FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS DEPARTMENT
UNDERGRADUATE COURSE IN CONTROL AND AUTOMATION ENGINEERING

Fabiano Junior Maia Manschein

Modeling of a Decision Support System for the ramp-up phase of Line-less Assembly Systems

Florianópolis
2022

Fabiano Junior Maia Manschein

Modeling of a Decision Support System for the ramp-up phase of Line-less Assembly Systems

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergraduate Course in Control and Automation Engineering of the Federal University of Santa Catarina.
Supervisor: Prof. Rodolfo César Costa Flesch, Dr.
Co-supervisor: Jonas Rachner, Eng.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Manschein, Fabiano Junior Maia
Modeling of a Decision Support System for the ramp-up phase of Line-less Assembly Systems / Fabiano Junior Maia Manschein ; orientador, Rodolfo César Costa Flesch, coorientador, Jonas Rachner, 2022.
72 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Decision Support Systems. 3. Ramp-up. 4. Line-less Assembly Systems. I. Flesch, Rodolfo César Costa. II. Rachner, Jonas. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Fabiano Junior Maia Manschein

Modeling of a Decision Support System for the ramp-up phase of Line-less Assembly Systems

This dissertation was evaluated in the context of the subject DAS5511 (Course Final Project) and approved in its final form by the Undergraduate Course in Control and Automation Engineering

Florianópolis, August 29, 2022.

Prof. Hector Bessa Silveira, Dr. Eng
Course Coordinator

Examining Board:

Prof. Rodolfo César Costa Flesch, Dr. Eng
Advisor
DAS/UFSC

Jonas Rachner, M. Sc.
Supervisor
Laboratory for Machine Tools and Production Engineering (WZL) of RWTH Aachen University

Yuri Triska, M. Eng.
Evaluator
PPGEP/UFSC

Prof. Eduardo Camponogara, Dr. Eng
Board President
DAS/UFSC

To my parents: Alessandra and Fabiano.

ACKNOWLEDGEMENTS

I would like to first express my unending gratitude to Prof. Rodolfo César Costa Flesch, who has been one of my biggest and most influential supporters throughout the years. Thank you for the many great opportunities and, specially, thank you for believing in me. I will never forget your readiness to assist me in whatever issues I brought up.

My sincere thanks to Jonas Rachner, who I see as not just my boss, but also as a friend and mentor. Your technical expertise was invaluable not only in the development of this work, but also throughout our 2 years of working together. Thank you for being there on both the good and the tough moments.

To all my friends and colleagues from WZL, specially Amon Göppert, Soeren Muenker, Lea Kaven, and Armin Buckhorst: it was a pleasure to work with all of you. I will be forever grateful for all that I learned from you.

I owe an immense amount of gratitude to my dear flatmates Ardit Latifi and Nahui Höllmann, who were with me through thick and thin. The endless supply of Ibuprofen, the Sunday late night wake-up calls, and all our balcony hangouts will not be forgotten. Thank you for being such great friends, and always being there for me.

To my decade-long friends Artur K. Neto, Gustavo Estácio, Marcelo Kauai, and Rafael A. de Meireles. Thank you for the great years together, and for many more to come. Special thanks to you, Kauai, for always being there for a quick call or gaming session so that I could take my mind off everything that was happening.

To Amanda Machado and Robson Felisberto, you both have been with me since I started university, and continue supporting me to this day. Our rushed study sessions and over-the-weekend group projects are part of my fondest memories. Thanks for showing me that working hard can be this much fun.

My deepest thanks to UFSC and the Department of Automation and Systems for providing me with both the chance and the skills necessary for the two-year-long journey in Germany that led to this work. I would like to specially thank Profs. Hector B. Silveira, Marcelo De Lellis C. de Oliveira, Ricardo J. Rabelo, and Eduardo Camponogara for all the patience and support provided. You have accommodated many of my requests, and helped me work through the difficult times. Thank you.

I cannot even begin to describe the lifelong support that my family has provided me. To my parents Alessandra Maria Maia and Fabiano Manschein, I specially thank you for always motivating me to seek bigger dreams, and never doubting the many decisions I have made to get to where I am today.

Last but not least, thank you Dr. Andreas Lang and all the nurses in W8 Onkologische Praxis Aachen, Luisenhospital Aachen, and Uniklinik RWTH Aachen. You treated me as one of your own, with great care and attention, and gave me a second chance at life. I sincerely thank you.

ABSTRACT

A major challenge for producing companies is the intensifying competition due to an increasing number of entering companies, with product individualization being a leading factor for success. As mass customization and other trends towards higher production volatility become more widespread, increased flexibility of industrial assembly is of high importance for a company's competitiveness, and support systems are a must-have to keep ramp-up times low in these highly-flexible scenarios. This work aims to support decision-makers in quickly solving problems that may arise during runtime of a Line-less Assembly System (LAS), and in shortening production ramp-up with a brownfield approach. This is achieved by modeling a system that uses assembly system state data from a digital twin to propose new potential assembly system configurations through an optimization model, which are then evaluated based on discrete-event simulation under uncertainties. Simulation results are then used to adjust and rerun the optimization model, forming the optimization loop. In this work, the mentioned system is modeled, including its requirements, data models, and the communication, optimization, and management modules. To verify the system design, the communication module and necessary parsers were implemented, as well as an optimization model for matching product requirements with the assembly system resource capabilities. Results show that the modeled system sufficiently satisfies the system requirements, providing a framework for future studies on production ramp-up of LAS.

Keywords: Decision Support Systems. Ramp-up. Line-less Assembly Systems.

RESUMO

Um grande desafio para as empresas de produção é a intensificação da concorrência devido a um número crescente de empresas que entram no mercado, sendo a individualização do produto um fator principal para o sucesso. Com a personalização em massa e outras tendências para uma maior volatilidade da produção se tornando mais generalizada, a flexibilização da montagem industrial é importante para a competitividade de uma empresa, e os sistemas de suporte são um fator imprescindível para manter baixos os tempos de *ramp-up* nestes cenários altamente flexíveis. Este trabalho visa apoiar os tomadores de decisão na rápida resolução de problemas que possam surgir durante a execução de um Sistema de Montagem sem linha (LAS), e na redução do *ramp-up* de produção com uma abordagem *brownfield*. Isto é conseguido através da modelagem de um sistema que utiliza dados de estado do sistema de montagem oriundos do seu gêmeo digital para propor novas potenciais configurações do sistema de montagem através de um modelo de otimização, sendo, então, avaliadas em uma simulação a eventos discretos sob incertezas. Os resultados da simulação são, então, utilizados para ajustar e reexecutar o modelo de otimização, formando o laço de otimização. Neste trabalho, esse sistema é modelado, incluindo seus requisitos, modelos de dados, e os módulos de comunicação, otimização e gerenciamento. Para verificar a modelagem do sistema, o módulo de comunicação e os *parsers* necessários foram implementados, além de um modelo de otimização para parear os requisitos de produtos com as capacidades de recursos do sistema de montagem. Os resultados mostram que o sistema modelado satisfaz suficientemente os requisitos do sistema, fornecendo uma estrutura para futuros estudos sobre o *ramp-up* de produção de um LAS.

Keywords: Sistemas de Apoio à Decisão. Ramp-up. Line-less Assembly Systems.

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

AGV	Automated Guided Vehicle
API	Application Programming Interface
AS-DM	Assembly System Description Model
DOI	Digital Object Identifier
DSS	Decision Support System
DT	Digital Twin
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LAS	Line-less Assembly System
MQTT	Message Queuing Telemetry Transport
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
ReA-D	ReAssign Data
ReAssign	Resource Assigner
REST	Representational State Transfer
SA-I	Scenario Analysys Input
TDD	Test-driven Development
UML	Unified Modeling Language

CONTENTS

1 INTRODUCTION

Constantly changing markets place increased demands on the adaptability of output in terms of quantity and type of products from manufacturing companies (WIENDAHL; GERST; KEUNECKE, 2004). Economic (e.g., increased number of variants), organizational (e.g., integration of new processes), and technological factors (e.g., increased development speeds) are of central importance for the competitiveness of existing production and assembly systems. In particular, increased flexibility of industrial assembly is of high importance for a company's competitiveness (KLETTI, 2015), as it is responsible for building product and part variants, and new methodologies and support systems are a must-have to keep ramp-up and integration times under control in these highly-flexible scenarios (KUHN et al., 2002). In order to meet the new demands of adaptability in companies, rigid boundary conditions in production and assembly are increasingly being relaxed (WIENDAHL; GERST; KEUNECKE, 2004).

The approach of Line-less Assembly Systems (LASs) implements the required adaptability by enabling individual product routes. This paradigm shift in assembly system design allows for a mapping of variant specific processes without efficiency losses and reconfiguration without interrupting production (HÜTTEMANN et al., 2017).

The continuous integration of new products with different requirements (henceforth called variants) require resources (e.g., robots for manufacturing) with a variety of capabilities (e.g., screwing, welding), and ramp-ups happen frequently due to the system requiring reconfiguration in a brownfield approach. This reconfiguration is currently done manually, which is inefficient and not aligned with Industry 4.0 ideals (IVANOV et al., 2021). A shift towards a data-driven approach is suggested, with a data-model providing enough information to a Decision Support System (DSS) capable of matching the new requirements to the existing assembly system capabilities.

Hereby named Resource Assigner (ReAssign), this supporting software module suggests potential new assembly system configurations capable of satisfying product requirements by relying on the flexible adaptability of LAS. In order to implement said adaptability, it must be possible to record current state information while considering external events (e.g., new incoming product orders, breakdowns). The main input variable of ReAssign is the current state of the assembly system provided by its Digital Twin (DT) counterpart. This state includes information about orders, products, stations, resources, and Automated Guided Vehicles (AGVs).

An optimization loop is responsible for finding suitable assembly system configurations. This loop consists of three phases: optimize (solve an optimization model to find potential solutions), simulate (evaluation of potential solutions based on discrete-event simulation), and adjust (adjust the optimization model according to simulation results). The best solutions are fed back to the DT.

Within the scope of this work, the DSS named ReAssign, along with data models and supporting modules for integration with other systems, was modeled. For this purpose, necessary key figures, attributes, and parameters were determined, which represent the configuration of the assembly system at any time of the operational business. The model was verified through the implementation and testing of 3 software modules.

1.1 SYSTEM PLANNING “AS-IS”

The planning of a LAS shop floor configuration in terms of resource placement is currently done manually by an employee, who will be referred to as “System Planner”. Whenever there is a new order for product manufacturing, or there is a machine breakdown, the System Planner must execute the following tasks:

- analyze the current state of the system in terms of resource status, capabilities, and bottlenecks;
- verify that the current assembly system configuration is capable of manufacturing the desired products by “matching” product requirements with resource capabilities (e.g., a product might have a process step that requires the capability “welding”);
- generate a new system configuration if any capabilities are missing, or if there are not enough resources to manufacture the products in time;
- simulate the solution under uncertainties (e.g., random breakdowns, delays, etc.) in the Simulation Module;
- analyze the results from the simulation and decide if they are satisfactory.

These tasks get increasingly more time-consuming as the number of product variants increases. The first problem is in matching all the requirements to the available capabilities in a brownfield approach, keeping both costs and implementation efforts low. For this case, implementation effort is defined as the effort to either relocate an existing resource or implement a new one into the assembly system.

The second problem encompasses generating a new system configuration based on the chosen resources. Questions such as “how many stations are needed?”, “where does resource X go?”, and “are these changes enough to maintain product throughput?” quickly arise, and it is not unheard of to use the trial-and-error approach until a satisfactory solution is found.

Finally, there are problems related to the Simulation Module. The generated system configurations must be manually mapped to a file in a specific format to be accepted as input. After the simulation is done, the results must be manually analyzed

to determine if the configuration is good enough, or if the whole process must be done again. All of this is done in the user's (i.e., System Planner's) local computer, as such simulation speed is determined by the local computer processing power.

