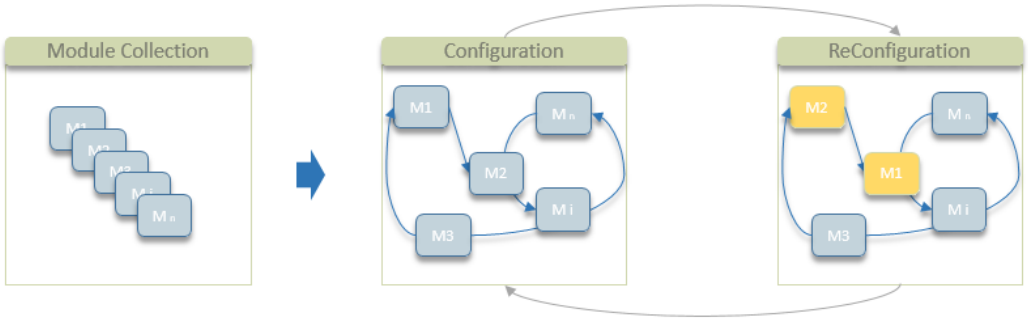

Figure 3.1: Modular pproach

So the change leads to a reconfiguration, once it's uploaded, it is considered as a new configuration. Thus, we can observe the reconfiguration cycle. The figure illustrates clearly the strong relationship between ***Configuration*** and ***ReConfiguration***, thus providing instant reconfiguration when needed, refers to formalizing the configuration.

3.1.1.2 History of RMS

In the beginning, the industry aimed to produce as much as possible in less time. In this context, the Dedicated manufacturing systemes (DMS) emerged, they are designed to produce big quantities around a few families of products. Through the years, the type of demand has changed, the industry moved from the manufacture of unique products to the product customization, this is when the Flexibles manufacturing systems (FSM) appeared. Nevertheless, the industry couldn't pursue the cadence of development and the raise in the demande with the aforementioned solutions. On the one hand, the DMS are fixed at one type of product, on the other hand the FMS are too expensive. It is when the Reconfigurable Manufacturing Systems (RMS) shows up.

3.1.1.3 Definition of RMS

Reconfigurable Manufacturing systems (RMS) are defined as systems whose physical and logical structures at all system composition levels can be changed quickly and inexpensively in order to harmonize production capacity with sudden changes in the market [6].

A RMS system must know where and how to act when a change occurs?

➢ Determine which modules to (modify, add, remove) to improve system performance. (principle of modularity)

➢ Determine how (modify, add, remove) the modules while keeping the consistency of the system. (principle of integrability)

3.1.2 The characteristics of RMS

The thesis defines the fundamental characteristics of RMS. In this document, we will focus on five of them.

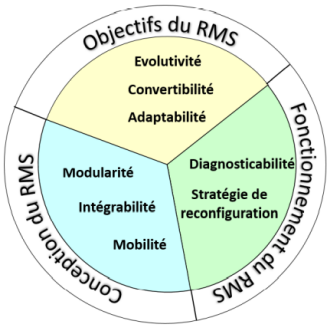

Figure 3.2: Classification of RMS characteristics

- Modularity

Modularity [4]is the central concept. It is implicitly embedded in the definition of ***Reconfiguration***. This hypothesis perceives the whole system as a composition of modules combined to allow it to function. The definition of modules, in turn, should be very accurate. There are four essential aspects to consider while defining a module, which are the physic aspect, control logic, performance (KPI) and model aspect. The definition depends on the level of granularity chosen, but it should also take into consideration a stable reference.

```
.....
22 Package A {
23    Class A1 {
24              Element a {...};
25              Element a' {...};
26    };
27    Class A2 {
29              Element a {...};
30              Element a' {...};
31    };
32  };
.....
```

```
.....
22 Package B {
23    Class B1 {
24              Element b {...};
25              Element b' {...};
26    };
27    Class B2 {
29              Element b {...};
30              Element b' {...};
31    };
32  };
.....
```

Figure 3.3: Example

In the above example, we have 8 elements, 4 classes and 2 packages, so finally we have 14 units. If we consider all the levels of granularity, the cardinality of the combination could be estimated to $\boldsymbol{C}$.

$$C = 2^{14} - 1$$

There are multiple possible combinations between modules, but not all of them are possible, and therefore defining the mapping relies on defining the reconfiguration framework, which is correlated to the notion of integrity.

- Integrability is the ability to create interfaces between modules.
- Convertibility is the ability to convert to another system to respond to changes.
- Customization is the ability to satisfy end-users, this depends on the diversity of the family of products.
- The diagnostic is an analysis that captures all the states of the system during its execution, to follow the operation of the system, reveal the causes of malfunctions, and make decisions.

3.1.3 Approach of the configuration choice

3.1.3.1 DataBase

Idea is to build a database consistent to be able to store existing configurations, and to capitalize on the knowledge of the company, hence feeding the database with new configurations, which may not yet be applied but have proven their consistency and relevance through their simulation models. Thus, the company can, at any time, find the appropriate configuration for the current situation.

3.1.3.2 Performance

Performance evaluation is an important step in the reconfiguration process, it goes beyond the technical and scientific studies that can be performed to properly configure the system. Because the question is, why choose this configuration and not the other configuration? Performance serves as a key metric to answer this question. Therefore, the reconfiguration process is closely related to performance analysis.

Modularity being a fundamental concept of reconfiguration, the performance must also follow a modular approach. So each level of granularity requires a definition of performance. Therefore, they are two categories of indicators to reveal: advanced indicators and retarded indicators.

Retarded indicators

- Capture system results.
- Example: annual production volume.

Advanced indicators

- Offer information on the causes that led to certain performance values.
- Measured over a short period.
- Example: the production volume of the system per day

Thus, the decision of the choice of reconfiguration is based on the retarded indicators, and the understanding of the causes that led to these indicators, rely on the analysis of the advanced indicators.

Indicators that can be beneficial to assess the system performance.

- Quality
- Reliability (ability of the reconfigured system to respond in the exceptional cases)
- Uncertainty and the influence of the environment
- The energy consumption

- Number of products produced by the system (over a given duration)
- The system utilization rate
- Machine utilization rate

3.1.3.3 Approach

The thesis [4] proposes the following approach to conduct a configuration choice for a specific industry situation.

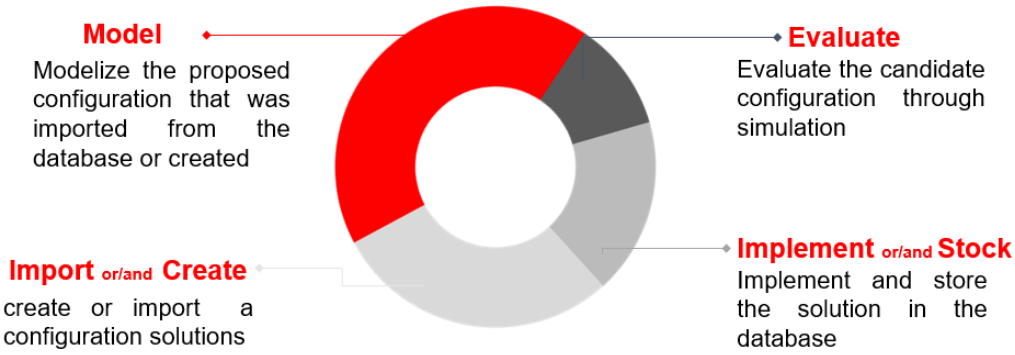

Figure 3.4: Configuration approach

At first, we start by creating or importing existing configurations, and then we try to adapt it to the current situation through a modeling phase. This allows us to simulate the work of the system with specific parameters, in order to perform an evaluation step. When the configuration fits all required specifications, it will be implemented and saved in the database to be re-imported when needed.

3.2 Doctoral Thesis of Beziers La Fosse

« Model-driven methods for dynamic analysis applied to energy-aware software engineering »

This thesis [2] aims to provide instant feedback to the developers about the energy consumption of their applications, in order to help them improve the energy efficiency of their systems. To do so, the thesis proposes an approach to annotate the xDSL of the systems under development with the energy-consumption concepts. This is close to the objective of our work, for the reason that, in our case the xDSL will be annotated with performance-based concepts instead of the energy concepts to be able to compute the different KPI based on the execution traces.

3.2.1 Required Concepts

3.2.1.1 Model transformations

Model transformation [12] can be endogenous, or exogenous. The endogenous structure consists in transforming models that have the same meta-model. When the exogenous model transformation concerns the transformation from one model to another and they don't have the same meta-mode. For instance, code generation is an exogenous transformation, from the fact that it transforms a model, which has its meta-model to the second model that is the language (generally its java language) that has its own meta-model.

3.2.1.2 Executable meta-modeling

The executable meta-models [5] are models that can be executed. They require a DSL that provides the execution semantics, which consist of rules attached to the meta-classes of the the meta-model «abstract syntax». Typically, there are two categories of them: Operational semantics, Translational Semantics.

1. Operational semantics

   Operational semantics are defined as endogenous model transformations that changes the state of the model during its execution.

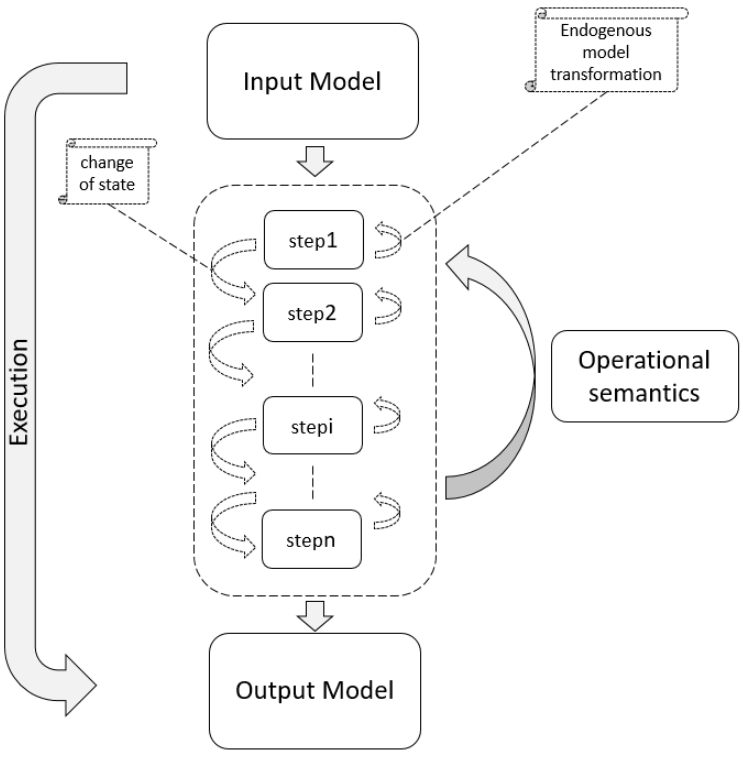

Figure 3.5: Operational semantics diagram

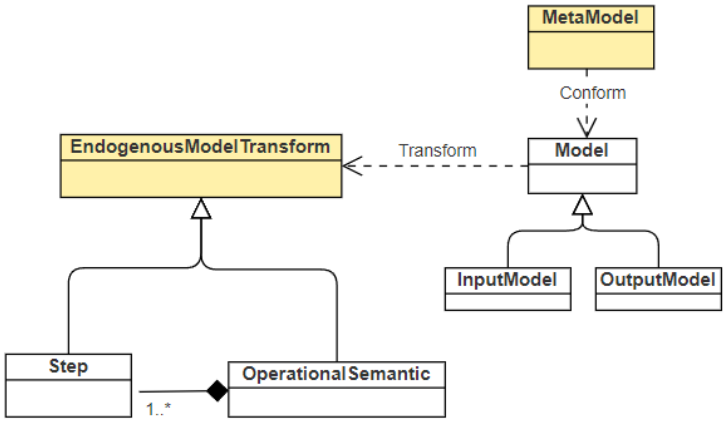

Figure 3.6: Operational semantics pattern

Operation semantic transformation with ArduinoML Language in Thibault Beziers La Fosse thesis.

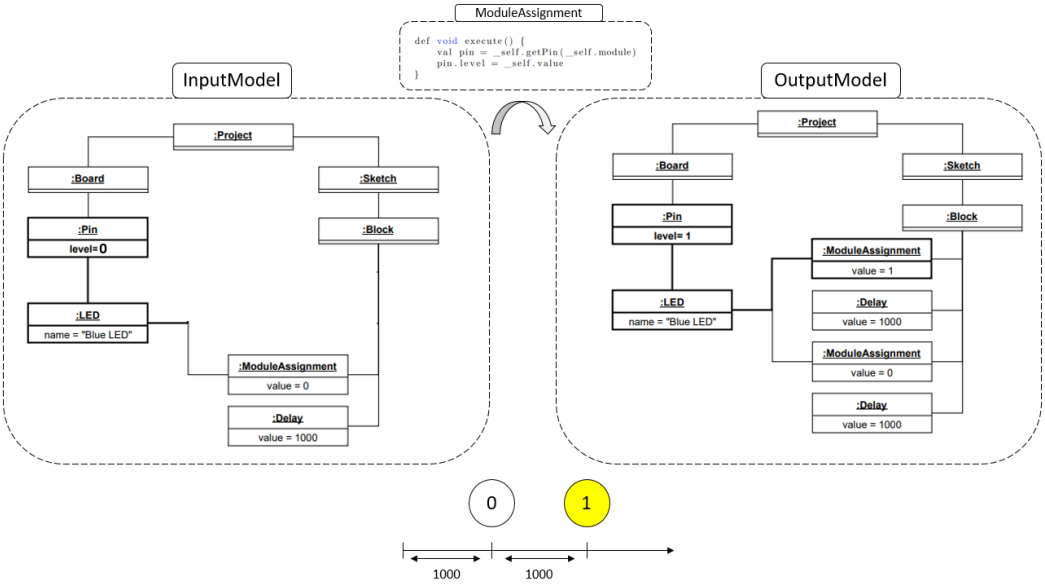

Figure 3.7: Operational semantics example

2. Translational Semantics

   Translational semantics consist in an exogenous model transformation, that translates the model into a second one. This second model conforms to the abstract syntax of a second xDSL.