1.2 OBJECTIVES

The aim of this work is to model a DSS and all the required data models and software modules for integration with already existing systems. The model is considered sufficiently verified through the implementation of 3 software modules. Furthermore, the following research question is the target of this work:

"How can a model-based decision support system for the station capability configuration of line-less assembly systems during production ramp-up be designed?"

Specifically, the following subtasks are to be completed:

- familiarization with the state of the art of planning and control of LASs;
- development of a DT data model for mapping and communicating the system state of a LAS;
- modeling of the proposed DSS;
- modeling of necessary data models and software modules for integration with existing systems;
- verification of the model through implementation of 3 software modules;
- documentation of the work.

These subtasks were formulated based on the objectives of this work, the research question, and on discussions with the project stakeholders The following section briefly introduces said stakeholders.

1.3 STAKEHOLDERS

This work was produced in cooperation with the Laboratory for Machine Tools and Production Engineering (WZL) at RWTH Aachen University, Germany, specifically in the Model-based Systems department of the Chair of Production Metrology and Quality Management (`https://www.wzl.rwth-aachen.de`).

The Laboratory for Machine Tools and Production Engineering (WZL) at RWTH Aachen University, with over 1000 employees, researches and develops innovative production technology in the fields of factory planning, assembly planning and technologies, production metrology, quality and production management, among others. The Chair

of Production Metrology and Quality Management, under the direction of Prof. Dr.-Ing. Robert Schmitt, researches questions of flexible assembly organization in numerous projects on metrology-supported assembly processes and in the customer-oriented evaluation of sustainable products. The institute is also a partner in the Internet of Production cluster of excellence (WZL, 2022a).

Guided by the paradigm “Automation of Automation”, the Model-based Assembly Automation group researches the paradigms of future assembly systems. Research topics of the group are, for example, novel control systems for line-free assembly, as well as automation and communication technologies (e.g., reinforcement learning, 5G) for the use of mobile robots and manipulators in assembly (WZL, 2022b).

1.4 DOCUMENT STRUCTURE

This work is structured in such a way that the reader is first introduced to important theoretical concepts used throughout the chapters, then presented with discussions about related literature, followed up by the design and implementation of the system. The document ends with a presentation of the results, an overall conclusion of the work, and a discussion of an outlook for future works. Specifically, the chapters content is as follows:

- in chapter 2, theoretical concepts used throughout this work are presented. LASs, ramp-up phase, DSSs, and DTs are discussed;
- in chapter 3, after the research methodology used in this work is presented, the state of the art in DSSs for production ramp-up and research question related topics is reviewed. The chapter ends with a brief discussion of the research deficit;
- in chapter 4, the design and implementation of the system is discussed. Requirements, an overview of the proposed solution, information flow, data models, and the optimization loop are documented here;
- in chapter 5, results of this work in terms of the system modeling and implementation are discussed;
- in chapter 6, a summary of this work is done, results are discussed, and an outlook for future works is given.

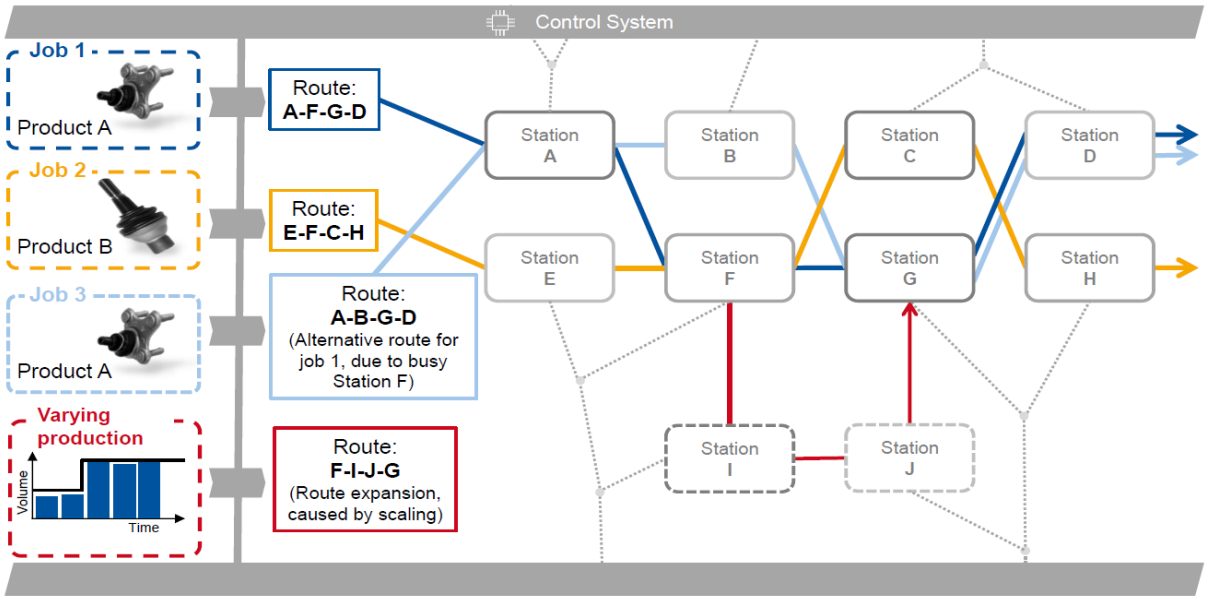
2 THEORETICAL BACKGROUND

In this chapter, theoretical concepts used throughout this work are presented. The concept of LASs and ramp-up phase are discussed, the different types of DSSs are presented, and the chosen definition of DTs used by this work is discussed.

2.1 LINE-LESS ASSEMBLY SYSTEMS

LASs are defined as a form of organization in which each individual product can be assigned its own flexible order route by dispensing with time and space restrictions. By eliminating the flow principle[1], it represents a paradigm shift to lean production, as well as allowing dynamic order routes. These routes are created individually for each order and can be continuously tracked, taking into account the availability of assembly resources (see Figure 1). A control system is used for this purpose, which contributes to maximizing efficiency by means of optimization algorithms (GÖPPERT et al., 2018).

Figure 1 – Dynamic order routes in Line-less Assembly Systems.



Source – Hüttemann et al. (2017).

On the physical level, avoiding a main transport direction requires corresponding transport technology. Mobile transport systems and assembly resources are conceivable for this purpose. Since non-stationary assembly resources are sometimes associated with considerable additional effort, LAS is limited to the use of AGVs. However,

[1] The flow principle has the product as the focus of the entire process, as well as a continuous material flow without unnecessary detours or storage.

there are also approaches with mobile assembly units (HÜTTEMANN; BUCKHORST; SCHMITT, 2019).

LASs are part of the effort to establish increasingly versatile assembly systems. The adaptability of LAS is expressed, among other things, in a flexibility of the orders with regard to resources and routes. Of course, the order routes are limited by design restrictions of the assembly sequence. The more possible assembly sequences are available for a product, the better this is for the utilization of assembly resources, since this results in a plurality of alternatives for order route planning (GÖPPERT et al., 2018).

Flexibility also provides better scalability and mobility in the context of transformation enablement. In case of scaling, no adaptation of the existing resources is necessary. New resources for capacity expansion can be built at any location and connected to the existing assembly system by the mobile transport units. New processes can thus be added, or existing resources can be relieved (GÖPPERT et al., 2018).

2.2 RAMP-UP PHASE

Ramp-up is the phase between completed product development and peak production, where the production system is modified continuously until it reaches peak production (see Figure 2). This transitional phase is characterized by restrictions, changes, modifications, and delays (BASSE; SAUER; SCHMITT, 2014), ranking among the most costly phases of the product life cycle (GLOCK; GROSSE, 2015). It is a highly complex parameter-tuning process with many interrelated factors leading to a well-defined goal.

During this phase, uncertainty is very high, making the process both difficult to manage and unstable[2], expressed in the unpredictability of the system (HANSEN; GRUNOW, 2015). This causes planning proactive steps to avoid some problems to be ineffective for many ramp-up problems (SCHUH; GARTZEN; WAGNER, 2015). As such, ramp-up time is highly dependent on the system complexity and the ability of system integrators and planners to make good decisions and react to unexpected situations (DOLTSINIS et al., 2020).

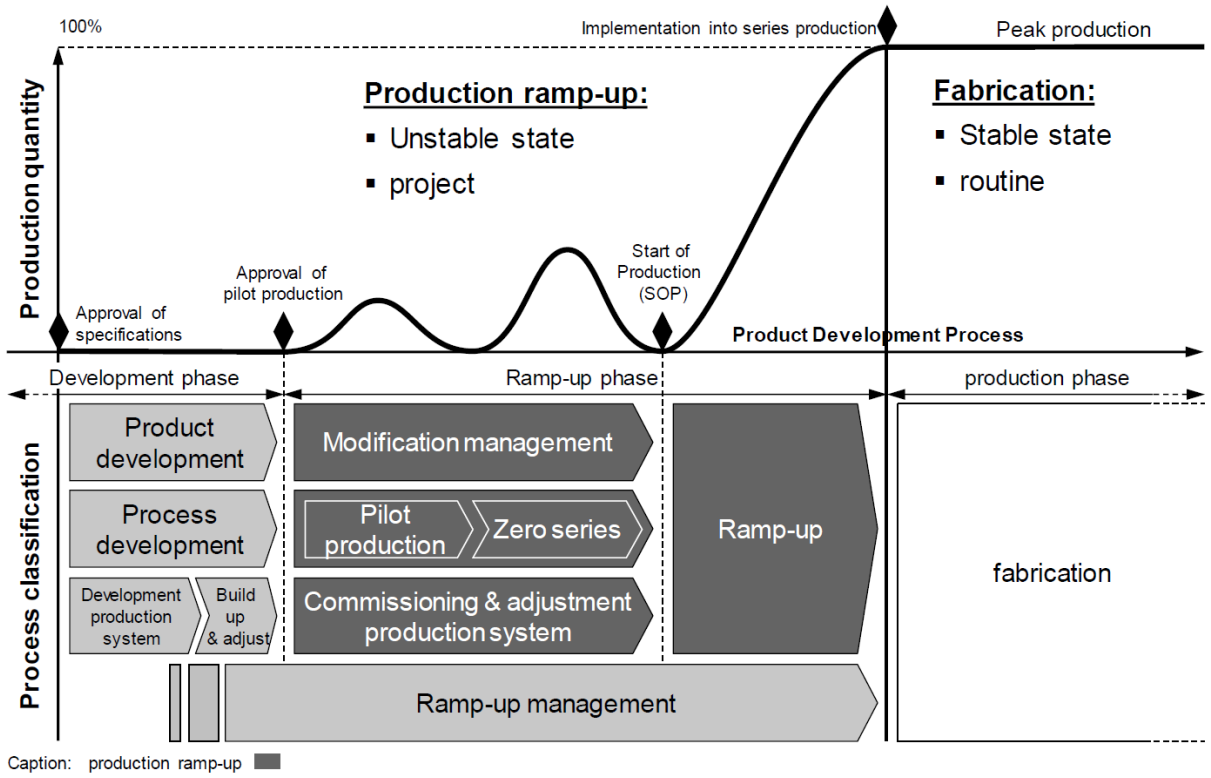
Numerous models and strategies have been developed to support decision-makers during the ramp-up phase, with some characteristics being common across those cases (DOLTSINIS et al., 2020). Of particular interest to this work are existing decision support models in ramp-up, which will be mentioned again in chapter 3.

2.3 DECISION SUPPORT SYSTEMS

Power (2002) defines DSSs as “interactive computer-based systems that help people use computer communications, data, documents, knowledge, and models to solve problems and make decisions” (POWER, 2002, p. 13). Such decisions are neces-

[2] Here, “unstable” means that the process is unpredictable.

Figure 2 – Process organization of production ramp-up.



Source – Dombrowski, Wullbrandt, and Krenkel (2018).

sary within the planning and execution phases of a manufacturing process. Capabilities, layout, and configuration of the system must be decided during planning, while uncertainties are handled within the execution phase.

According to Power (2002), five different types of DSSs can be distinguished: data-based, model-based, document-based, communication-based, and knowledge-based.

1. Data-based DSSs can be used to derive new knowledge or prepare data clearly. Key figures are taken up and processed in such a way that the decision-maker can see the alternative courses of action.

2. Model-based systems use various types of models to help make decisions. For example, they use simulations to estimate the system behavior when executing possible decisions. By formulating optimization problems, a decision to be made can be broken down to the solution of a mathematical problem. Forecasts can be used to make statements about the future.

3. Communication-based DSSs facilitate communication between decision-makers and support joint work on a task.

4. Document-based DSSs support the search, distribution and versioning of unstructured data.

5. Knowledge-based systems are used to provide the user with recommendations for action. They rely on knowledge models such as deterministic rule systems or probabilistic networks.

The trend is shifting towards model-based by integrating simulation models into DSSs, thus allowing it to forecast the system behavior when executing possible decisions. When handling problems during the manufacturing phase, these models require a detailed and current system status, and such representation can be provided to the DSS through the use of DTs (GRAHN et al., 2022). This approach is further discussed along this work.

2.4 DIGITAL TWINS

The concept of a DT varies from one research field to another, as in the last decade researchers from different universities and institutes have been proposing their own definitions. Most of them consider the DT to be a virtual representation capable of interacting with its physical object counterpart throughout its lifecycle. Evaluation, optimization, and prediction are some of the functionalities it provides in relation to the physical object. More recent definitions also add data and services to the DT, enabling the fusion of data from both the physical and virtual aspects for more comprehensive and accurate information capture (TAO; ZHANG; NEE, 2019).

A common understanding of DTs is that of digital counterparts of physical objects. Terms such as Digital Model, Digital Shadow and Digital Twin are often employed as seemingly the same, although they differ in data integration levels between the physical and the digital. According to Kritzinger et al. (2018):

- digital Models do not use any form of automated data exchange between the physical and the digital objects, thus changes in the state of either do not affect the other;

- with Digital Shadows, an automated one-way flow of data exists. Changes in the state of the physical object influence the state of the digital object, but not vice versa;

- finally, when the data flow is bidirectionally integrated between the physical and digital objects, it can be referred to as a Digital Twin. Changes to the state of the digital object influence the physical object and vice versa, thus enabling the digital object as a controlling instance of the physical object.

In this work, the concept of a DT follows that as described by Kritzinger et al. (2018). Further mentions of the term will therefore relate to that of a digital representation with bidirectional data flow and the ability to influence the physical object.

3 LITERATURE REVIEW

3.1 RESEARCH METHODOLOGY

The methodology used is based on the PRISMA2020 Statement, as described in (PAGE et al., 2021). In the “identification” phase, a research question was defined, search terms and key-words were identified, and combinations of those were used for searching documents in a database. Automated exclusion rules were applied here. In the “screening” phase, documents were checked for relevance by reading both their titles and abstracts. The resulting pool must then be assessed for eligibility by having its contents read entirely, whereas irrelevancy to the research focus was considered a reason for exclusion.

Identification

The following research question was defined:

“How can a model-based decision support system for the station capability configuration of line-less assembly systems during production ramp-up be designed?”

From this research question, keywords were identified and grouped together according to their meaning:

- Optimization:
  - operations research;
  - linear programming;
  - mixed integer linear programming.
- Flexibility:
  - matrix;
  - modular;
  - adaptive;
  - flexible;
  - reconfigurable.
- Production:
  - manufacturing;
  - assembly;
  - production.

- Planning:
  - planning;
  - configuration;
  - capability;
  - ramp-up.
- Others:
  - digital twin.

A program was developed to automate the process of generating search strings, searching through the Scopus database, and downloading and filtering the data into an Excel sheet. It is briefly discussed in Appendix A.

The following are 3 of the 360 generated search strings:

- “operations research” AND “matrix” AND “manufacturing” AND “planning” AND “digital twin”;
- “linear programming” AND “flexible” AND “assembly” AND “configuration”
- “mixed integer linear programming” AND “reconfigurable” AND “production” AND “ramp-up”

The search through Scopus resulted in a pool of 1520 documents. From this pool, 825 were duplicates, 24 were conference reviews, and 123 had no Digital Object Identifier (DOI), adding up to 972 excluded documents.

Screening

The following reasons were considered for exclusion of documents:

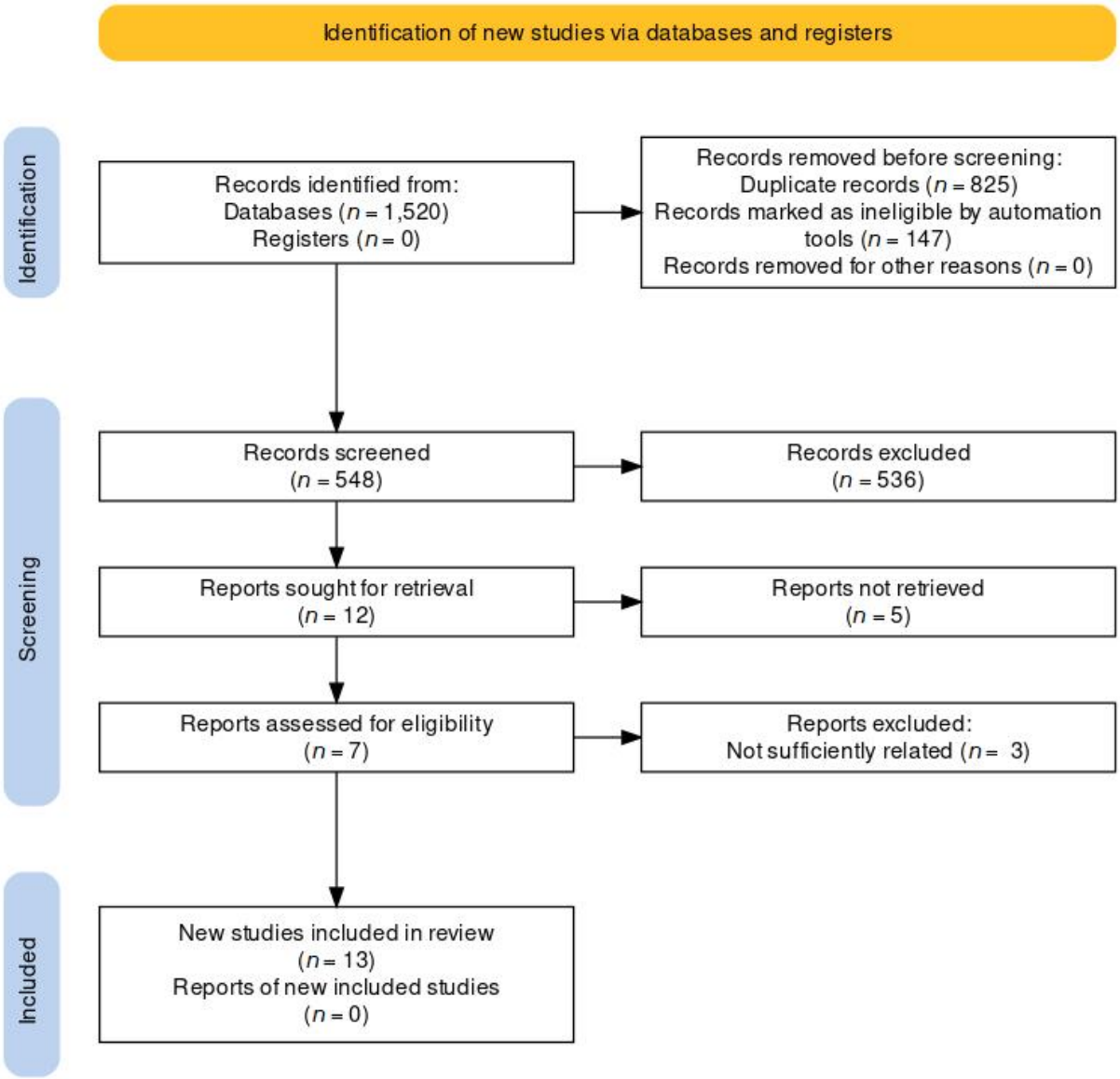
1. not sufficiently related to this work;
2. not retrievable (i.e., not available publicly);
3. duplicates that were not automatically detected and excluded.

Documents focused on line-assembly, non-reconfiguration problems (i.e., scheduling, material handling, etc.), or from unrelated fields (e.g., regional forest planning) were considered as part of the “not sufficiently related” category.

Finally, only 12 documents remained in the pool. Out of those, 5 were not retrievable (required a subscription) and 3 were not eligible due to not being sufficiently related to this work. Only 4 papers remained. A summary of the whole process can be found in Figure 3 (HADDAWAY et al., 2022).

Other 13 documents were also manually included through snowball search or as suggestions from field experts. The following section discusses those that were of most interest to this work.

Figure 3 – PRISMA Flow Diagram.



Source – Author.

3.2 STATE OF THE ART

Glock and Grosse (2015) provide a comprehensive overview of decision support models for production ramp-up that apply mathematical optimization or simulation approaches. Results show that few works developed models for worker assignment, workflow management, imperfect production, interruptions, and price reduction dur-

ing ramp-ups, with uncertainty infrequently being considered in literature. It is noted that employing feature-rich simulation approaches and suitable forecasting models that consider various ramp-up process characteristics can be promising.

Schmitt et al. (2018) apply a mixed-method design based on a quantitative pre-study and qualitative interviews in ramp-up-relevant fields to present research hypotheses and practical implications on developing a novel ramp-up management approach. Results show that although a high uncertainty amongst experts concerning future developments exists, real-time data infrastructure is considered central to support the learning and decision-making process, with Industry 4.0 seen as a major enabler. Nonetheless, transferring decision-making competencies to digital systems is still seen controversially amongst researchers.

Ivanov et al. (2021) provide a focused analysis to examine the state-of-the-art research in Industry 4.0 topics by conducting a large-scale, cross-disciplinary, and global survey amongst researchers in industrial engineering, operations management, operations research, control, and data science. Results reveal a strong focus on descriptive analysis and a lack of predictive and real-time, prescriptive models, calling for multidisciplinary collaborations with engineering, data science, and control disciplines. Furthermore, firms with established technologies for manufacturing visibility and digital control were able to react to disruptions caused by the COVID-19 pandemic more flexibly and responsively, showing that digitalization and resilience is a promising future research avenue.

Göppert et al. (2021) design an end-to-end digital twin pipeline to lower the threshold of creating and deploying digital twins. Of particular interest to this work is the developed description model for digital twins. It consists of submodels that implement a header with their defining information. Furthermore, submodels can either be a parameter, variable, function or component, enabling the depiction of complex systemic relationships and the differentiation into time-dependent characteristics. An example ontology of the manufacturing domain is also presented.

3.3 RESEARCH DEFICIT

Although Industry 4.0 topics are garnering increasingly more attention as research advances, there seems to still be a gap in the application of real-time and predictive models to solve manufacturing problems (IVANOV et al., 2021). When considering the more specific problem of production ramp-up, although many decision support models have been developed to support the decision-making during this phase, there is still the unexplored potential of combining a simulation approach (in particular one that considers uncertainties) with the high data availability of Industry 4.0 advances, such as the DT (GLOCK; GROSSE, 2015; SCHMITT et al., 2018). Furthermore, most works focus on either Flexible Manufacturing Systems or Reconfigurable Manufactur-

ing Systems (SABIONI; DAABOUL; LE DUIGOU, 2022; WIKAREK; SITEK; NIELSEN, 2019; BRUCCOLERI; PASEK; KOREN, 2006), both a level beneath LAS in terms of flexibility.

Therefore, this work seeks to help fill the gap by modeling a system that applies concepts of DSS, DTs, simulations, and optimization to the ramp-up management problem of LAS in the context of mass customization and product co-production. This model is a proposed solution and framework for future works in this research area, and seeks to be the basis of their implementation for evaluation of ramp-up-parameters-based decision-making in LAS.