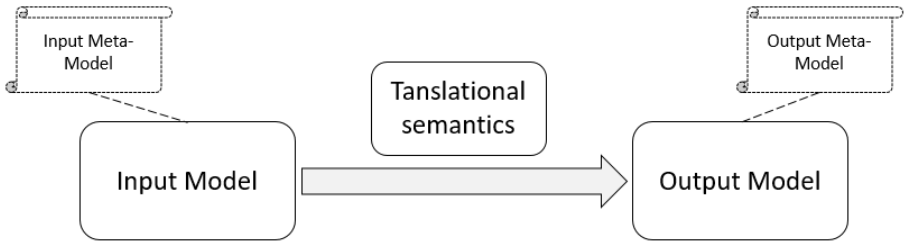

Figure 3.8: Translational semantics diagram

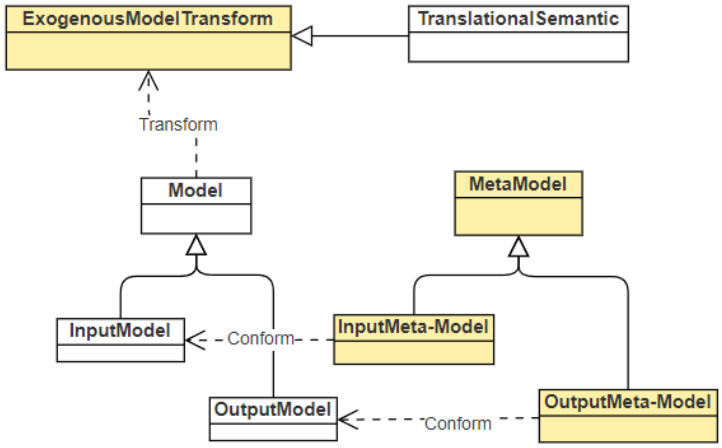

Figure 3.9: Translational semantics example

3.2.1.3 Instrumentation

Software instrumentation [14] tracks the different states that have been recorded when running the program. There are two instrumentation methods : ***Static*** (source-code) and ***Dynamic*** (binary and on-the-fly).

Source code instrumentation consists in adding a line code in the source code. While the binary instrumentation approach consists in adding a compiled code in the compiled source code which means there is no need to recompile the source code, and that reduces the complexity. Otherwise, in the source code approach, the new lines will also be interpreted, compiled, executed and verified but it is also an accurate approach.

3.2.1.4 Execution traces

The execution traces record all the states of the program execution, all parameters and values that were set in a step or other, hence they provide a dynamic analysis while executing a "Model". So execution traces are the result of instrumentation methods.

3.2.1.5 Impact analysis

Impact analysis focus on identifying the lines in the code which are considered by a change.These will be heavily beneficial, inter alias, in the regression test selection, in order to reduce the duration of regression testing phases, where instead running the entire test, only the newly code changes will be tested.

3.2.2 Approaches

The thesis proposes the following approach to annotate the meta-models by energetic measurements.

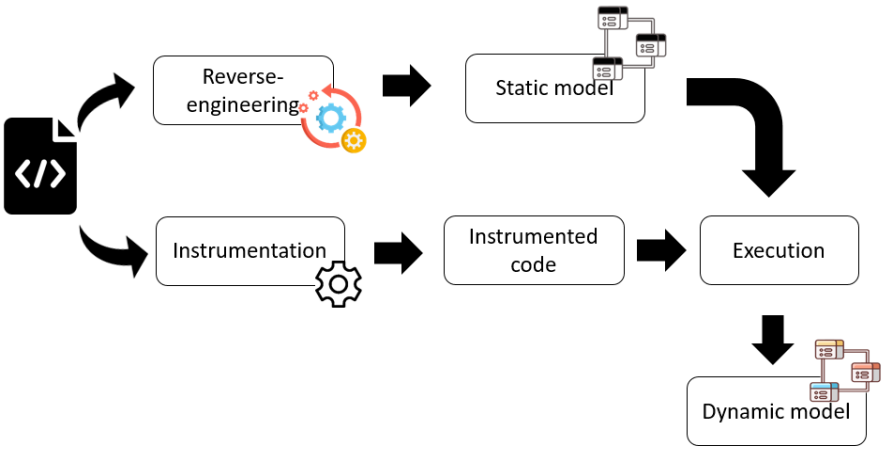

Figure 3.10: Approach

Approach of the thesis relies on three fundamental steps. (1) Generate the static model from the source code, (2) instrument the source code and (3) inject the energy measurements into the generated model.

The third step requires a prior phase to calculate the measurements, this proposal leads to another approach, which is Energy Estimation Language (EEL). This language was designed for annotating any given xDSL with energy-estimation formulas. Therefore, for each runtime platform, an energy specialist writes an Energy Estimation Models (EEM).

3.3 Article of Faezeh Khorram

« Adapting TDL to Provide Testing Support for Executable DSLs »

Software testing is thoroughly intertwined with software development, and it is unimaginable that an application can be deployed without undergoing a testing phase. Moreover, testing is more important in the software industry to maintain the product quality and gain customer satisfaction. Faezeh's article [8] came to answer this question and allow running testing processes on any xDSL. This article will be related to our work on the fact that the second objective of the subject is to perform test cases on the model under execution, by taking into account the KPIs defined in the xDSL.

3.3.1 Objective

Domain Specific Language (DSL) describes the behavior of models conforming to it. Thus, for each new DSL, we will have a new different modeling environment. The objective is to provide testing facilities for any given xDSL, which means the development environment must be equipped with verification and validation tools specific to an xDSL, to help the domain expert get previous results and estimations gathered from the execution of model. For this reason, the article proposes a test framework that includes (1) Way to write test cases and (2) Way to execute tests written, for each new xDSL.

3.3.2 Challenges

- Create a testing language that allows the domain expert to write test cases, and it must define the domain concepts that determine how a model can be tested.
- Connect the execution semantics of xDSL with that of the testing language, so that the language will be able to demand the execution of models when needed.

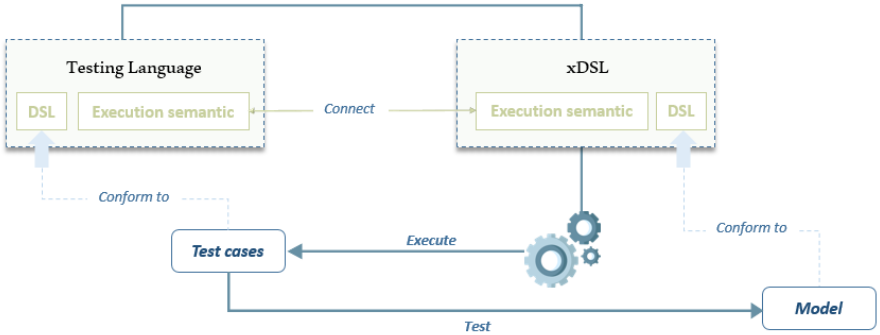

Figure 3.11: Illustrative diagram

- The testing language must perform a dynamic analysis of the state of the model under test and check whether or not the state conforms to the expected one.

To address the above-mentioned challenges, the article is based on an existing test language TDL[1], and it will adapt it to meet all the requirements raised.

3.3.3 Background

The e**X**ecution **D**omain **S**pecific **L**anguage is basically composed of two main parts: abstract syntax and the execution semantics; this article focuses more on the operational semantics. The abstract syntax designs the meta-model of the model considered, whereas the operational semantics (interpreter) specifies the dynamic aspect held by the model during its execution, so it includes the possible runtime states of the model under execution and the execution rules to determine the model behavior over time.

For a given xDSL, the possible runtime states are the features able to be modified by the execution rules during executing the model (dynamic features).

3.3.4 Approach

The approach offers two basic components:

1. TDL Library Generator : this components aims to generate a library specific to a given xDSL, which can provide (1) all data types to write test data, (2) default test configuration, (3) and element enabling TDL test case to request the execution of the MUT (Model under Test), and (4) element to write OCL queries in TDL test cases.

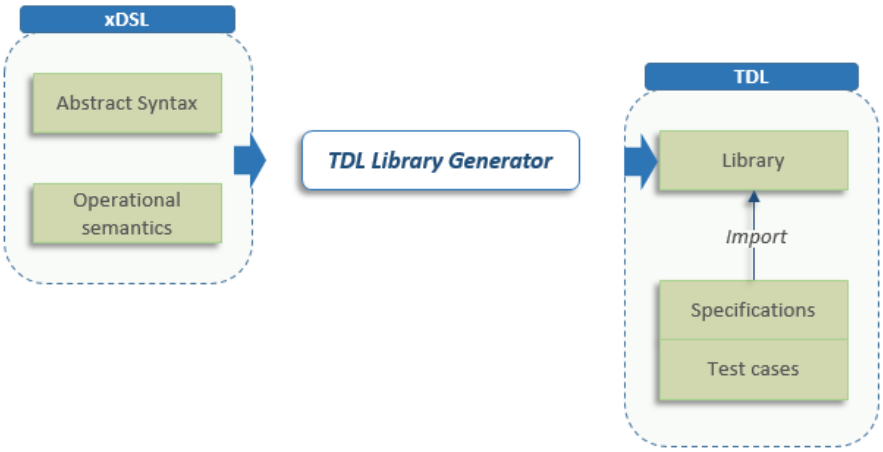

Figure 3.12: TDL Library

2. TDL Interpreter : is based on the operational semantics, which is connected to two other components :
   - Execution Engine: is able to load model, load xDSL, and execute the model. This engine is used by the TDL Interpreter to start the execution of the model.

- Query Evaluator: evaluate the OCL expression in the test cases.

The approach followed to execute test cases on a model uses the operational semantics of the concern xDSL, which can be defined using different metaprogramming approaches, which in turn has a specific execution engine.

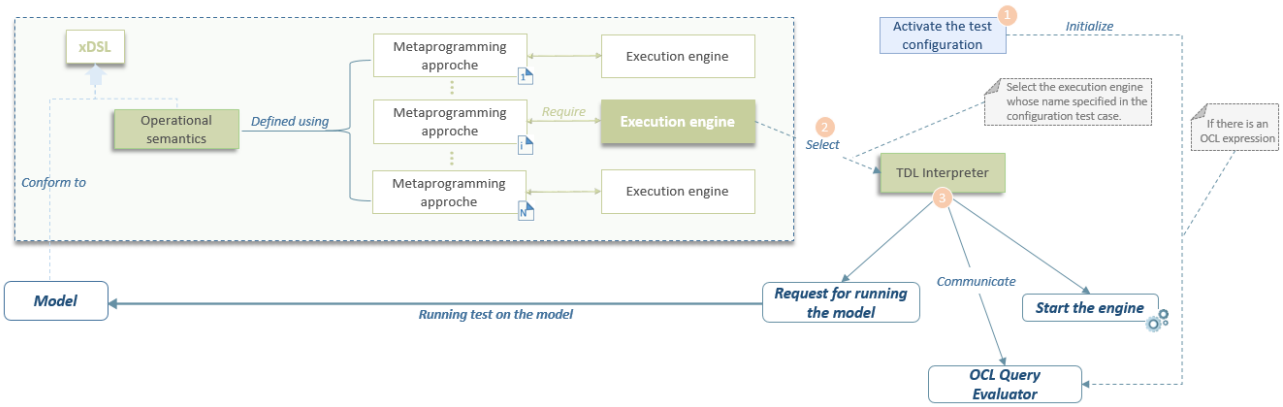

Figure 3.13: TDL approach

It starts by activating the configuration which contains the appropriate DSLName, so that the TDL interpreter selects the corresponding engine, thereby it can ask for running the test on the given model after having started the engine.

3.4 Conclusion

During this chapter we present a holistic view of the state of art, where we expose the work of two thesis and Faezeh's article. The object of the first cited thesis is to give an infrastructure about the basic concepts that will be used during the RODIC project and an effective methodology to reconfigure the RMS. Then we had seen the work of the second thesis that relies on annotating a given xDSL by the energetic-consumption concepts, which will benefit in terms of annotating the xDSL with performance concepts. And Thanks to Faezeh's work, we can run test cases on the chosen configuration model.

Chapter 4

Functional study and analysis

This chapter will be dedicated to disclose the needs through a whole functional study. It will underline different steps followed to analyze the current system and understand the project expectations. The study will count on a simplified case to demonstrate various concepts and functionalities.

4.1 Configuration vs Test Scenario

As stated before, the goals of this internship is to provide the operator with different possible configurations that meet the current situation. For this reason, the suggested configurations will be evaluated with different test scenarios. It is therefore necessary to specify the definition scope for each of the two concepts, and what would be the points of intersection and the points of divergence.

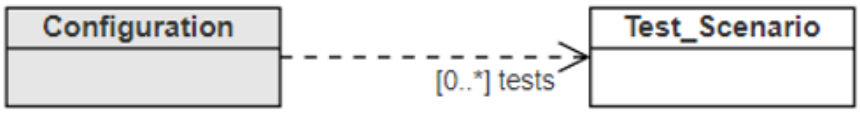

Figure 4.1: Configuration vs Test Scenario

The configuration is linked to a set of test scenarios, where the reconfiguration could be an addition, deletion or modification of a module(conveyor, operator..), and the scenario depends on a specific configuration (changing the location of a module, changing the task execution order for a module . . .).

4.1.1 Example

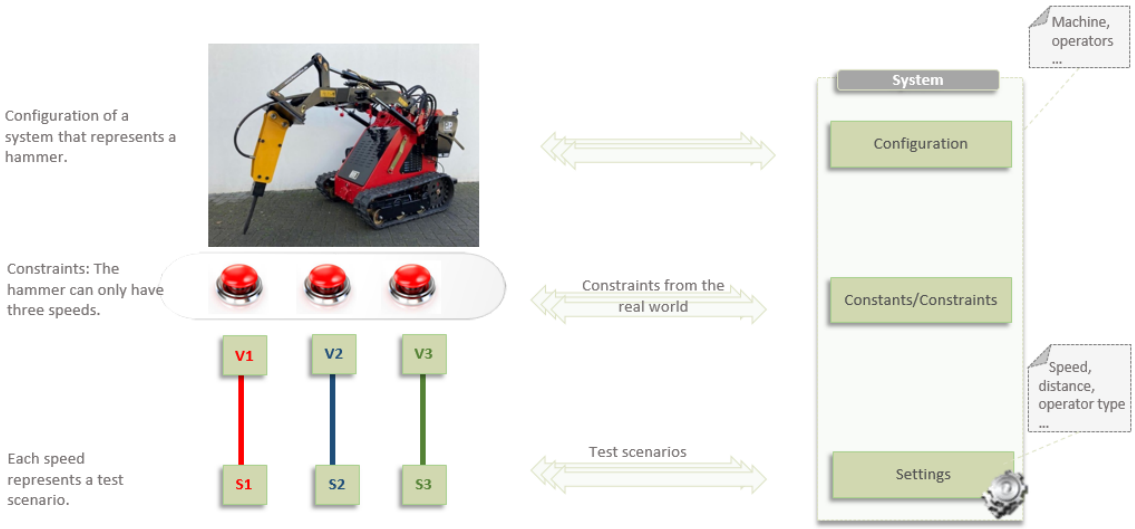

Figure 4.2: Example

Here we consider a hammer to represent our system which is configured to have no more than three speeds. Thus, the evaluation of the system could be performed based on the specific input which here is the speed value.

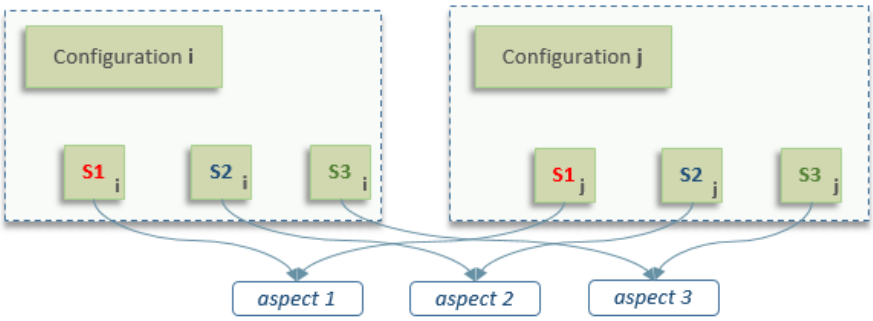

Figure 4.3: Configuration & Test Scenario

To make a choice between different possible configurations, it is necessary to compare them. For this reason, the meta-model of reconfiguration must be annotated with performance concepts. The idea is that for each configuration, we will provide multiple test scenarios, each of them assesses the configuration based on a specific aspect, and at the end we can compare the different values obtained for each aspect.

4.2 Simplified case description

The physical system considered here is imaginary, and was never actually deployed in a workshop. Here is the description from which the study was conducted.

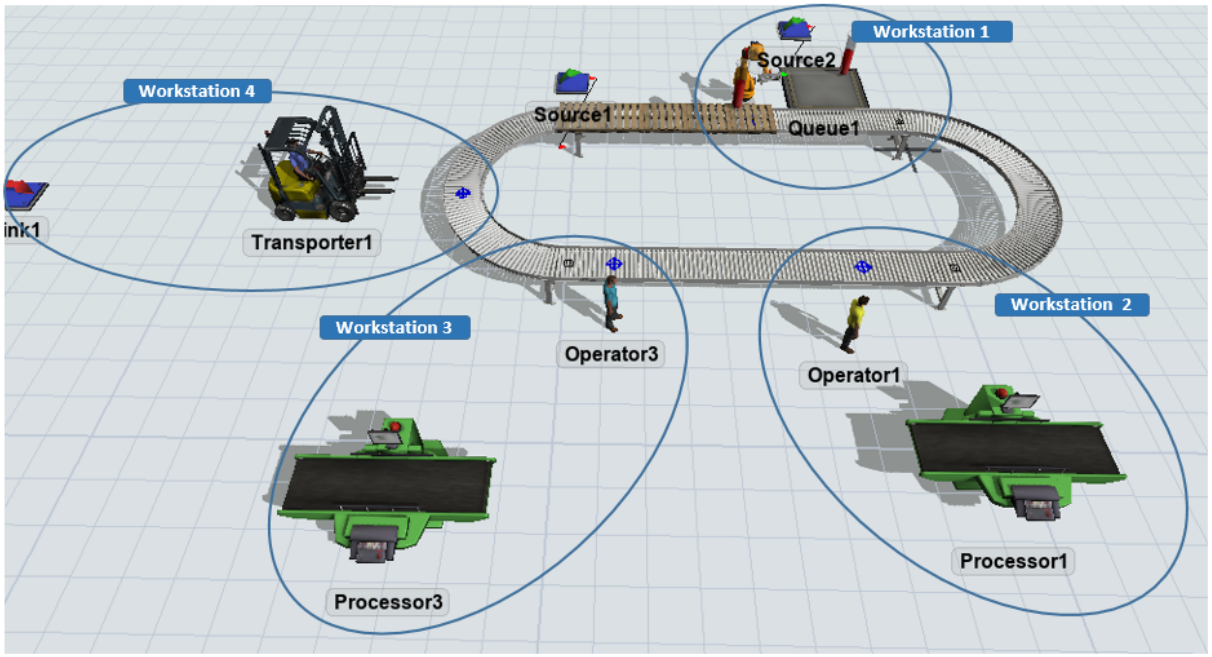

Figure 4.4: Simplified case

The system mainly consists of a loop of conveyors on which the pallets move. The pallets are always on the conveyor, and advance at a constant speed of 1m/s. Blockers are installed in front of the work sites to allow production operations to

take place in good mechanical conditions.

The first workstation considered is equipped with a poly-articulated robot(6-axis). This robot takes the products stocked in Queue1 when a pallet arrives at the station. Picking up a product requires 5 seconds, placing it on the pallet also requires 5 seconds. The products arrive in stock one by one, separated by an inter-arrival time that is normally distributed with a mean $\mu = 14.5$s and standard deviation $\sigma = 3$s. The robot places 2 products on the pallet to let it move forward.

At the second workstation, located 15m50 after the first workstation, an operator takes a product, then places it on a processing machine Processor1 located 5m from the conveyor. Once the operator picks up the product, the pallet moves toward the next station. The operator processes the product on the machine, then puts it back on the pallet, which has therefore moved forward in the meantime.

At the third workstation, located 6m after the second workstation, a second operator takes the second product to process it on Processor2 located 5.6m from the conveyor. When the pallet has received the 2 products, it moves to the next post.

At the fourth workstation, located 5m10 after the third workstation, Transporter1 takes the two products one by one, and puts them on the final storage. The whole process takes 18 seconds (full pallet to empty pallet exit). When the pallet is empty, it is released and returned to the first station 10m away from the fourth station.

4.2.1 Module Definition

As seen before, modularity is our main theory that holistically conducts the approach, so the question is what would a module be in a concrete production line system?

After a long discussion between the expert of the industry and the software teams that includes specially the workpackage2 and the workpackage3, we initially agreed that a module could represent a workstation in the considered case (simplified case).

4.2.2 Synchronization cases

Based on the simplified case, the workstation 2 & 3 can be perceived as a single station that will be considered as a critical workstation in the synchronization study.

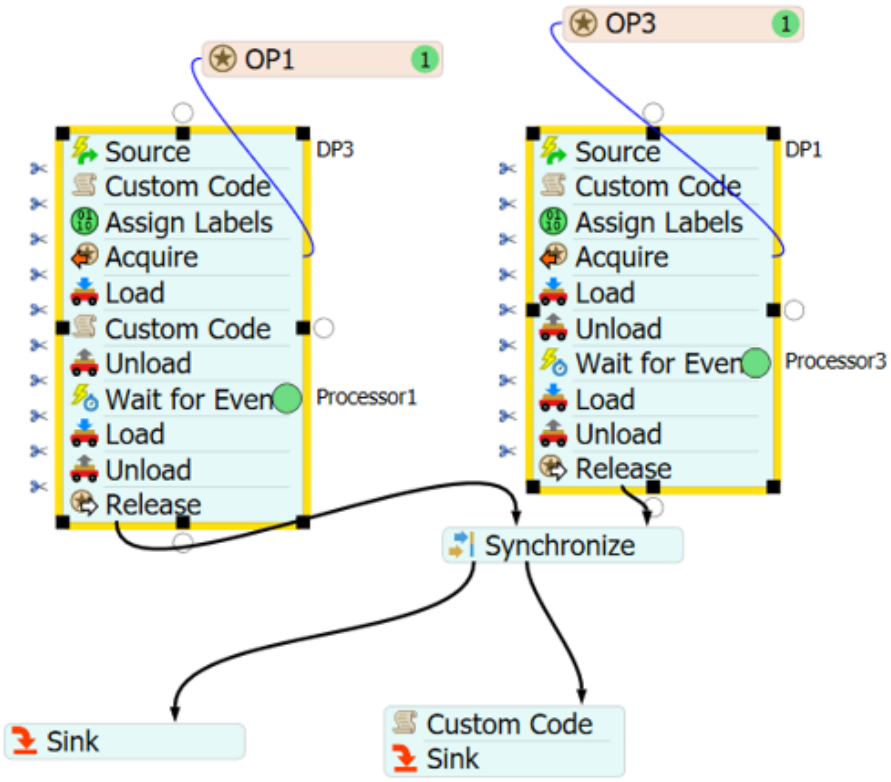

Figure 4.5: Synchronization case

To simulate the work of the critical workstation through the Petri network, there is a set of parameters to consider : Frequency of Processor 1 & 2 and Frequency of OP1 and OP3.

Suppose that the processors work with the same frequency. In case the operators do not have the same frequency, the question would be whether the pallet could continue its work if all the products were processed by a single operator?

If the two operators do not work with the same frequency, then it must be specified that the OP1 cannot process more than one product.

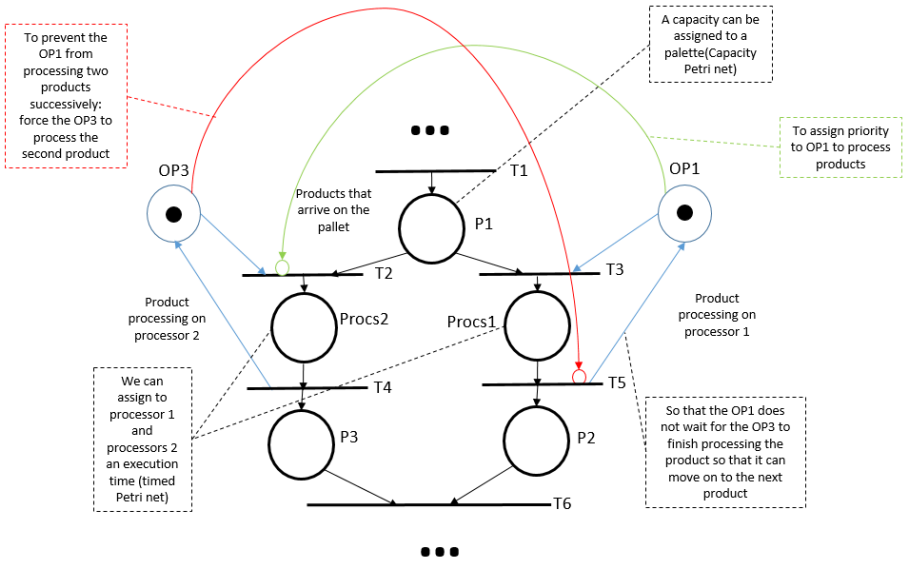

Figure 4.6: Petri network 1

This solution is still limited, it strongly depends on the time taken by the pallet to arrive at station 3, which means that it depends on the speed of the conveyor and the distance between the two workstations (the time taken by the pallet to arrive at workstation 3 must be less than the frequency of the processor), it also depends on the distance between the conveyor and the processor for each station.

Assume for simplicity that the operators work with the same frequency and that all distances and speeds are adjusted so that each operator processes a single product.

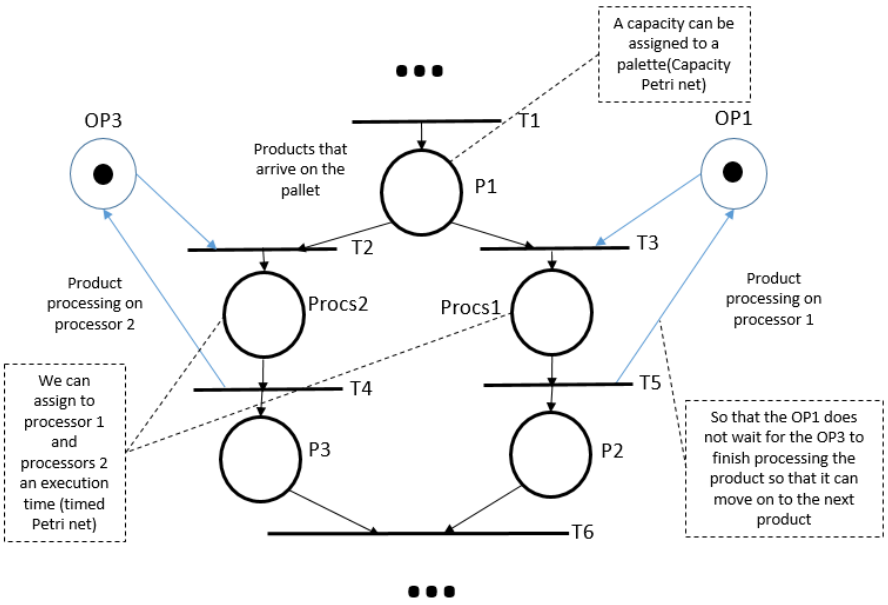

Figure 4.7: Petri network 2

So the Petri net for the whole system may be summarized as follows.

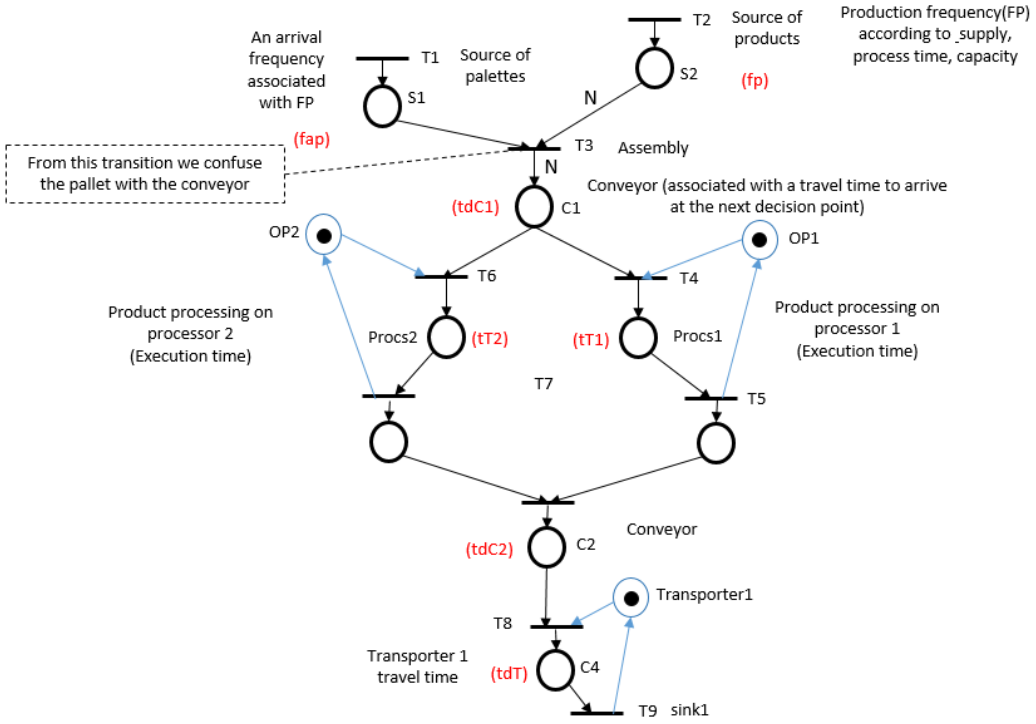

Figure 4.8: Petri network- whole system

Another synchronized case that could be observed, which assembles the two sources. Assume that the product cannot wait for the pallet.