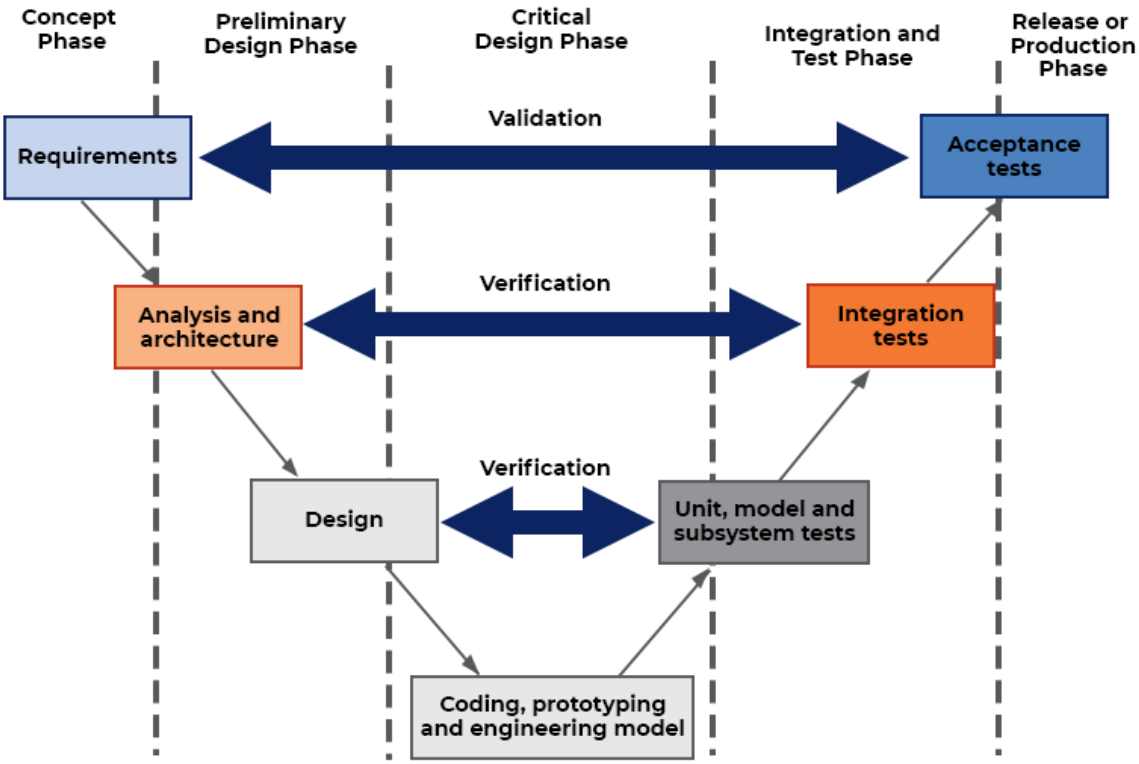
4 DESIGN AND IMPLEMENTATION

In this chapter, the system (nicknamed ReAssign) design and implementation is explained in detail. Development methodology, requirements specification, and the proposed solution are discussed.

4.1 METHODOLOGY

The development of ReAssign followed the V-model development process. It is often viewed as a variant of the waterfall model, which has its life cycle phases progressing in a linear development process, but it differs in the test phase, where development is carried out parallel to the previous phases, thus forming a V shape (as seen in Figure 4). This parallel between phases motivated the employment of Test-driven Development (TDD).

Figure 4 – V-model diagram.

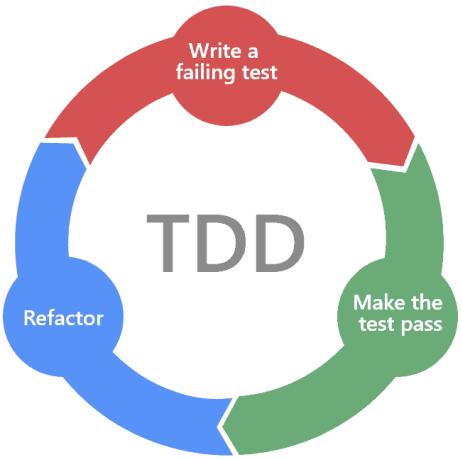


Source – Adam (2021).

TDD is a software development practice in which test code is written prior to production code, thus directing the development of software units. It can reduce the amount

of introduced defects and lead to more maintainable code, with the downside of initial development potentially lasting longer (MAKINEN; MÜNCH, 2014). Figure 5 presents the TDD cycle, where a failing test is written, then made to pass by implementing the missing functionality, which is then refactored to better fit the system.

Figure 5 – Test-driven development cycle.



Source – Marsner Technologies (2022).

For modeling the system, the Unified Modeling Language (UML) notation was selected. It "provides a spectrum of notations for representing different aspects of a system and has been accepted as a standard notation in the industry" (BRUEGGE, 2010, p. 29). The following sections present the system modeling after discussing its requirements.

4.2 REQUIREMENTS SPECIFICATION

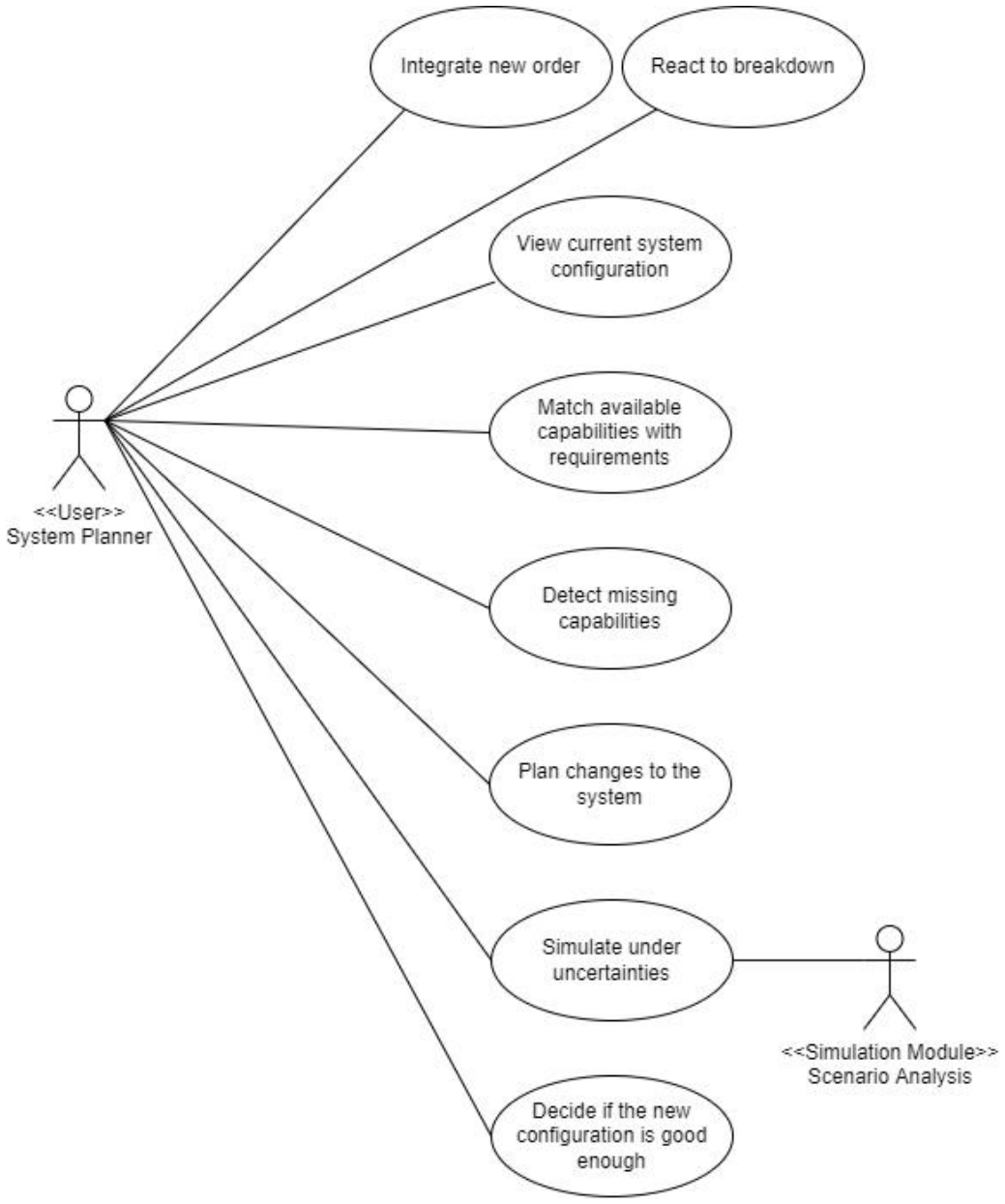
Before detailing the system requirements elicitation and specification, it is important to briefly review the "as-is" process described in section 1.1. The use case diagram in Figure 6 provides an overview of the "as-is" process. These use cases represent the different tasks of the System Planner, and the actors are both the System Planner and the Simulation Module.

The top two use cases, "Integrate new order" and "React to breakdown", encompass the whole process, as they're essentially the reason as to why the other mentioned tasks are executed. They both start the process, and are considered completed only after every other task is executed and a satisfying solution is found. As such, other use cases are simply the steps taken by the system planner to complete the process.

It is worth mentioning that the Simulation Module actor is named "Scenario Analysis" in this specific diagram. This is done to differentiate the simulation module

in the “as-is” diagram from the “to-be” diagram, discussed in section 4.4. In the “as-is” process, the simulation software must be manually executed in the local computer, and does not have the feature of communicating with other systems or software modules.

Figure 6 – Use case diagram for the “As-is” system planning process.



Source – Author.

Based on the problems presented in section 1.1 and multiple interviews with stakeholders, user stories were created as a means to better understand what the system requirements should encompass. Afterwards, functional and non-functional requirements were derived from both the user-stories and the use-case diagram in Figure 6. The created user stories and derived requirements are presented in the

following subsections.

4.2.1 User stories

Written from the perspective of the end user or customer, a user story is an informal, general explanation of a software feature. It uses non-technical language to provide context for the development team as to why and what they are developing, and what value it creates (REHKOPF, 2022). The following user stories encompass the project: as a System Planner, I want...

1. to be able to seamlessly integrate[1] new orders into the assembly system so that products can be co-produced;
2. a planning system that automatically reacts to breakdowns and replans the assembly system configuration by assigning resources to stations so that response time, down-time, and costs are minimized;
3. the planning system to generate optimized assembly system configurations that take into account effort, processing time, and product requirements;
4. the replanning to be done in a brown-field approach, e.g., change as little as possible of the existing configuration;
5. the assembly system, resources, orders, and products to be digitally represented as digital twins;
6. the generated assembly system configurations to be compatible with the existing simulation module so that they can be simulated under uncertainties.

These user stories were used in the formulation of the system requirements, presented in the following subsection.

4.2.2 Requirements

Interactions between the system and its environment, independent of its implementation, are described by functional requirements. The environment includes the user and any other external system with which the system interacts (BRUEGGE, 2010, p. 125). The following functional requirements were identified: the system must...

- (RQ-01) be able to process and integrate new product orders;
- (RQ-02) be able to react to breakdowns by replanning the assembly system configuration;

[1] For this work, “seamlessly integrate” means that the new order (e.g., product variant) is integrated into the assembly system without affecting throughput of already existing variants in production.

- (RQ-03) be able to communicate with the digital twin of the assembly system, and get its current state so that existing resource capabilities and product requirements are identified;
- (RQ-04) be able to detect and identify which capabilities are missing for product manufacturing;
- (RQ-05) be able to generate optimized assembly system configurations that satisfy all product requirements in a way that minimally changes the existing configuration;
- (RQ-06) be able to communicate with the existing simulation module so that generated assembly system configurations can be simulated and evaluated under uncertainties;
- (RQ-07) be able to understand the output of the simulation module so that it can adjust the optimization model when necessary;
- (RQ-08) allow the user to choose which assembly system configuration will be used;
- (RQ-09) be able to automatically choose which assembly system configuration will be used;
- (RQ-10) be able to feed back the chosen assembly system configuration to the digital twin of the assembly system.

Functional requirements specific to the optimization model include: the optimization model must. . .

- (RQ-11) satisfy all product requirements by matching them with available resource capabilities;
- (RQ-12) consider if the given orders can be produced in time or not (i.e., a station can't produce an infinite amount of products in any given time period);
- (RQ-13) consider that a station can only work on one product at a time;
- (RQ-14) output at least 5 assembly system configurations that satisfy the requirements;
- (RQ-15) consider that stations can have up to a maximum number of resources;
- (RQ-16) consider that the assembly system can have up to a maximum number of stations.

Aspects of the system that are not directly related to its functional behavior are described by non-functional requirements. They include a broad variety of requirements that apply to many aspects of the system, from usability to performance (BRUEGGE, 2010, p. 126). The following non-functional requirements were identified:

- (RQ-17) the optimization model must be solvable in less than a minute;
- (RQ-18) the data model of the digital twins must be based on the stakeholder's proprietary data model;
- (RQ-19) the outputted assembly system configuration must be a digital twin of the desired assembly system state.

Based on the presented system requirements, a solution is proposed. An overview of the proposed solution is presented in the following section.

4.3 PROPOSED SOLUTION OVERVIEW

An overview of the proposed solution is depicted in Figure 7. There, four actors can be identified: the assembly system, the DT, ReAssign, and the Simulation Module. The assembly system is mentioned for context, as the DT and all data originated from it will be simulated.

The assembly system provides process data to its DT counterpart, which is capable of influencing the assembly system state by means of "actions". Process data is defined as information about orders, resources, and the assembly system itself. Orders are assumed to also contain information about products. This definition is an adaptation of the ontology developed by Göppert et al. (2021).

Actions are based on decisions taken by the DT in order to influence the state of the assembly system. The goal of these actions is to transition the assembly system state to that of one of the "best suggestions" provided by ReAssign. The interactions between the assembly system and the DT are outside the scope of this work and were mentioned for the sake of providing context, besides being a suggestion for future works.

"Best suggestions" is a group of assembly system configuration suggestions. "Best" is defined by a function of throughput, utilization, and total cost, similar to a leaderboard. These suggestions are the result of a successful optimization loop.

For the optimization loop to be triggered, the assembly system has to be affected by a "problem": either a new order is incoming and must be integrated into the system, or a machine (also called a resource) suffered a breakdown. This information is transmitted by the DT in the form of assembly system state updates.

The optimization loop will then execute an optimization model to find potential system configurations that solve the occurring problem(s). These potential solutions

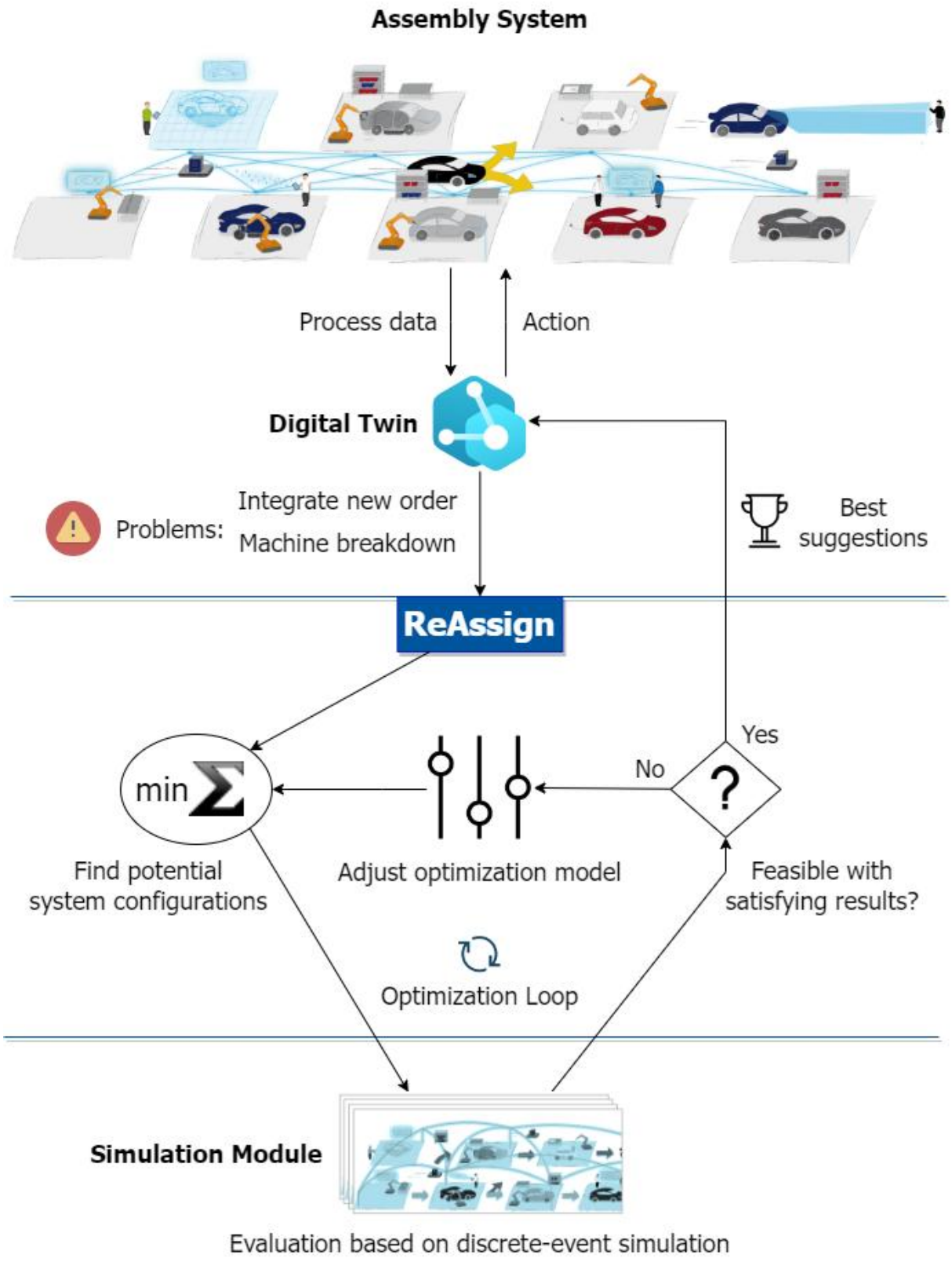
are then sent to the Simulation Module to be simulated under uncertainties. Afterwards, ReAssign will receive a report of the simulations with a variety of statistics, which will be used to determine which (if any) of the solutions is feasible (e.g., if the expected throughput is achieved). In the case of “yes”, there are feasible solutions, they’re saved in memory and, if there are more than five feasible solutions saved, the loop is over and they are sent to the DT. Otherwise, adjustments to the optimization model are applied based on the simulations statistics and the loop repeats itself.

Which solution will effectively be used as the target assembly system state is to be defined in the DT level, with possible options being: manual selection by the System Planner; automatic selection based on the definition of “Best”, as described previously; or another method or rule-based system for selection. In this work, only suggestions of how this selection could be implemented are made. Manual, automated, and even hybrid methods are all possible choices, but the controversy of transferring decision-making to digital systems should be taken into account (SCHMITT et al., 2018).

For the purpose of verifying the system model, the following modules were implemented: optimization, parsing, and communication. The chosen programming language for the implementation was Python 3. The reasons for this choice were: variety of useful and lightweight modules (e.g., paho-mqtt); high code maintainability; and the fact that it is used in many of the stakeholder’s other research projects.

The following sections will go into detail about the system design and implementation of the proposed solution.

Figure 7 – Overview of the proposed solution.



Source – Author.

4.4 USE CASES

The use case diagram of the proposed solution is illustrated in Figure 8. It provides an overview of the system functionalities, as well as a glimpse into how they interact with each other. In this section, the different use cases are discussed.

In Figure 8, three groups of use cases can be identified by their relationships: the top two use cases; the remaining use cases inside the "«Subsystem»" with the addition of "Simulate under uncertainties"; and the singular "Choose a solution" use case. There are also three actors: the System Planner; the assembly system DT; and the Simulation Module. In the "to-be" process, the Simulation Module encompasses both the "Scenario Analysis" simulation software and "Scenario API". This Application Programming Interface (API) implements the necessary communication, and is discussed in section 4.6.2.

The first group relates to the communication between ReAssign and the DT for the transmission of assembly system state data. "Get current state of the system" means ReAssign is capable of receiving the current state of the assembly system, and this use case is extended by "Connect to the MQTT Broker" to indicate that Message Queuing Telemetry Transport (MQTT) is used to accomplish this goal. This constrains the implementation of this use case group to the MQTT Protocol, a decision that will be discussed in section 4.6.2. Requirement RQ-03 is resolved by this group.

The second group is related to the optimization loop, except for "Feed back the best results". This use case resolves requirement RQ-10 and represents a successful optimization loop, thus ReAssign communicates back to the DT a group of system configuration suggestions. Use cases "Integrate new order" and "React to breakdown" represent how ReAssign responds to problems in the assembly system, i.e., they are triggers for the optimization loop. Requirements RQ-01 and RQ-02 are the focus of these use cases. "Check for missing capabilities" satisfies requirements RQ-04 and RQ-11. The current state of the assembly system must be inspected such that ReAssign may know if it is capable of producing all required products. This is achieved by matching product requirements with resource capabilities (e.g., a product might require a resource with a welding tool). The following use cases represent the functionalities of the optimization loop:

- "Generate optimized system configurations" represents the optimization model, which outputs potential system configurations. This use case satisfies requirement RQ-05;
- "Request simulation of generated results" represents the communication between ReAssign and the Simulation Module, and satisfies requirement RQ-06;
- "Simulate under uncertainties" is a use case that belongs to the Simulation Module.

Simulation under uncertainties means that unpredictable problems may occur inside the simulation, e.g., breakdowns. This use case is related to RQ-06;

- "Process simulation results" corresponds to RQ-07 and the functionality of determining which (or if) the simulated configurations are feasible with satisfying results;
- "Adjust optimization model constraints" represents ReAssign's ability to adjust the optimization model based on the simulation results. This use case is related to RQ-07.

The consecutive include and extend relationships were modeled to depict the behavior in the optimization loop. Further discussions are documented in section 4.7.