- **Synchronous mode**

In the synchronous mode, the pallets cannot wait for products. Let tN be the time to produce N products, and tpp be the time to have an available pallet on the source1. We estimate $tpp = 0s$. let Vc be the conveyor speed and the Ds be the distance between the two sources. So $ts = Ds/Vs$ is the time the pallet takes to arrive at the source 2. And therefore $tN = tpp+ts+T$, where T is the time required to synchronize the sources.

- Advantages

  waiting times of source 1 = 0
  usage rate of source 1 = 100%

- **Asynchronous mode**

The pallets can wait for products. The only parameter to consider is the capacity of the pallet.

- Inconveniences

  waiting times of source 1 = tN
  usage rate of source 1 = $\Sigma(1 - tN/Te) * 100\%$, where Te is the time the whole process takes.

4.3 Performances Indicators

The main performance indicators that have been approved by the companies involved and the industrial team are composed of two categories.

Global KPIs

- Quality rates (Good, Rework, Reject)
- Number of good products on the horizon (excluding warm-up time)
- Buffer sizes between lines

Local KPIs

- Mean operating time of the station & waiting times
- Average utilization rate of the station
- Number of products assembled on the station
- Quality rate (Good, Rework, Reject) in number of product and cost

4.4 Use case diagram

The purpose of a use case diagram is to demonstrate the different ways that a user might interact with a system. The use case is one of the diagrams proposed by Unified Modeling Language (UML).

UML is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

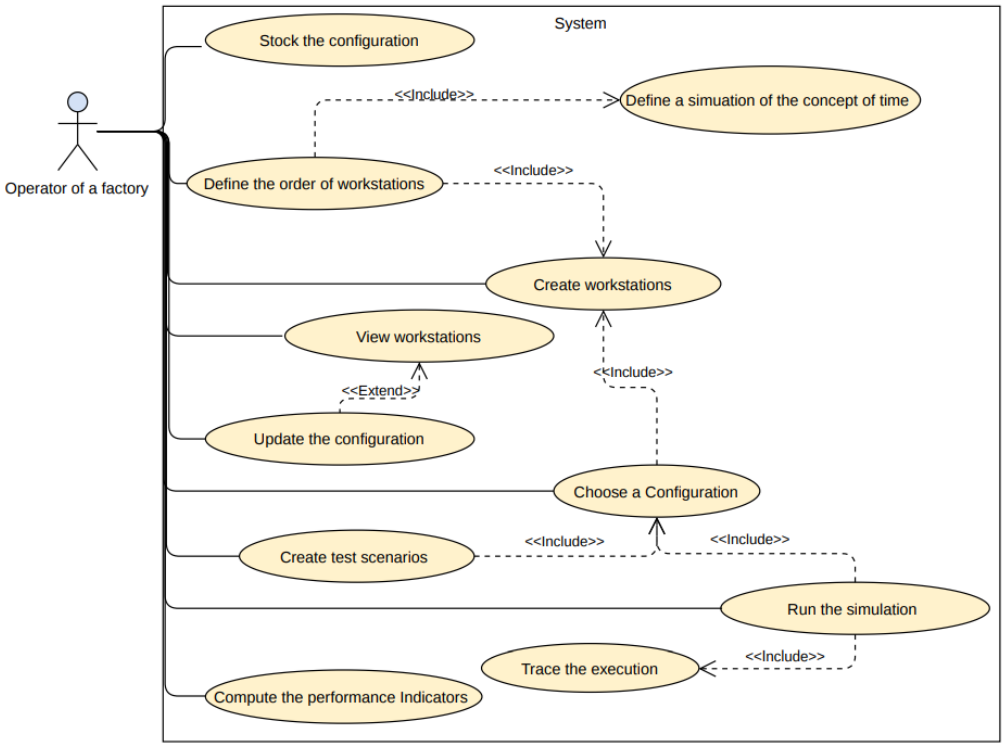

Figure 4.9: Use case diagram

4.4.1 Text description

4.4.1.1 Analysis of the Operator use case

The Operator of a factory is the end-user of the RODIC project, thus the project must address industrial issues and meet the operator's requirements. In the above diagram, the use cases of the operator were surrounded. Thereby, to simulate a production line system, the operator must be able to create different workstations and order them according to a logical chain that he wants to establish and then launch the simulation. Also, he must be able to update the configuration in case he decides to add other specifications. The operator must also have the possibility to perform test scenarios on a particular configuration and evaluate its performance, then choose between candidate configurations depending on the situations. Eventually, he must be able to stock the relevant configuration chosen.

4.4.1.2 Analysis of the System (Simulator)

The simulator is our system that we seek to build to allow industrial systems to be evaluated before being implemented in the physical world. The simulator must therefore define the concepts of the simulated time and be able to catch the execution traces.

4.5 Conclusion

This chapter explains several basic concepts in industrial engineering that should be considered when implementing a simulator. A simplified case has been presented to approximate the idea, along with a use case diagram that identifies the expected functionalities.

Chapter 5

Design

After analyzing the problem and revealing the expected functionalities, the design chapter will focus on the approaches and solutions that have been established to tackle the issue, and to improve the quality and performance of industrial systems.

5.1 Approaches

Many approaches have been established to suggest solutions. Therefore, two categories of approaches can be distinguished: general approaches that offer a holistic solution, specific approaches that are used for more detailed solution design.

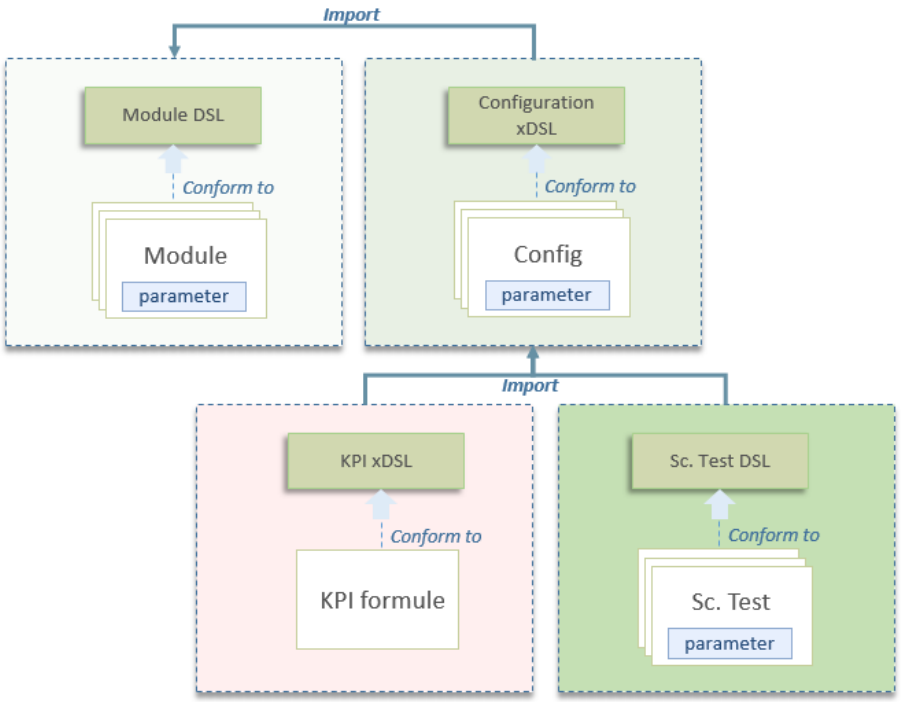

Figure 5.1: Approach

The idea here is that we have four basic components. Module component concerns the definition of the physical and logical parts of the system, the modules from which the system is composed and how the production chain takes place. It also defines the static parameters necessary for the basic operation of the module.

The second component imports the module component to specify a particular configuration for the system, thus the configuration component will add other values for certain parameters. The test scenario component, in turn, will import a specific configuration to run multiple tests on it. Ultimately, the KPI component will serve to perform calculations of several performance indicators.

5.1.1 Overall architecture diagram

The following architecture helps in the implementation of the generic approach, thus defining two basic components. First workspace, the language workbench, in which the language engineer creates a modeling language that relies on several essential elements. The second use space is a modeling workbench, where the solution is deployed and users can create and run models to calculate KPIs for each configuration.

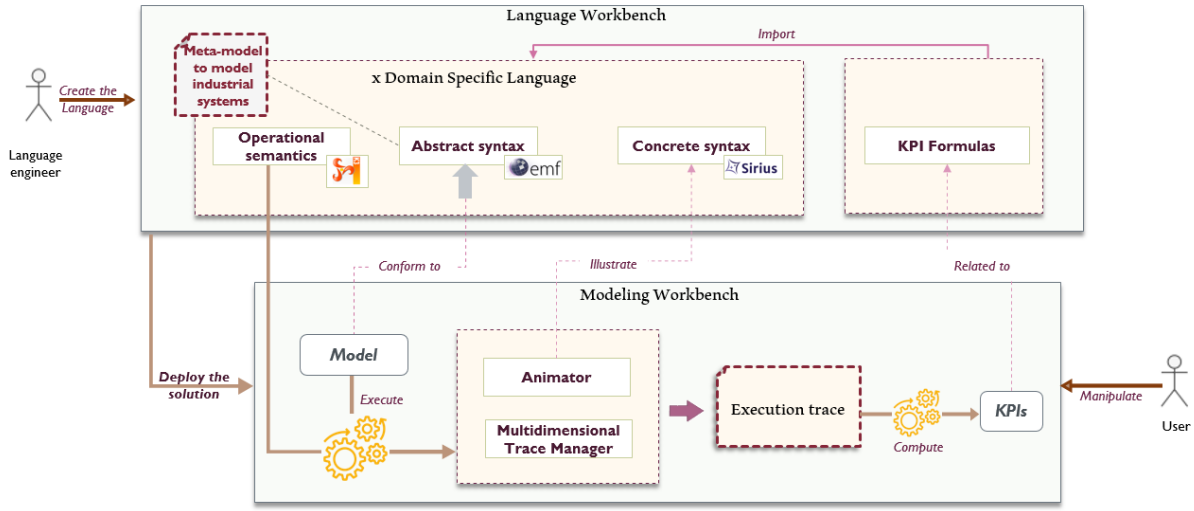

Figure 5.2: Overall architecture diagram

5.2 Static vs Dynamic (Modeling approach)

The approach that we follow to model the system composed of two parts, the static part describes the different modules that could be included in the real system, like the machines and the conveyors..., where the dynamic part includes objects that represent the dynamic image for each static component.

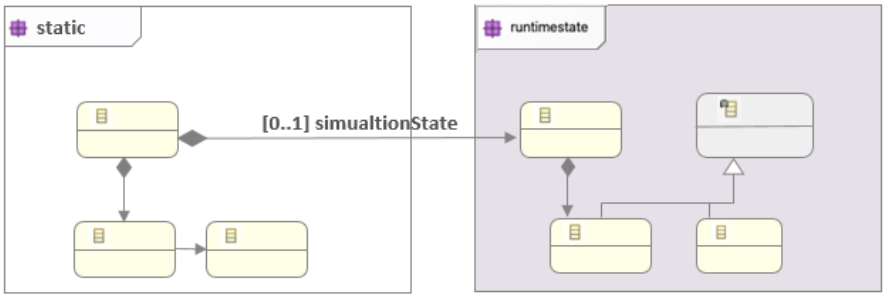

Figure 5.3: Modeling approach

The composition relation *simulationState* that appears in the figure, links between the static and the dynamic package , with a lower cardinality equal to 0 which means that when the system is not yet running there will be no state. Otherwise, when the execution starts, for each instant, there will be only one state representing the system at that specific moment with specific values, which explains the fact that the upper cardinality is equal to 1.

5.3 Building the notions of a simulator

5.3.1 State capture

The capture of the state is a picture of the system at a particular instant, the pictures will be taken during the execution of the models specifically at the END of

each particular task, i.e. when a particular module completes its job.

5.3.2 Element capture

To track the work of each element during the execution, the model must define a class that will encapsulate the different values of the module parameters at each moment. It will include the start time, end time, the state to indicate if the element is starting or in progress or finishing its work , and the type of the element to specify whether the element is active or pending.

5.3.3 Simulated time

The time in the simulator cannot be real time, otherwise it will not be a simulator. For this reason, concepts of time must be refined to suit the needs of the situation, but a simulator must also retain the underlying principle of time that it should always move forward in a positive way.

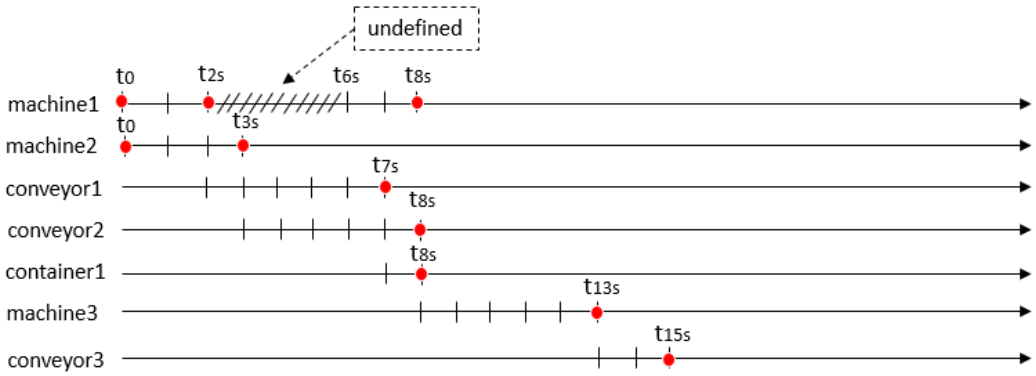

Figure 5.4: Simulated time

In our case, we define the time as follows.