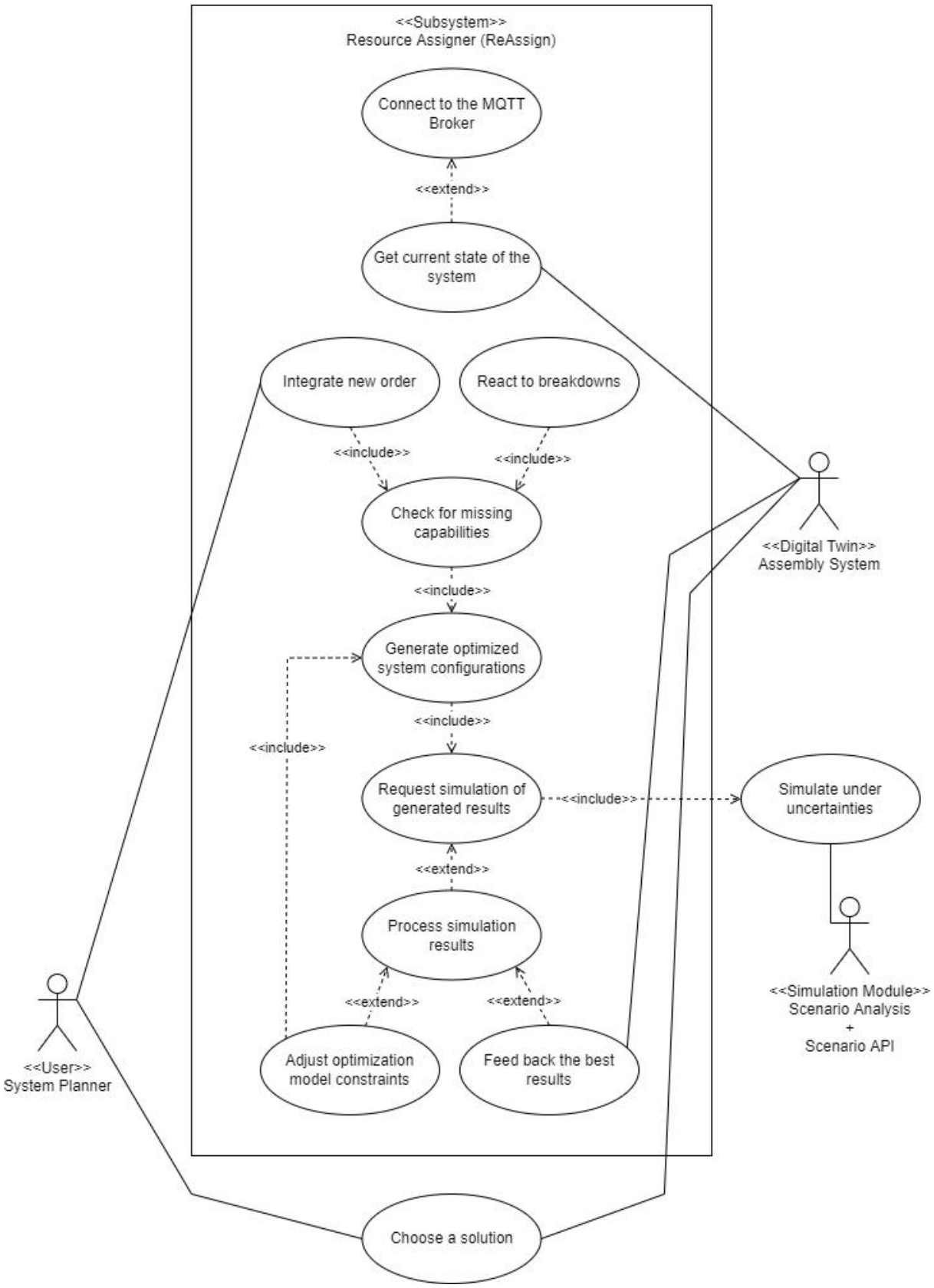
The third group, consisting of only the "Choose a solution" use case, satisfies RQ-08 and RQ-09. It is modeled with the assumption that the DT is configured in such a way that the System Planner can either manually choose a preferred configuration, or set the DT to automatically decide based on a rule or function (e.g., always picks the less costly). Transferring the responsibility of choice from ReAssign to the DT allows better integration potential, since any other systems that communicate with it (or have access to the MQTT topic) would also have access to the solutions.

Table 1 summarizes which use cases satisfy which requirements. These use cases are part of the basis for the design of the class diagram presented in the following section.

Table 1 – Summary of use cases and the requirements they satisfy.

Use case	Requirements
Get current state of the system	RQ-03
Connect to the MQTT Broker	RQ-03
Integrate new order	RQ-01
React to breakdown	RQ-02
Check for missing capabilities	RQ-04, RQ-11
Generate optimized system configurations	RQ-05
Request simulation of generated results	RQ-06
Simulate under uncertainties	RQ-06
Process simulation results	RQ-07
Adjust optimization model constraints	RQ-07
Choose a solution	RQ-08, RQ-09

Figure 8 – Use case diagram of the proposed solution.



Source – Author.

4.5 CLASS DIAGRAM

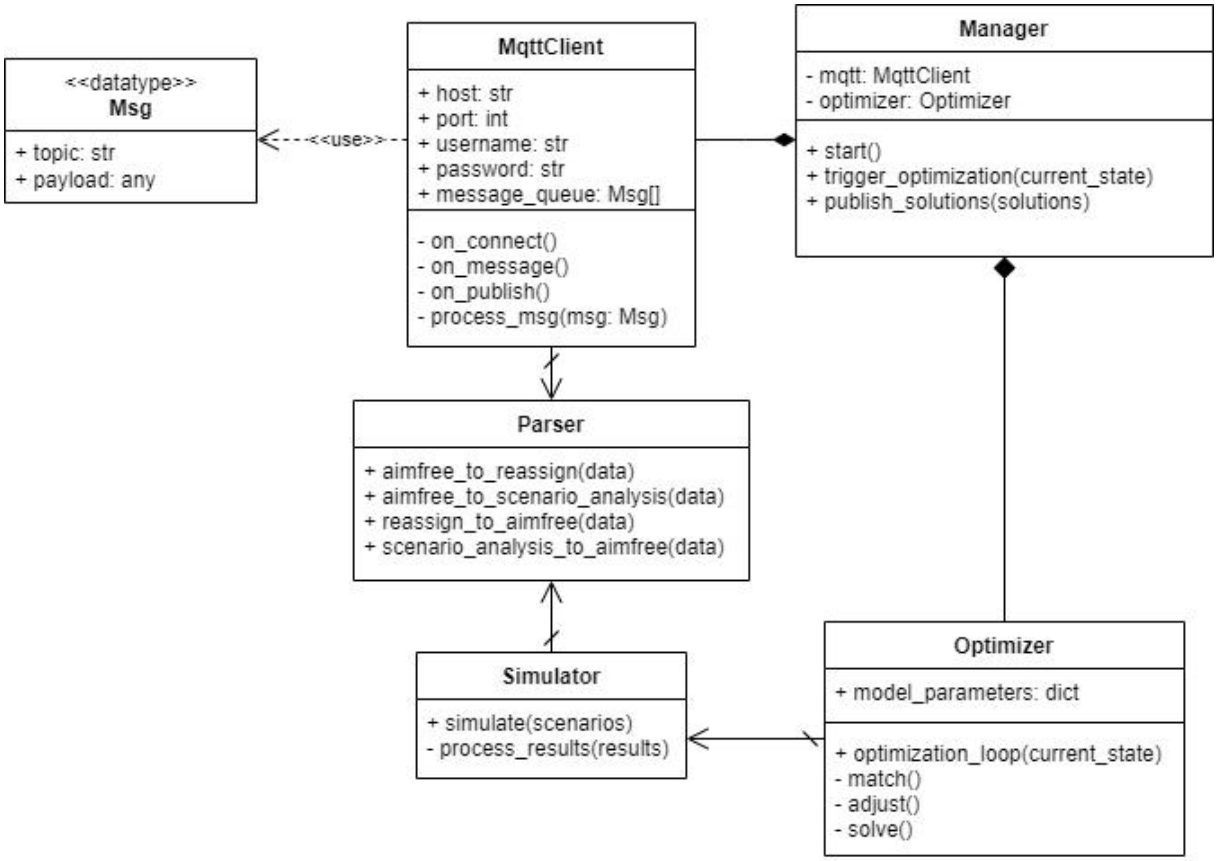
The class diagram in Figure 9 was designed by considering three main functions for the system: communication, optimization, and high-level logic management.

The classes associated with communication are the following:

- MqttClient: responsible for all functionality related to MQTT;
- Simulator: responsible for the integration with the Simulation Module;
- Parser: responsible for parsing data to a compatible format between the different data models.

The Optimizer class is responsible for all the optimization functionality, while the Manager class handles high-level logic management. The design and implementation details are discussed in the following sections.

Figure 9 – Class diagram of the proposed solution.

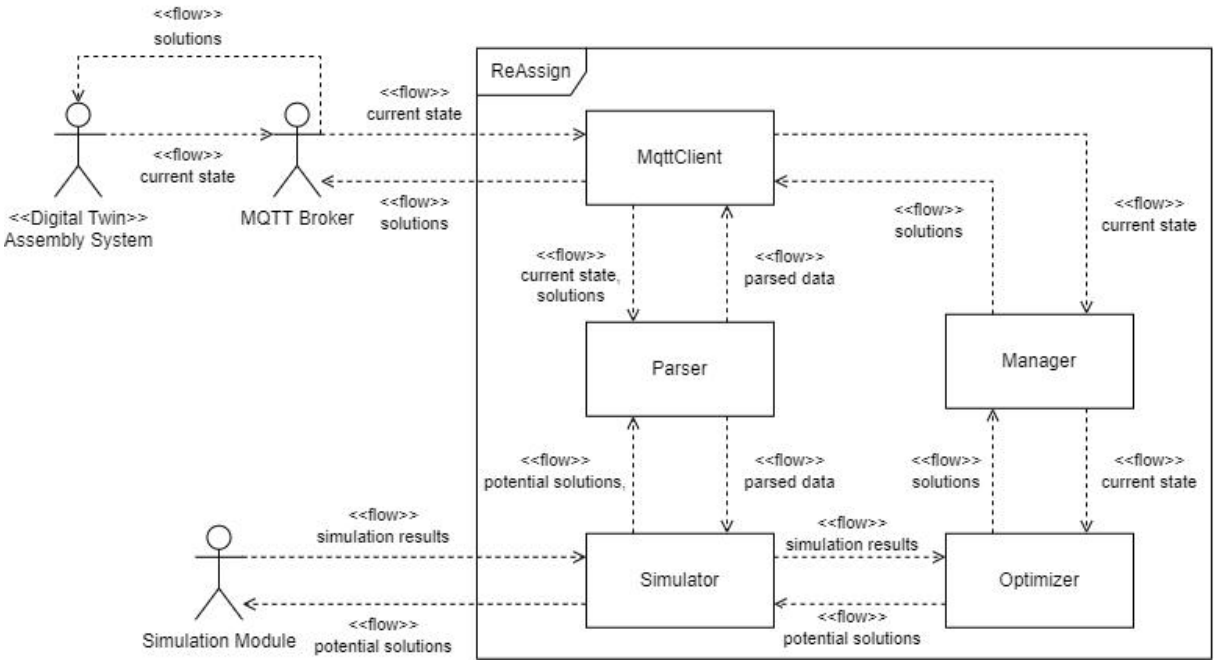


Source – Author.

4.6 INFORMATION FLOW

An overview of information flow is illustrated in Figure 10. This diagram provides a simplified view of the data exchange between actors and modules of the system. Although not explicitly illustrated, there are 3 data models present in the flows, which are highlighted in the following subsection.

Figure 10 – Information flow diagram of the proposed solution.



Source – Author.

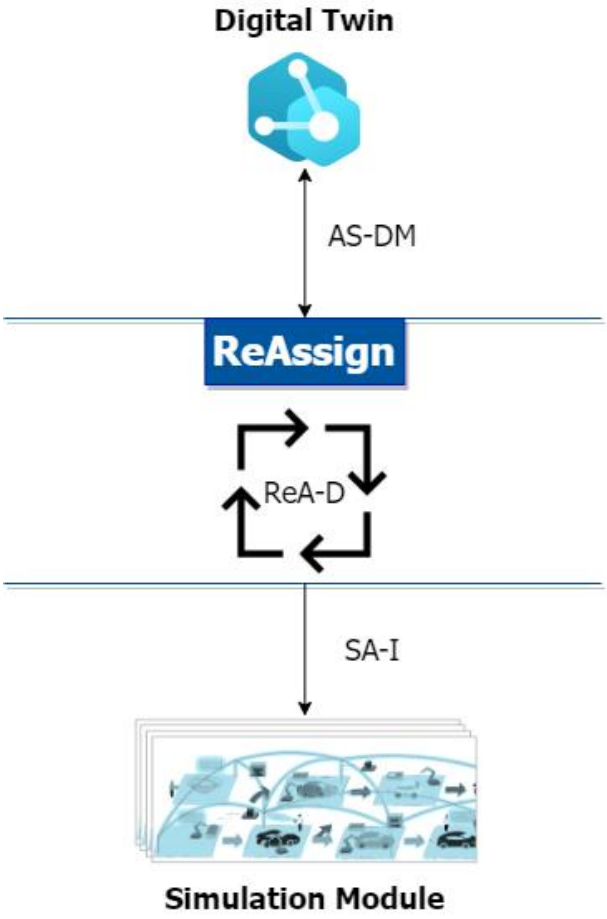
4.6.1 Data models

In this subsection, 3 different data models are presented: the Assembly System Description Model (AS-DM), Scenario Analysys Input (SA-I), and ReAssign Data (ReA-D). Figure 11 depicts an overview of the information flow where these models are applied.