T= t + Di / t0: initial time & Di: duration of each element, Undefined is real time that is undefined in the space T. The red circles represent when the system captures will be taken.

5.3.4 Hypotheses

To simulate the industrial system, we rely on two assumptions.

- H1: An element cannot receive run requests when it's already running.
- H2: Containers cannot start their jobs unless they are triggered by other elements.

5.4 Sequence diagram

In this part, we will use a sequence diagram to describe the dynamic aspect of the whole system focusing on the interactions between different components, emphasizing the chronology of events.

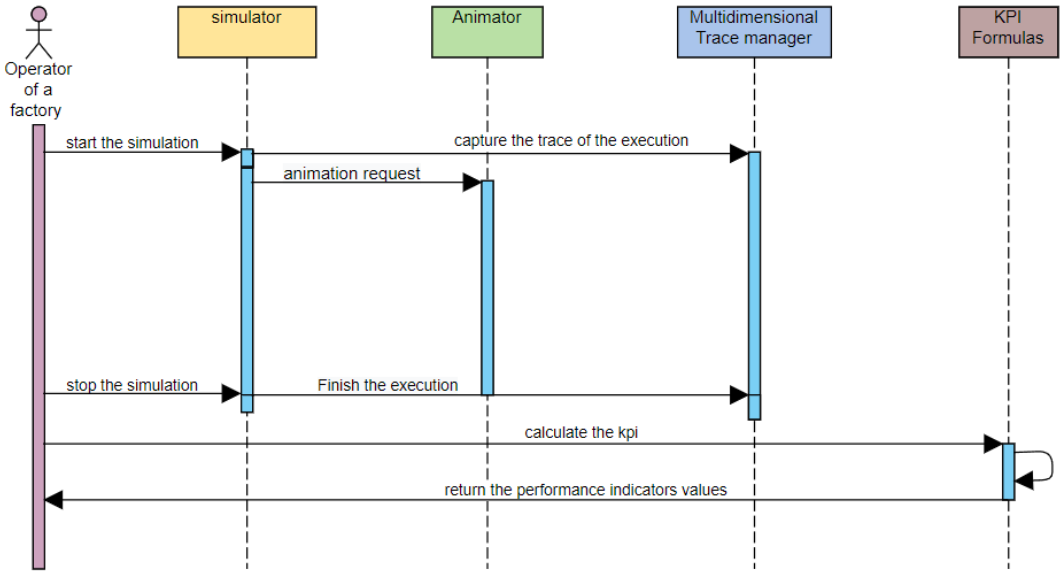

Figure 5.5: Sequence diagram

5.5 Class diagram

The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. Thus, we seek to represent the structure of an industrial system in a more abstract view by using a class diagram, where the classes, their attributes, the operations and the relationships between them are described.

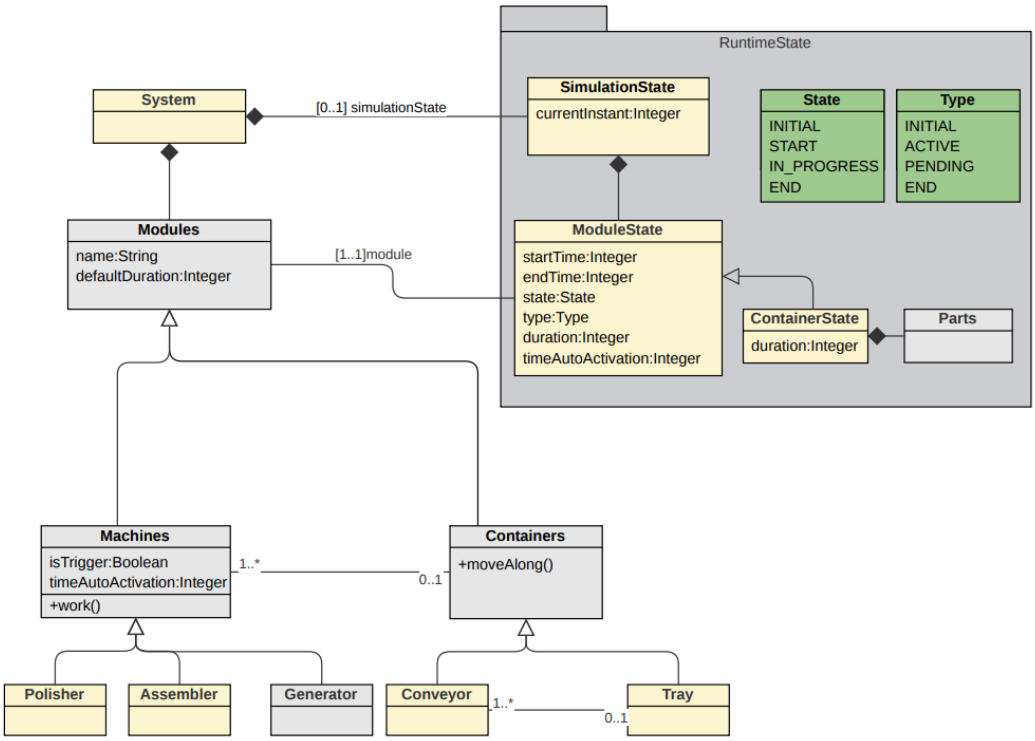

Figure 5.6: Class diagram

5.6 Conclusion

After analyzing the situation, this chapter highlights the ideas to solve the problem raised. For that reason, several concepts have been defined and the overall structure has been described along with a sequence diagram that allows the visualization of the chronology of events.

Chapter 6

Development

The development phase is where we bring life to models, hence this chapter will cover all development steps, will implement the notions previously seen, such as the operational semantics, the simulated time and the graphical design.

6.1 Description of the use case (Production Line System)

This use case shows a simple production line producing hammers from heads and handles.

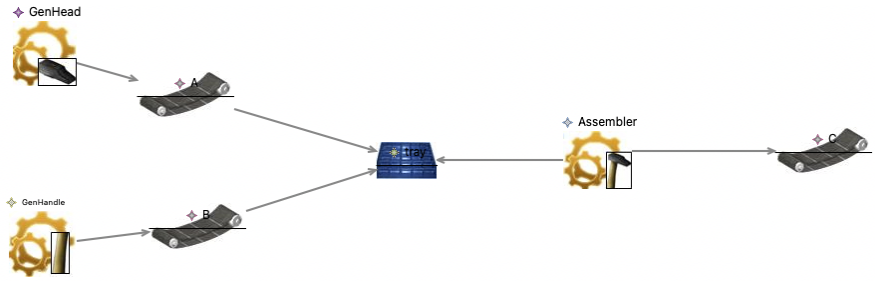

Figure 6.1: Production Line System

This production line contains 3 machines, each of them connected to a conveyor. A GenHead and a GenHandle to generate a head and a handle respectively, and put them on the conveyor to bring them to the tray. The tray simulates temporary storage that holds generated parts while waiting for the assembler to retrieve them. An assembler couldn't start producing a hammer unless it got at least a handle and a head as an input . Once a hammer is generated, it will be placed on a conveyor.

6.1.1 Operational semantics

The semantics of this modeling language are given by the set of Henshin rules shown below.

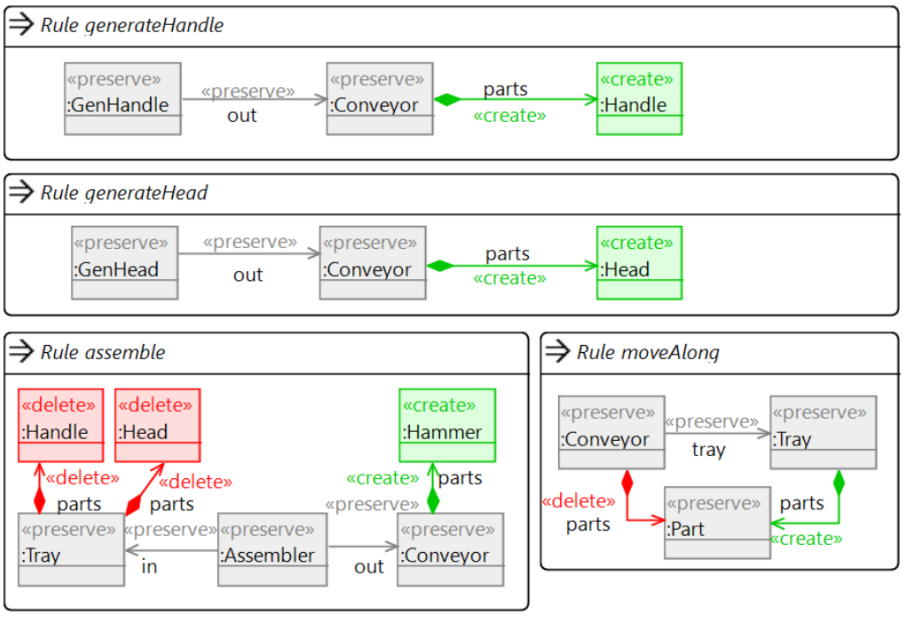

Figure 6.2: Operational semantics

6.1.2 Meta-model

The *org.eclipse.emf.ecore* package provides an API for the Ecore dialect of UML. This package allows the implementation of different business classes with a graphical interface through a file .aird, furthermore it allows the generation of the code.

The metamodel designed for the production-line system use case can be seen in the figure below.

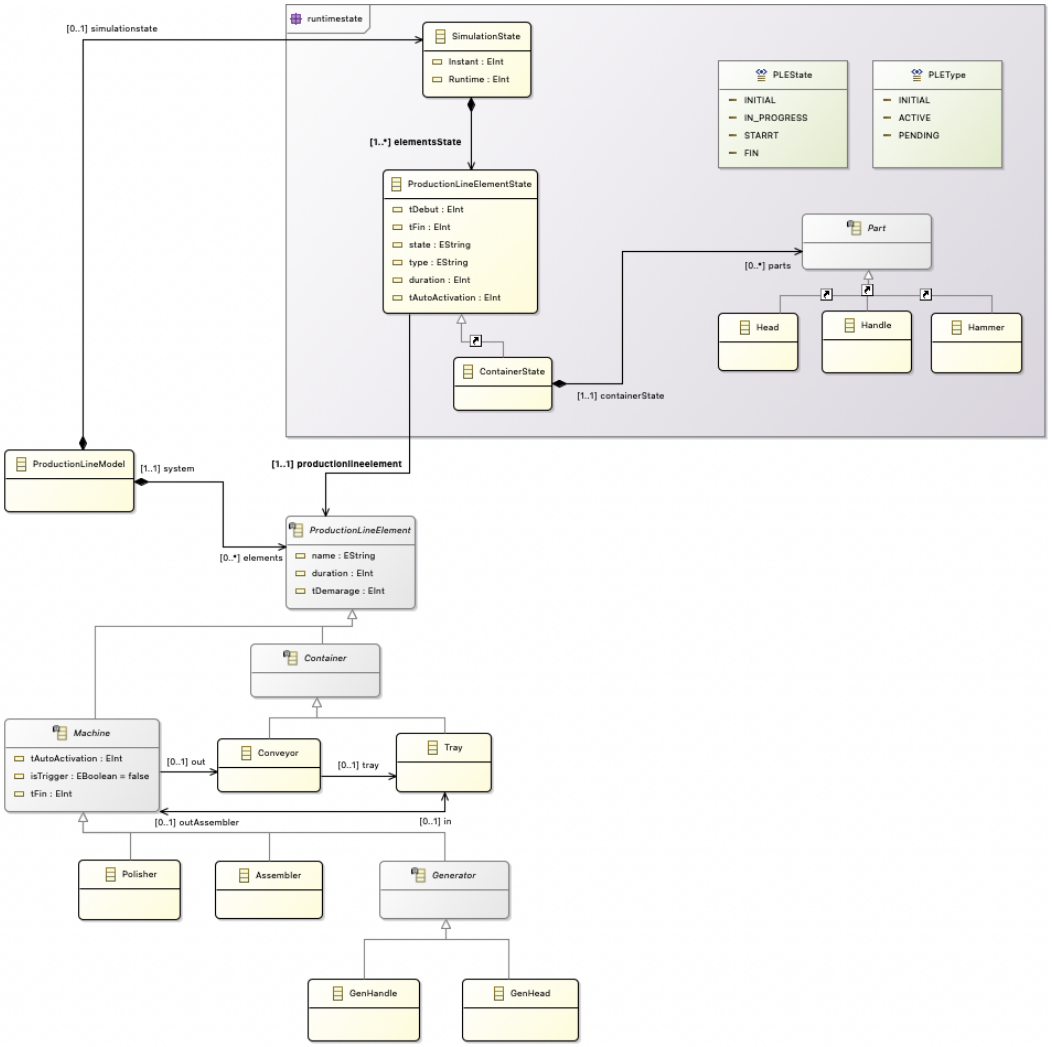

Figure 6.3: Meta-model

6.1.3 Runtime approach

As mentioned earlier, the *ProductionLineElementState* class in the runtime package captures the different state of each element during the execution. For example, the *ProductionLineElementState* corresponding to the GenHead machine at an instant equal to 0, will have certain values that will change for the next instant, as it is demonstrated below.

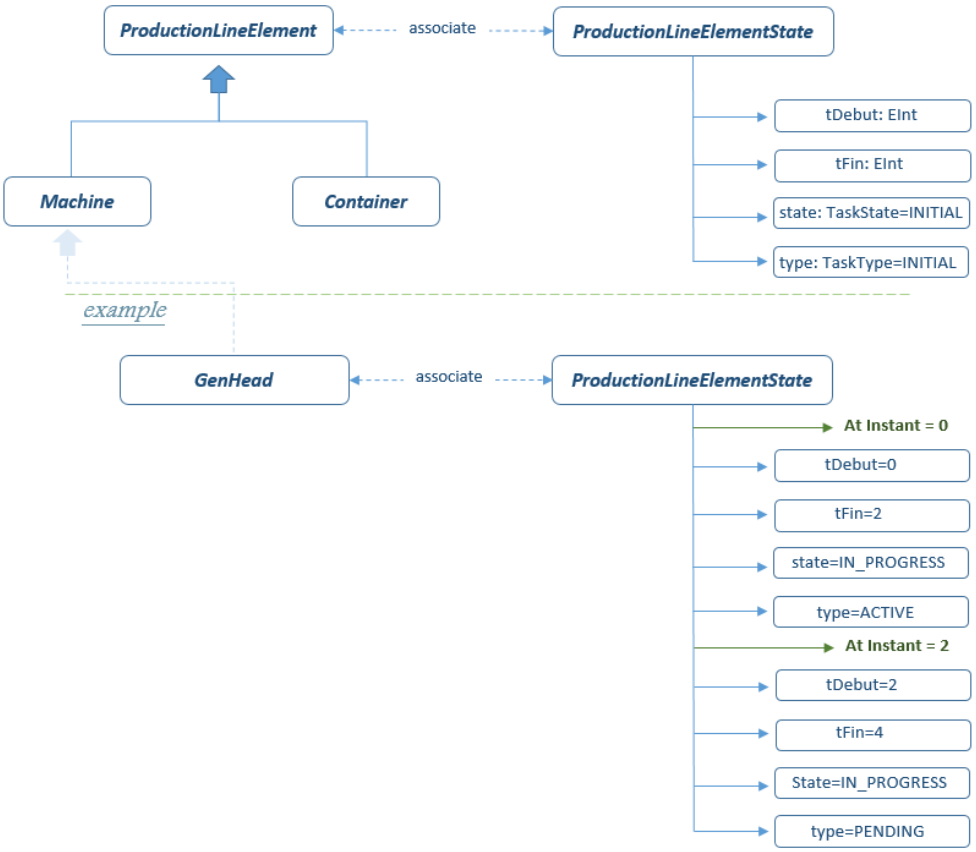

Figure 6.4: Runtime approach

6.1.4 Graphical representation

To provide a graphical representation, we use the sirius project that allows us to easily create graphical modeling for a given ecore meta-model. It is based on a viewpoint approach where it is possible to create diagrams, add components and associate forms or images to modules. Furthermore, there is a possibility to create section tools to enable the user to directly modify the diagram.

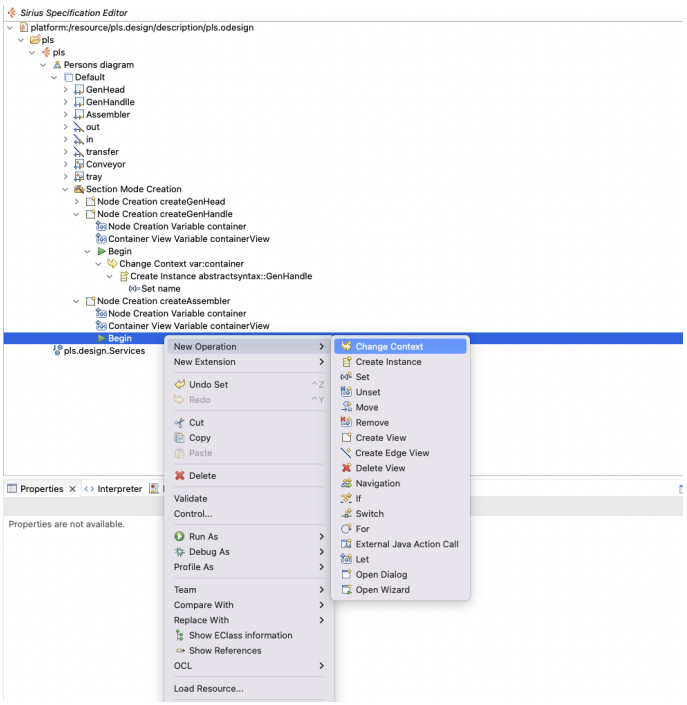

Figure 6.5: Sirius editor-1

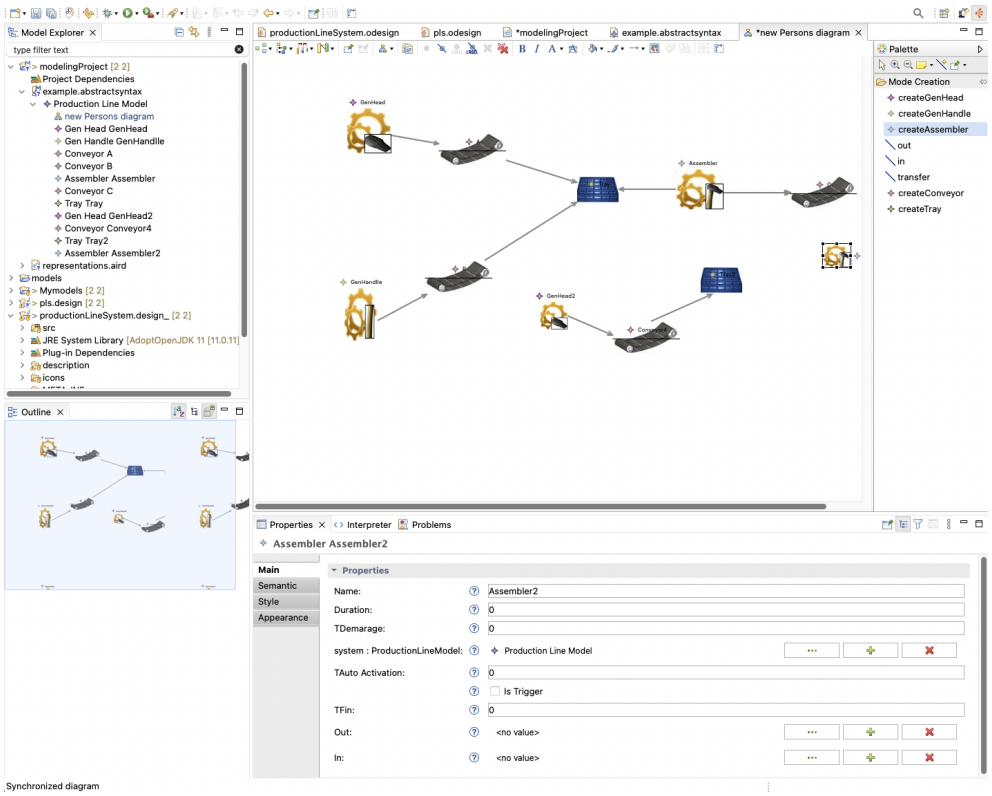

Figure 6.6: Sirius editor-2

6.1.5 Run the project in debug mode

After creating the meta-model and implementing the operational semantics in the language workbench , we run the project in the debug mode as an Eclipse Application to open the modeling workbench. Once done, we create a modeling project to create a model conforming to the meta-model and relaunch the debug mode with specific parameters. It is important to enable gemoc trace manager in the debug configuration.

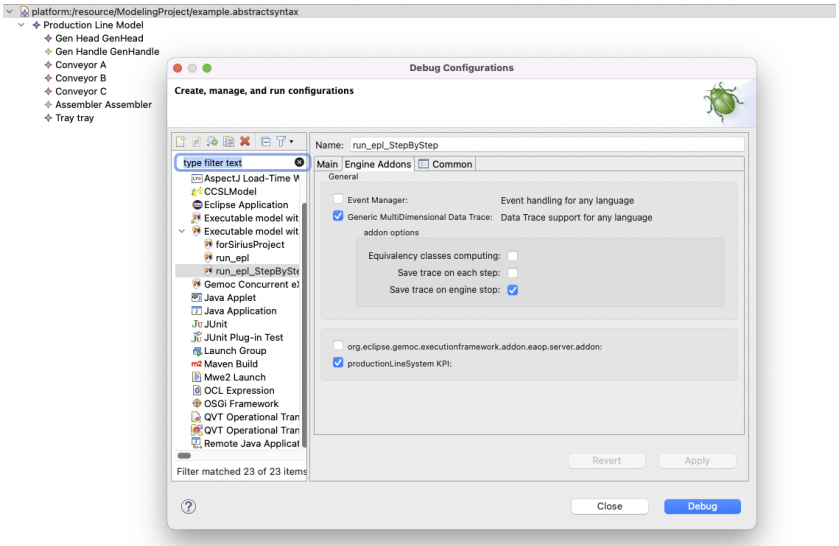

Figure 6.7: Debug Configuration

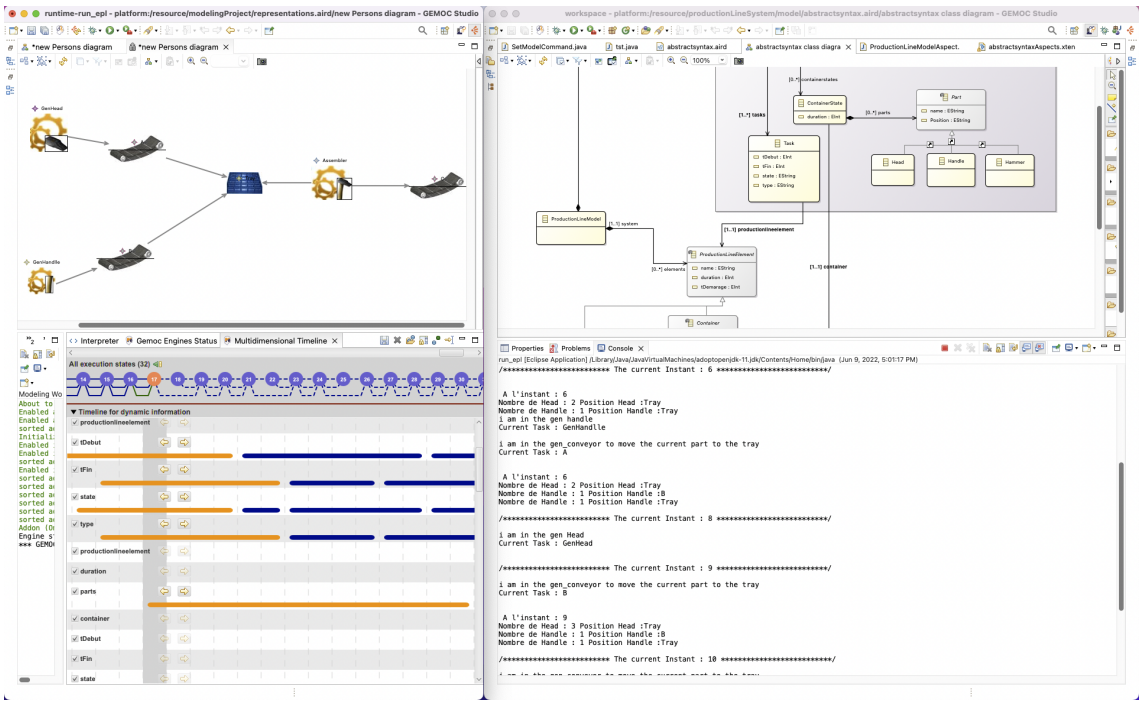

Figure 6.8: Debug

6.1.6 Run the project with variable parameters

In this step, we can specify not only system parameters, but also model parameters. Running the model with different values of different parameters such as system running time, machine duration, conveyor speed, etc., allows different test scenarios to be simulated.
Thus we can assign these arguments in the corresponding box "Model initialization arguments".

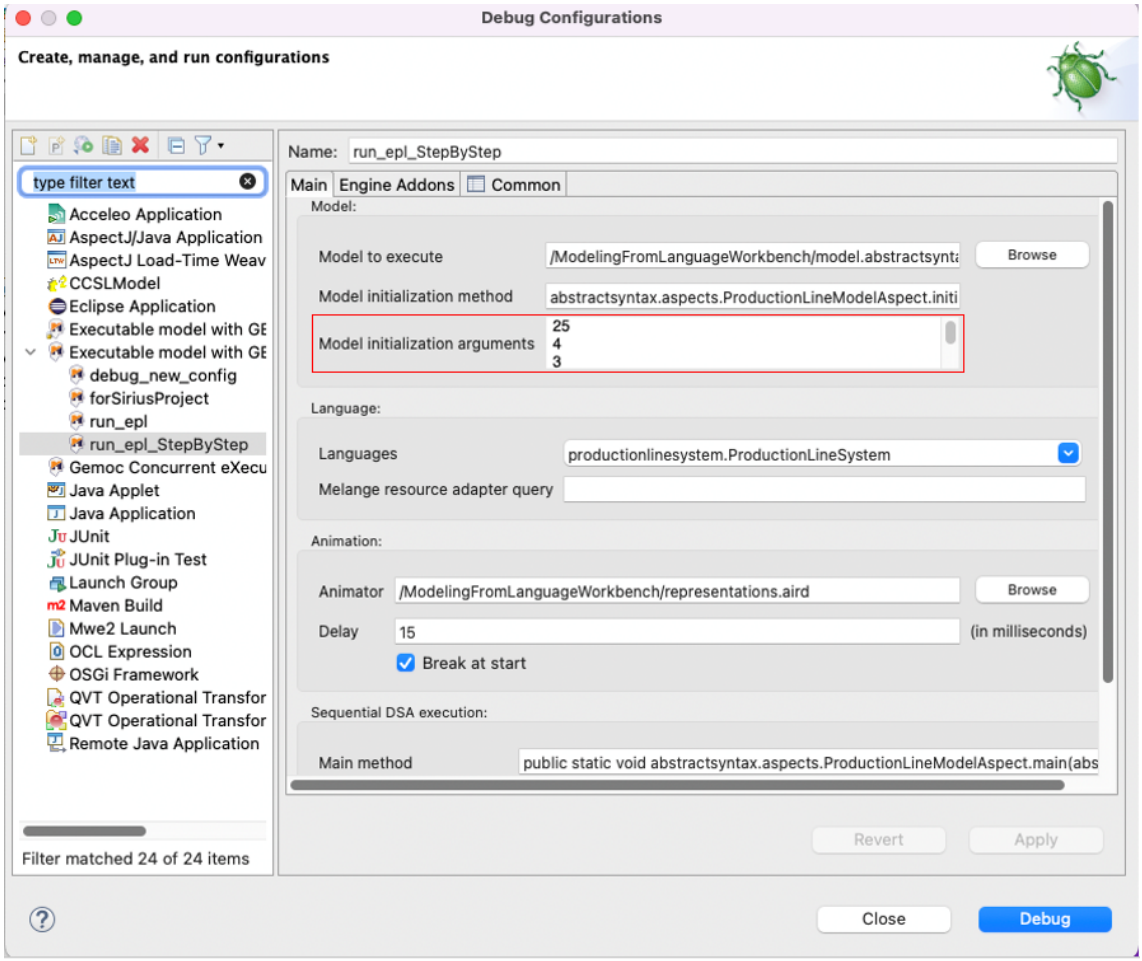

Figure 6.9: Model setting

6.1.7 Console capture

After executing the model, we try to capture the trace execution in the console.

```
start initialisation...
i am in the gen Head
Current Task : GenHead

i am in the gen handle
Current Task : GenHandle

end initialisation
Starting the execution

/************************* The current Instant : 2 *************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : A


/************************* The current Instant : 3 *************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : B


/************************* The current Instant : 4 *************************/

i am in the gen Head
Current Task : GenHead


/************************* The current Instant : 6 *************************/

i am in the gen handle
Current Task : GenHandle

i am in the gen_conveyor to move the current part to the tray
Current Task : A

i am enter the  Assembler
i am in the gen Assembler
Current Task : Assembler


/************************* The current Instant : 8 *************************/

i am in the gen Head
Current Task : GenHead


/************************* The current Instant : 9 *************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : B


/************************* The current Instant : 10 *************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : A


/************************* The current Instant : 11 *************************/

i am in the hammer conveyor
Current Task : C


/************************* The current Instant : 12 *************************/

i am in the gen Head
Current Task : GenHead

i am in the gen handle
Current Task : GenHandle

i am enter the  Assembler
i am in the gen Assembler
Current Task : Assembler
```

Figure 6.10: console-1

```
/************************ The current Instant : 13 ************************/

finish

/************************ The current Instant : 14 ************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : A


/************************ The current Instant : 15 ************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : B


/************************ The current Instant : 16 ************************/

i am in the gen Head
Current Task : GenHead


/************************ The current Instant : 17 ************************/

i am in the hammer conveyor
Current Task : C


/************************ The current Instant : 18 ************************/

i am in the gen handle
Current Task : GenHandle

i am in the gen_conveyor to move the current part to the tray
Current Task : A

i am enter the  Assembler
i am in the gen Assembler
Current Task : Assembler


/************************ The current Instant : 19 ************************/

finish

/************************ The current Instant : 20 ************************/

i am in the gen Head
Current Task : GenHead

Init PropertyTesterClass

/************************ The current Instant : 21 ************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : B


/************************ The current Instant : 22 ************************/

i am in the gen_conveyor to move the current part to the tray
Current Task : A


/************************ The current Instant : 23 ************************/

i am in the hammer conveyor
Current Task : C


/************************ The current Instant : 24 ************************/

i am in the gen Head
Current Task : GenHead

i am in the gen handle
```

Figure 6.11: console-2

6.1.8 Manual representation of the simulation

For the created model, the manual simulation will proceed as follows.

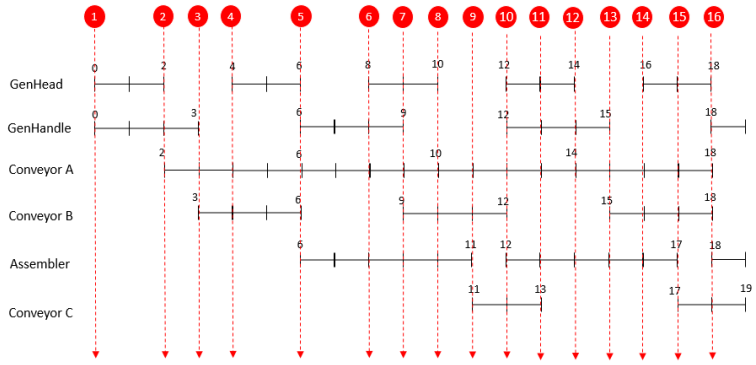

Figure 6.12: Manual representation of the simulation

6.1.9 Trace of gemoc

When the execution finishes, we obtain an *execution.trace* file that contains values of different execution states for each specific element in the runtime package. For instance, the figure below depicts the trace of *ConveyorB* and we can clearly observe that it conforms to what was expected in the manual simulation.

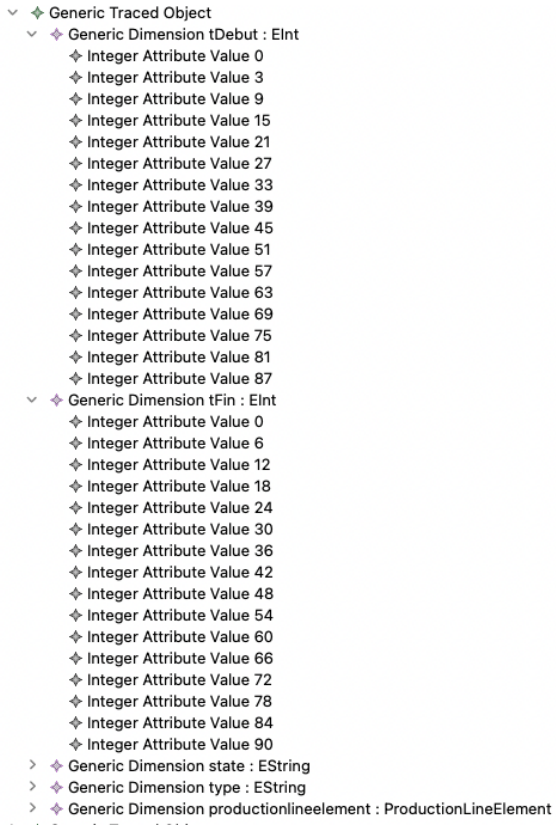

Figure 6.13: Trace of gemoc

6.1.10 Performance indicators

After getting the execution trace file. We can right click and choose «Compute» from the pop-up list. And we can get the value of KPIs corresponding to the selected

execution. In this picture below, we show the KPI calculations for two different executions. The first execution completes in 9 time units, and the second in 322 time units.

```
----------------------------------------

------------KPI LOCAL-------------
/************************ concernedElement : tray ************************/
 Operating Time : 0 uT
 Percentage Operating Time : 0.0 --> 0%
 Percentage Waiting Time : 1.0 --> 100%
 Number of product assembled on the station : 0

/************************ concernedElement : GenHead ************************/
 Operating Time : 5 uT
 Percentage Operating Time : 0.5555555555555556 --> 55%
 Percentage Waiting Time : 0.4444444444444444 --> 44%
 Number of product assembled on the station : 2

/************************ concernedElement : GenHandle ************************/
 Operating Time : 6 uT
 Percentage Operating Time : 0.6666666666666666 --> 66%
 Percentage Waiting Time : 0.33333333333333337 --> 33%
 Number of product assembled on the station : 2

/************************ concernedElement : A ************************/
 Operating Time : 7 uT
 Percentage Operating Time : 0.7777777777777778 --> 77%
 Percentage Waiting Time : 0.2222222222222222 --> 22%
 Number of product assembled on the station : 1

/************************ concernedElement : B ************************/
 Operating Time : 3 uT
 Percentage Operating Time : 0.3333333333333333 --> 33%
 Percentage Waiting Time : 0.6666666666666667 --> 66%
 Number of product assembled on the station : 1

/************************ concernedElement : C ************************/
 Operating Time : 0 uT
 Percentage Operating Time : 0.0 --> 0%
 Percentage Waiting Time : 1.0 --> 100%
 Number of product assembled on the station : 0

/************************ concernedElement : Assembler ************************/
 Operating Time : 3 uT
 Percentage Operating Time : 0.3333333333333333 --> 33%
 Percentage Waiting Time : 0.6666666666666667 --> 66%
 Number of product assembled on the station : 0

------------KPI GLOBAL-------------

         :..........: :............:
         :  Nombre  : :    Type    :
.....................................
A        :    1     : :   Head
.....................................
B        :    1     : :   Handle
.....................................

 ##System Operating Time : 9
 ##Total number of products produced : 0

----------------------------------------
```

Figure 6.14: KPIs for the first execution

```
-----------------------------------------

-----------KPI LOCAL-------------
/************************ concernedElement : A ************************/
 Operating Time : 320 uT
 Percentage Operating Time : 0.9937888198757764 --> 99%
 Percentage Waiting Time : 0.006211180124223614 --> 0%
 Number of product assembled on the station : 80

/************************ concernedElement : GenHead ************************/
 Operating Time : 162 uT
 Percentage Operating Time : 0.5031055900621118 --> 50%
 Percentage Waiting Time : 0.49689440993788825 --> 49%
 Number of product assembled on the station : 81

/************************ concernedElement : GenHandle ************************/
 Operating Time : 162 uT
 Percentage Operating Time : 0.5031055900621118 --> 50%
 Percentage Waiting Time : 0.49689440993788825 --> 49%
 Number of product assembled on the station : 54

/************************ concernedElement : B ************************/
 Operating Time : 160 uT
 Percentage Operating Time : 0.4968944099378882 --> 49%
 Percentage Waiting Time : 0.5031055900621118 --> 50%
 Number of product assembled on the station : 53

/************************ concernedElement : C ************************/
 Operating Time : 104 uT
 Percentage Operating Time : 0.32298136645962733 --> 32%
 Percentage Waiting Time : 0.6770186335403727 --> 67%
 Number of product assembled on the station : 52

/************************ concernedElement : Assembler ************************/
 Operating Time : 264 uT
 Percentage Operating Time : 0.8198757763975155 --> 81%
 Percentage Waiting Time : 0.18012422360248448 --> 18%
 Number of product assembled on the station : 52

/************************ concernedElement : tray ************************/
 Operating Time : 0 uT
 Percentage Operating Time : 0.0 --> 0%
 Percentage Waiting Time : 1.0 --> 100%
 Number of product assembled on the station : 0

-----------KPI GLOBAL-------------

         :..........: :............:
         :  Nombre  : :    Type    :
.....................................
A        :    1     : :   Head
.....................................
B        :    1     : :   Handle
.....................................
C        :    52    : :   Hammer
.....................................
tray     :    27    : :   Head
.....................................

 ##System Operating Time : 322
 ##Total number of products produced : 52

-----------------------------------------
```

Figure 6.15: KPIs for the second execution

6.1.11 Gemoc Trace Manager

Gemoc collects the traces from the execution.

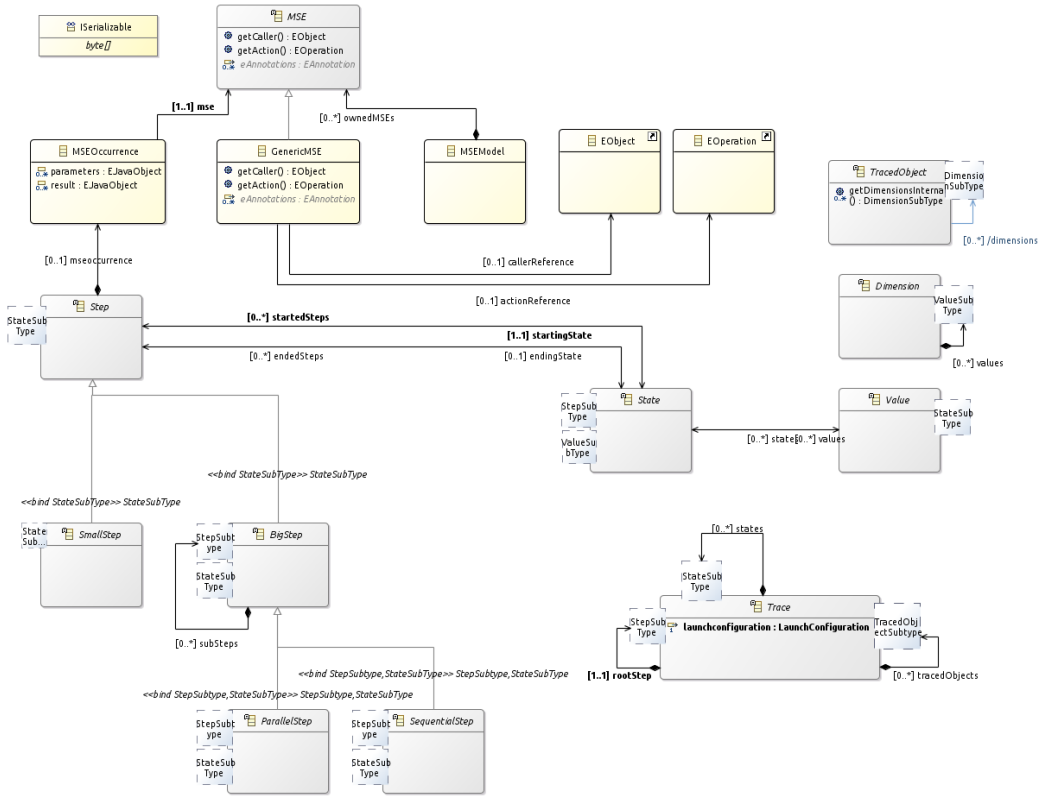

Figure 6.16: Meta-Model Trace Manager

Model specific event (MSE), the root abstract class. In the execution, the ***MSE*** presents a pair composed of the instance of the ***EClass*** and the corresponding ***EOperation***, and it operates at the Model level of the executable System. In our case, if we assume that we have two conveyors, the conveyor “A” and “B”, and the Conveyor class has a “*start*(_currentInstant)” operation, so the pair (“A”, “*start*(_currentInstant)”) it’s a ***MSE***, and ***MSEOccurence*** will represent in this case the two occurrences of the conveyor : (“A”, “*start*(_currentInstant)”); (“B”, “*start*(_currentInstant)”).

6.1.12 Approach

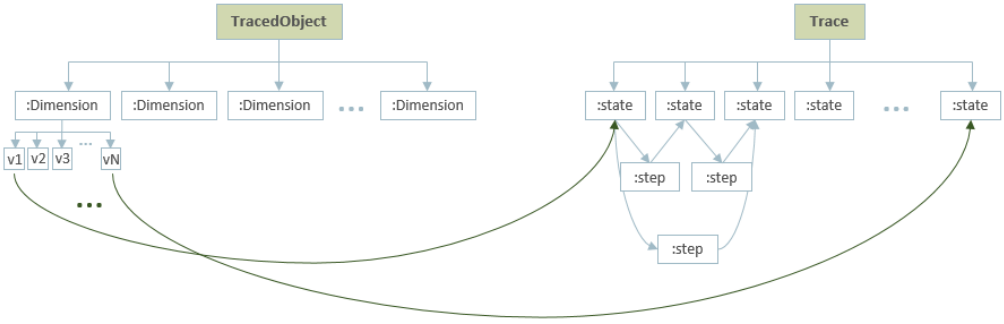

Figure 6.17: Trace Manager Approach

So here we capture the execution traces at two different levels, and they are linked by a commun orthogonal intersection. The global level is where we photograph the state of the entire system at an instant, so the ***Trace*** object is composed of different states, each ***State*** is linked to the next ***State*** by a ***Step*** object, which could be end steps or start steps. As the ***State*** object stores the different values of each class in the executable model, then there exists a reference relation between the ***State*** and ***Value***. Thus, we can navigate to the local level through this connection.

The local level refers to a particular class in the model, with different values that had been assigned to each attribute during the execution. Thereby, we represent the concerned class with the ***TracedObject*** and its different attributes with ***Dimension***. The ***TracedObject*** is necessarily contained within the run-time package.

Figure 6.18: Example

6.1.13 Simple Trace Method

Simple trace addon is another plugin to generate the execution trace of the model.

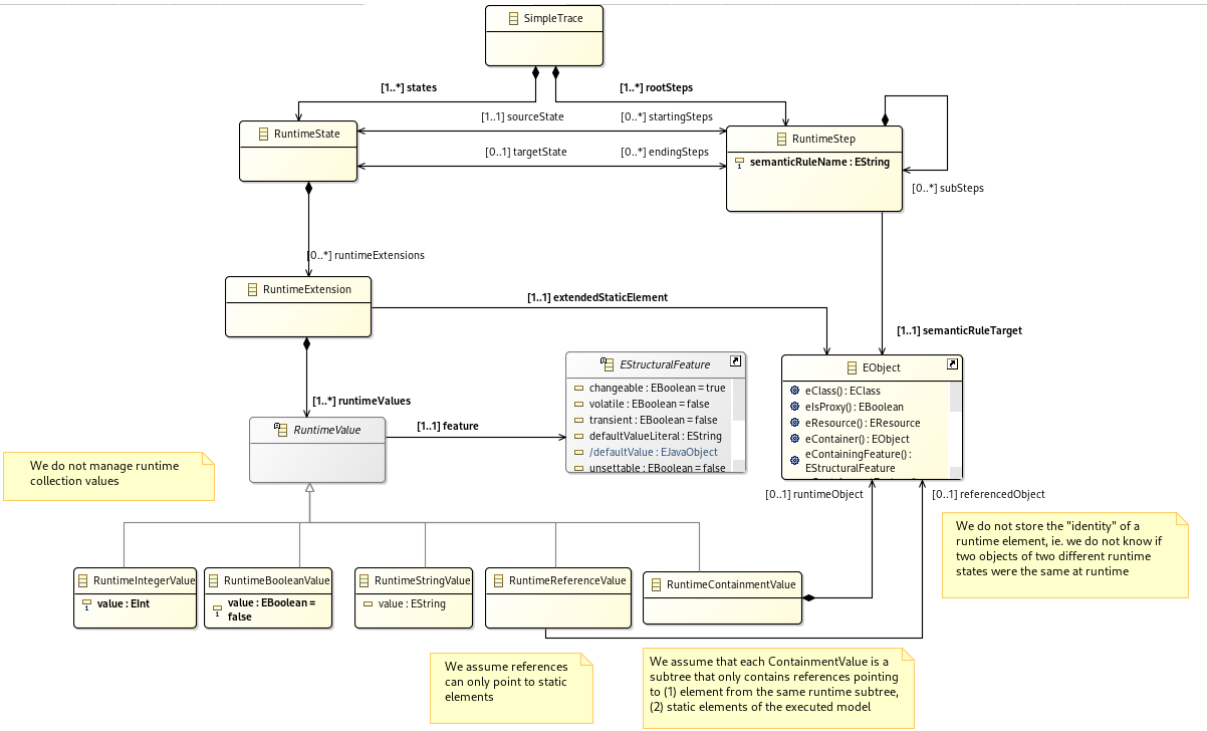

Figure 6.19: Simple Trace Meta-model

Other than the previous approach, the simple trace method does not capture the set of values of each dimension alone, instead it captures for each particular instant the complete view of the run time state.

Figure 6.20: Simple Trace File

6.1.14 Failing work

The approach adopted in the previous section was an offline approach, i.e. the performance measurements will be performed after completing the execution and obtaining the trace. But before that, we used the online approach taken in Thibault's paper section 3.2,where the computation of performance metrics happens on-the-fly.

The previous approach was based on creating a language to evaluate the performance: *Performance Estimation Language (PEL)*. The meta-model of the PEL language is given by an ecore diagram shown below.

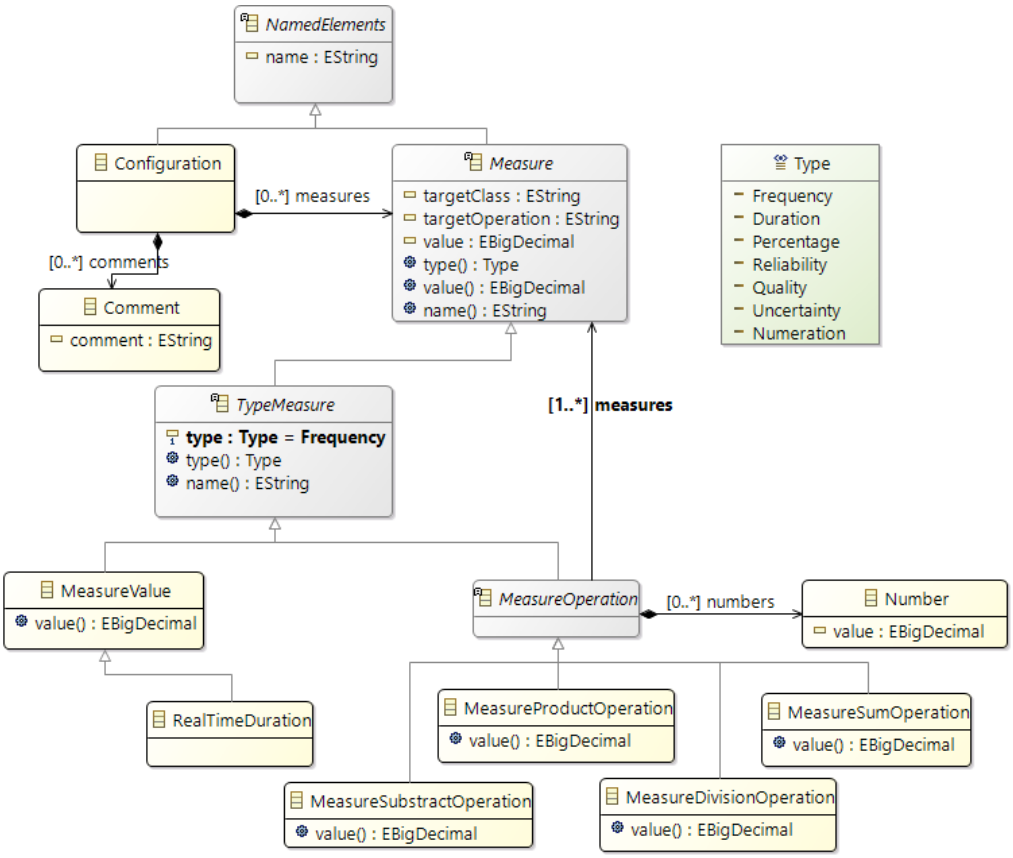

Figure 6.21: PEL metamodel

After that, we generate a xtext project from the language metamodel, to allow users to configure the system and to calculate the measurements.

```
model.epl ×
Configuration Ayt28{
    /* ****************Static Value*****************/
    Measure "GHeaD": GenHead.duration=2000.00;
    Measure "GHanD": GenHandle.duration=3000.00;
    Measure "AssD": Assembler.duration=12000.00;

    Measure "GHeaF": GenHead#work.frequency=5.00;
    /*Measure "ocl": ocl: 'calculate the number of hammer generate '*/
    /* ****************Calculated Value************ */
    GenHead#work.duration= "GHeaD" *  3.01 *2.00*6.00;
    GenHandle#work.duration= "GHanD" *  3.01 *2.00*6.00;
}
```

Figure 6.22: Configuration model with *PEL*

Thus, we obtained the following results.

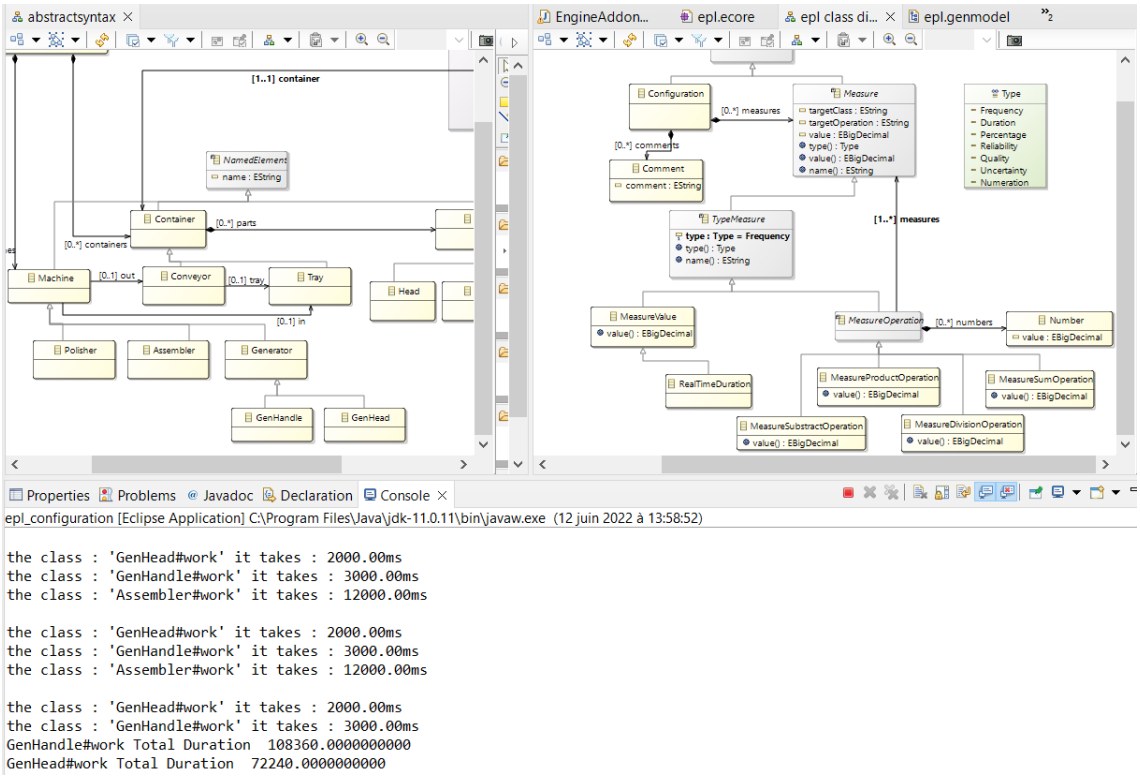

Figure 6.23: Results

6.2 Conclusion

This chapter focuses on the development part with the objective of showing how the previous explained approaches have been implemented. It is important to note that the subject of this internship is a thesis subject which will be studied over the next three years.

General conclusion

Simulation process is very important in any production domain, especially when it comes to the industrial sector, where time and cost are the most crucial factors. The system should pass through the verification and validation phases before being implemented in the real world, otherwise each failing part will cost money and time. In the domain of Model-Driven Engineering, the systems could be simulated with models and driving rules where the code could be easily generated. This MDE characteristic was widely used to perform the work of this internship that aims to evaluate the performance of a configuration of RMS with models and test scenarios.

In the present document, we outline six chapters that show the study phases. We first set the context general of the internship then we present the scientific background required for the work presented. After that, we discuss the thesis related to our work and then we illustrate the analysis process and the solution suggested. Eventually, we depict the development phase.

Very valuable work has been done to fulfill the objective of this internship, nevertheless it is still only a beginning stage for the thesis work, I hope this work will live up to the expectations of the LS2N Lab, which gave me the opportunity to be a member of the team for 6 months, working with highly qualified people, And learn from their expertise and expand my scientific and academic horizon.

Bibliography

[1] The ETSI Test Description Language TDL and its Application:. In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, pages 601–608, Lisbon, Portugal, 2014. SCITEPRESS - Science and and Technology Publications.

[2] Thibault Beziers la fosse. *Model-driven methods for dynamic analysis applied to energy-aware software engineering*. PhD Thesis, 2021.

[3] Hugo Bruneliere, Jordi Cabot, Frédéric Jouault, and Frédéric Madiot. MoDisco: a generic and extensible framework for model driven reverse engineering. In *Proceedings of the IEEE/ACM international conference on Automated software engineering - ASE '10*, page 173, Antwerp, Belgium, 2010. ACM Press.

[4] Erica Capawa Fotsoh. *Contribution à la reconfiguration des lignes de production : définition et démarche de choix de configurations alternatives*. PhD Thesis, 2021.

[5] Tony Clark, Andy Evans, Paul Sammut, James Willans, and Xactium Limited. An eXecutable Metamodelling Facility for Domain Specific Language Design. page 8.

[6] S. Jack Hu, Yoram Koren, and Kathryn E. Stecke. Introduction. *International Journal of Flexible Manufacturing Systems*, 17(4):259–260, October 2005.

[7] Stuart Kent. Model Driven Engineering. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Michael Butler, Luigia Petre, and Kaisa Sere, editors, *Integrated Formal Methods*, volume 2335, pages 286–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

[8] Faezeh Khorram, Erwan Bousse, Jean-Marie Mottu, and Gerson Sunyé. Adapting TDL to Provide Testing Support for Executable DSLs. *The Journal of Object Technology*, 20(3):6:1, 2021.

[9] Isabela Maganha, Cristovao Silva, and Luis Miguel D.F. Ferreira. Understanding reconfigurability of manufacturing systems: An empirical analysis. *Journal of Manufacturing Systems*, 48:120–130, July 2018.

[10] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren. [No title found]. *Journal of Intelligent Manufacturing*, 11(4):403–419, 2000.

[11] Stephen J. Mellor, Kendall Scott, Axel Uhl, and Dirk Weise. Model-Driven Architecture. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Jean-Michel

Bruel, and Zohra Bellahsene, editors, *Advances in Object-Oriented Information Systems*, volume 2426, pages 290–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

[12] Tom Mens and Pieter Van Gorp. A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, March 2006.

[13] Alessia Napoleone, Alessandro Pozzetti, and Marco Macchi. A framework to manage reconfigurability in manufacturing. *International Journal of Production Research*, 56(11):3815–3837, June 2018.

[14] Mustafa M. Tikir and Jeffrey K. Hollingsworth. Efficient instrumentation for code coverage testing. *ACM SIGSOFT Software Engineering Notes*, 27(4):86–96, July 2002.