Assembly System Description Model

The AS-DM is applied in the data exchange between the DT and ReAssign. As described by Göppert et al. (2021), its metamodel consists of a header and a body, shown in Figure 12. The header contains defining information about the object, whereas the body contains submodels, which can be of style parameter, variable, function, or component. Components consist of further components, parameters, variables, and functions, thus complex systemic relationship can be depicted.

Figure 11 – Overview of the information flow where the data models are applied.



Source – Author.

Thus, the AS-DM is used to describe the assembly system state in the JavaScript Object Notation (JSON) format, with its components being entities of the assembly system: orders, stations, and AGVs. Figure 13 illustrates the modeling of the assembly system state.

An order is assumed to have a single product. Thus, the “quantity” parameter refers to the amount of products to be produced. Parameters ESD and LFD mean “Earliest Start Date” and “Last Finish Date”, respectively, and are meant to provide information on the amount of time available to manufacture the product. The variable “current lateness” indicates how late its production is. These parameters and variable are available in order to satisfy RQ-12 of the optimization model: it has to know if the order can be completed in time. The state variable indicates if the order is “incoming”, “in progress” or “completed”. Orders with status “incoming” are considered “order integration” problems and trigger the optimization loop.

Orders also have a “Product” component. Products possess the “requirements”

parameter, a list of necessary capabilities for their production. Although Göppert et al. (2021) modeled parts and process steps, which were the components containing requirements, it was decided to simplify the modeling and move this parameter up to the product, as ReAssign does not require to know process steps, only to match requirements with capabilities.

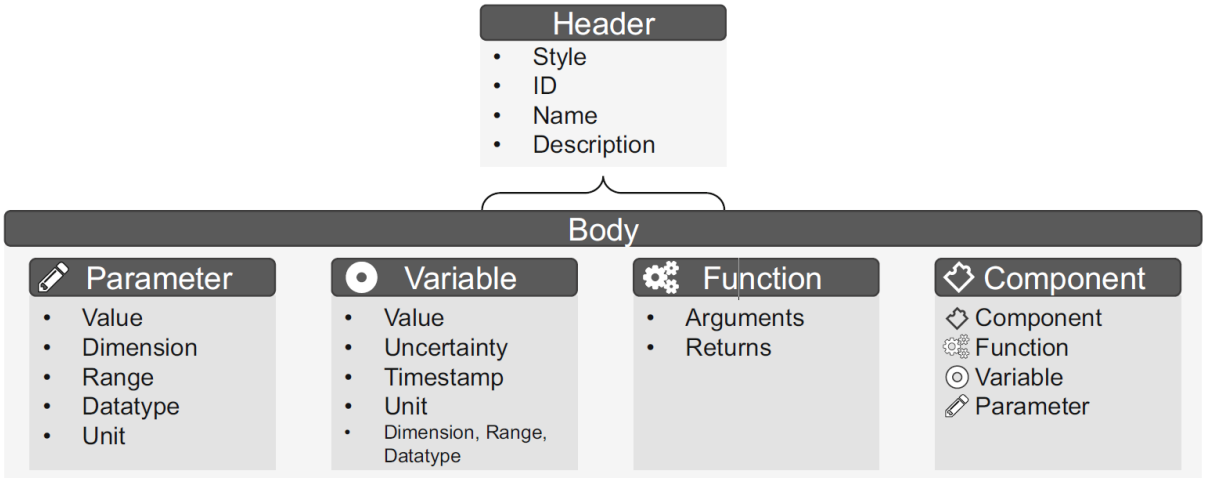
Stations contain the “location” parameter and the “Resource” component. A station location is its xyz coordinates in the assembly system, which can be used to check which cells of the assembly system contain which resources.

Resources contain the “status” variable and the “Capability” component. A resource status can be either “idle”, “in progress”, or “breakdown”. The “breakdown” status on a resource is considered a “breakdown” problem and will trigger the optimization loop.

Capabilities contain an empty body, as such only their headers contain information. Matching with requirements is done by comparing the capability ID with the contents of the product requirements list. If it is contained in the list, a match is made, i.e. the requirement is satisfied. Capabilities were modeled as a component to allow more flexibility in future modifications to the model as the AS-DM is developed, since a capability may be restricted by certain parameters not considered in this work (e.g., a pick-and-place weight limit).

AGVs contain the “status” variable, which is virtually the same as a resource status. They were modeled as their own entity because AGVs have parameters and variables that are configurable in the Simulation Module. Since having the DT contain the parameter configurations for the Simulation Module might be of interest to future works, this component was modeled like so to allow additions to be made more easily.

Figure 12 – Meta-model of the Assembly System Description Model (AS-DM).



Source – Göppert et al. (2021).

Scenario Analysis Input

SA-I is an input format in the JSON format required by the Simulation Module. It contains a variety of simulation parameters to describe layout, product variety, complexity, flexibility of the assembly system and their variation levels (GÖPPERT; RACHNER; SCHMITT, 2020). Those are separated under 3 different categories:

- Structure: defines the basic structure of the scenario, e.g., simulation parameters, algorithms and global layout;
- Product: defines all product information, e.g., requirements, variants, distributions;
- Ressource Assembly: defines all resources information, e.g., capabilities of work stations, and process times.

As the Simulation Module was already developed, SA-I was used as-is, without any further development or adaptation required.

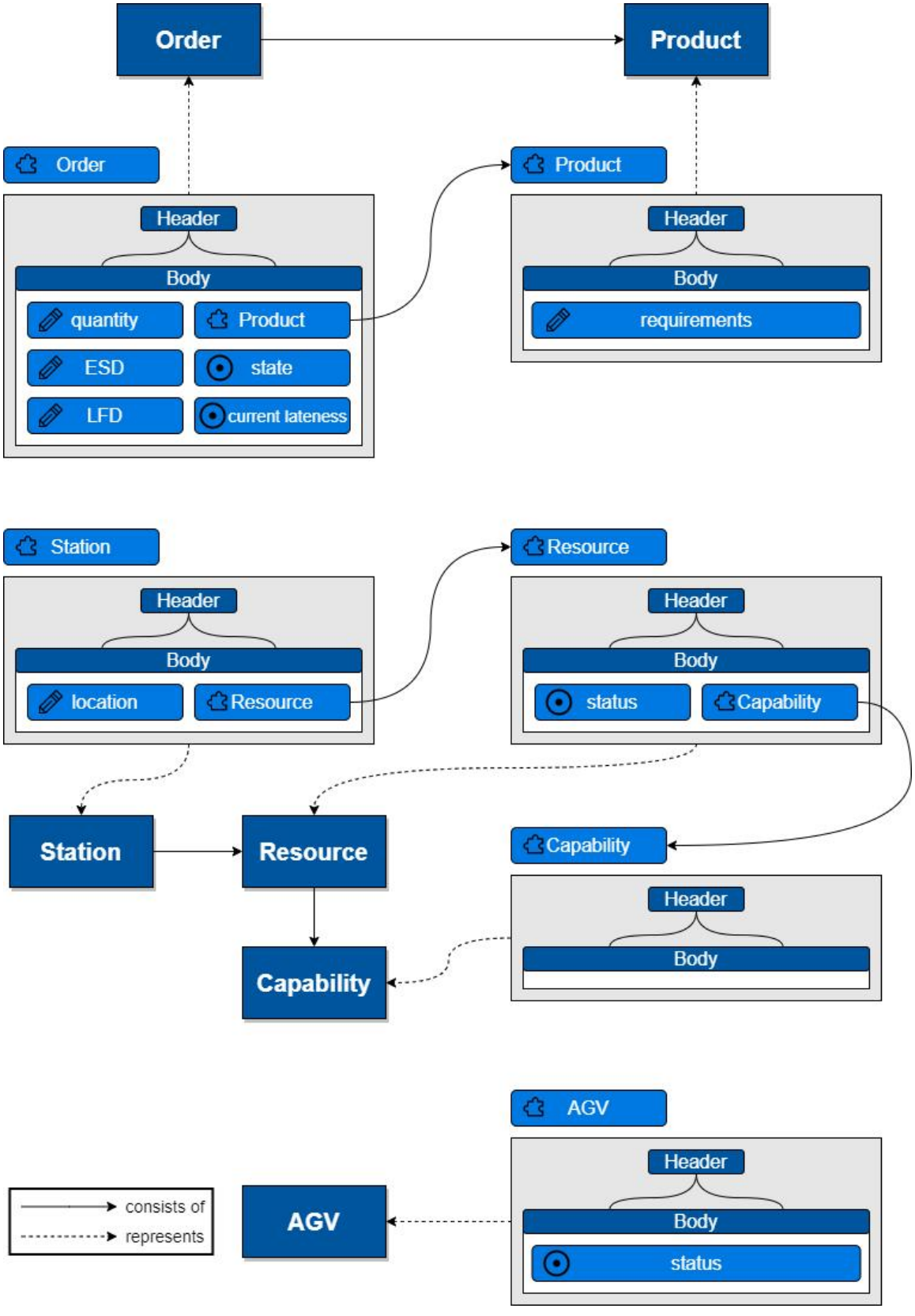
ReAssign Data

To simplify operations inside ReAssign and facilitate both development and use of functionalities, ReA-D was modeled. Its main goal is to store data from the assembly system state into objects with attributes that could be easily accessible by ReAssign classes. As such, the modeling is similar to AS-DM. Figure 14 illustrates the ReA-D model.

Some design choices are worth mentioning:

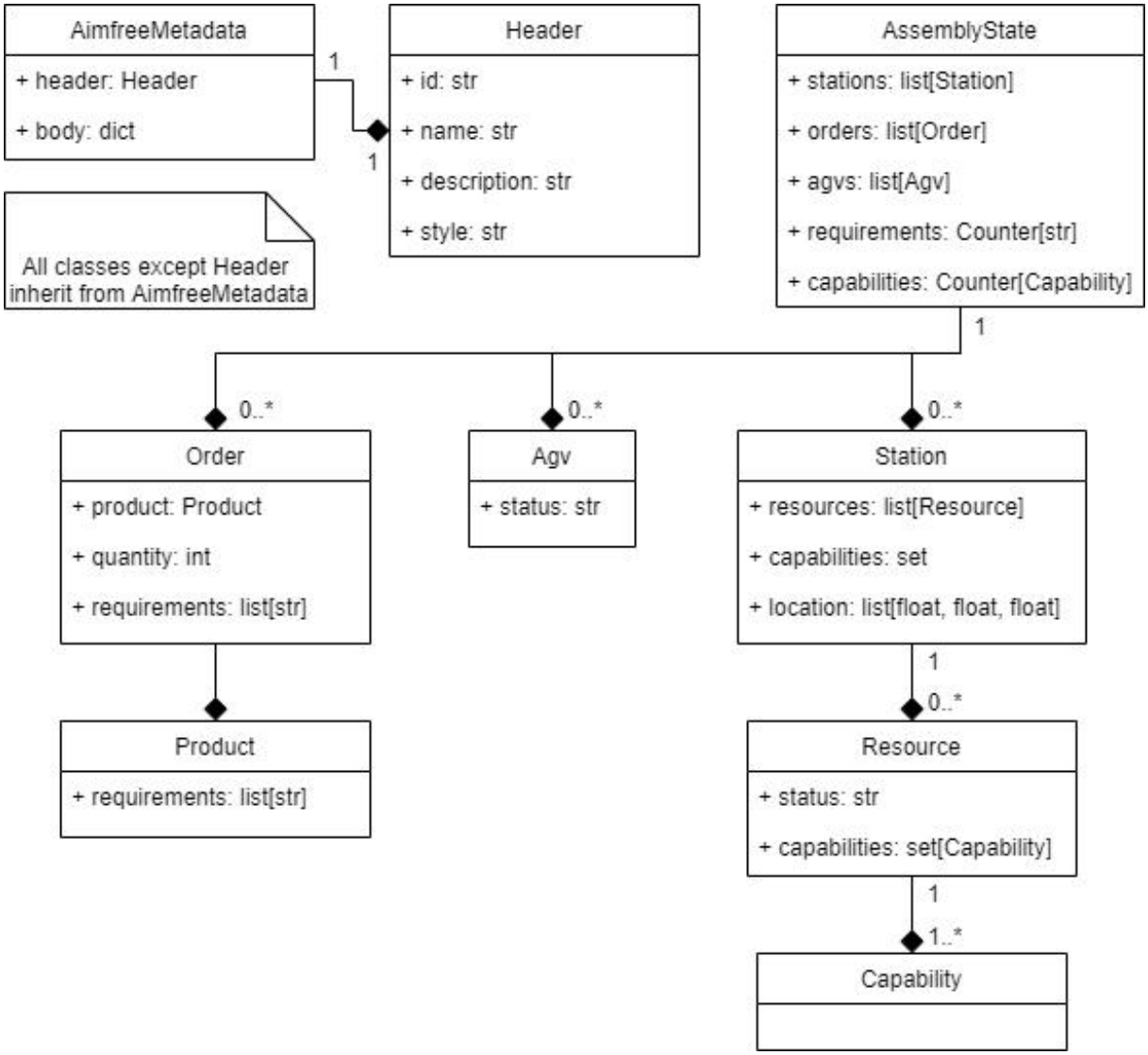
- requirements and capabilities are passed to the parent class upwards to the AssemblyState. This facilitates access to the data;
- capabilities from a Resource are only passed to the parent Station if the Resource has its status attribute different from “breakdown”. As such, Station and AssemblyState capabilities attributes indicate which capabilities the current system has available for use;
- similar to AS-DM, Capability is modeled as a class without attributes (other than those inherited from AimfreeMetadata). For matching with requirements, the id contained in the header is used;
- AimfreeMetadata is inherited by all classes for later use by the Parser class when parsing from ReA-D to AS-DM, so that defining information is persisted.

Figure 13 – Assembly system state modeled from AS-DM.



Source – Adapted from Göppert et al. (2021).

Figure 14 – ReAssign Data (ReA-D) model.



Source – Author.

4.6.2 Communication

There are two communication flows that must be addressed: DT and ReAssign, and Simulation Module and ReAssign. This subsection discusses both flows and the required parsing between data models.

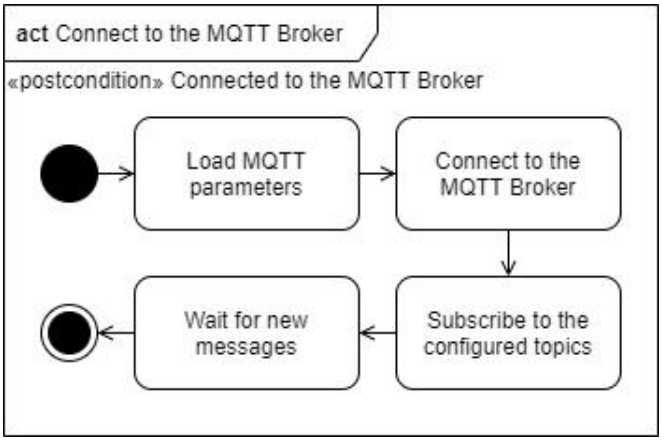
With the Digital Twin

Exchange of data between the DT and ReAssign uses the MQTT protocol. It is a lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth (MQTT.ORG, 2022).

MQTT functionality is implemented by the MqttClient class. Once instantiated, the object sets all connection parameters and callback functions, and then starts its idle loop in a different thread, waiting for messages after a successful connection to the broker. Figure 15 illustrates this by means of an activity diagram.

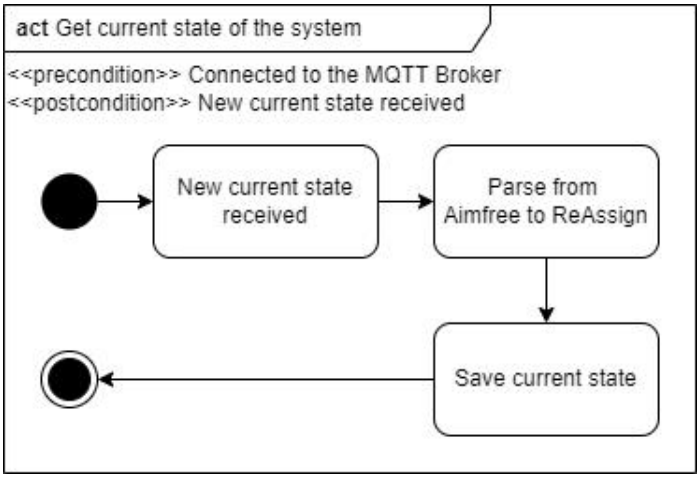
The MqttClient object stays idle until a message is received, i.e., an assembly system state update. Once received, the message is parsed from the AS-DM to the ReA-D model and appended to the message queue, to which the Manager class has access. A summary of this process is depicted in Figure 16. To publish back to the DT, parsing from ReAssign to AS-DM is required, as depicted in Figure 17.

Figure 15 – Activity diagram for the “Connect to the MQTT Broker” use case.



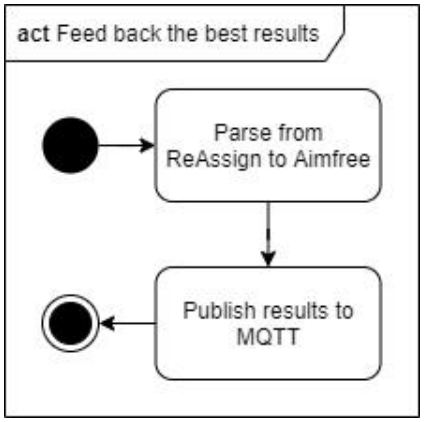
Source – Author.

Figure 16 – Activity diagram for the “Get current state of the system” use case.



Source – Author.

Figure 17 – Activity diagram for the “Feedback the best results” use case.



Source – Author.

With the Simulation Module

The Simulation Module is a two-part module. The first part is called “Scenario Analysis” and contains the already-developed discrete-event simulation. The second part is a Representational State Transfer (REST) API, nicknamed “Scenario API”, which was developed to allow communication with Scenario Analysis by the use of Hypertext Transfer Protocol (HTTP) requests. Figure 18 provides a high-level view of the Simulation Module and its components.

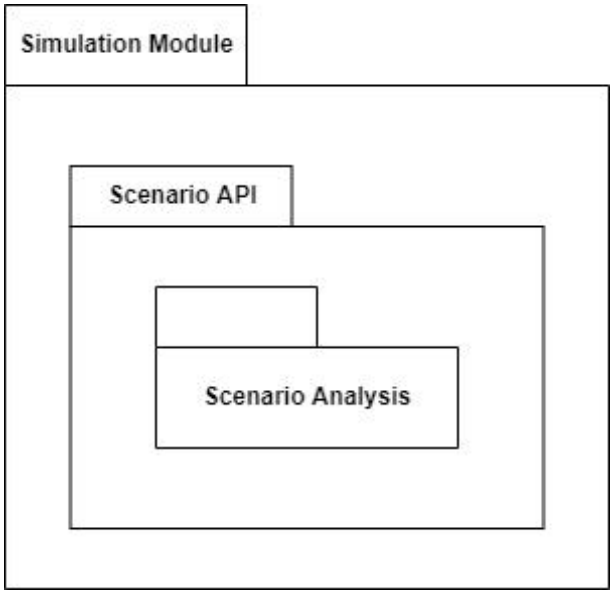
Scenario API allows calls to Scenario Analysis methods and access to the resulting simulation data through its endpoints. The request to create a simulation contains the “scenario_analysis_input”, which must be in the SA-I format. Scenario API is further

detailed in Appendix B.

Figure 19 shows the interaction between ReAssign Simulator class and the Simulation Module, which represents the use case “Request simulation of generated results”. The following sequence of events occur:

1. a request is made to create a simulation instance with the input set as one of the assembly system configurations generated in the optimize phase. This input must be parsed from ReA-D to SA-I;
2. the Simulation Module responds with a 200 OK status code and data about the created simulation;
3. a second request is made to run Scenario Analysis on the created simulation;
4. once the simulation is finished, the Simulation Module responds with a 200 OK status code and a results object.

Figure 18 – High-level view of the Simulation Module and its components.



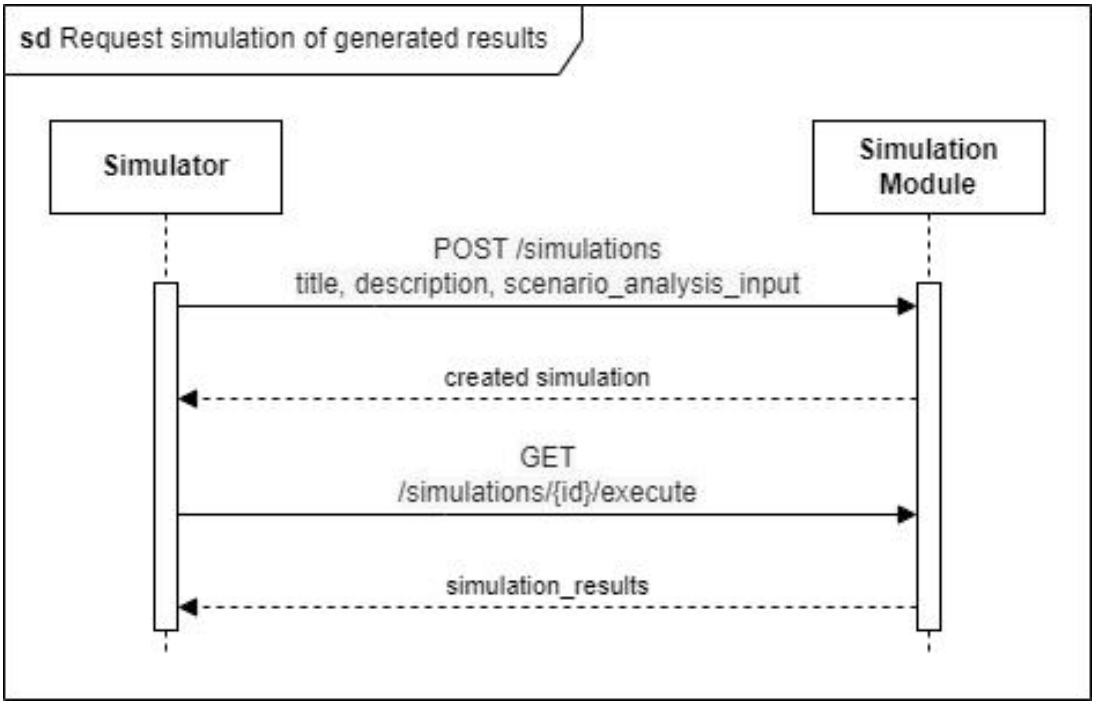
Source – Author.

The communication with the Simulation Module is a part of the optimization loop, specifically the simulate phase. The following section presents the loop and its phases.

4.7 OPTIMIZATION LOOP

The optimization loop is triggered when problems in the assembly system are detected (e.g., machine breakdowns). Product requirements are matched with resource

Figure 19 – Sequence diagram for the interaction between the Simulator class and the Simulation Module.



Source – Author.

capabilities, the optimization model is solved, and thereafter a simulation of the solutions is requested and executed, returning a report of the results that are analyzed to either approve certain solutions, or adjust the optimization model parameters and re-run it. The process from matching to simulation result analysis is referred to as the optimization loop, even though the matching operation happens only once. Figure 20 illustrates the “Check for missing capabilities” (i.e., matching) use case in the form of an activity diagram.

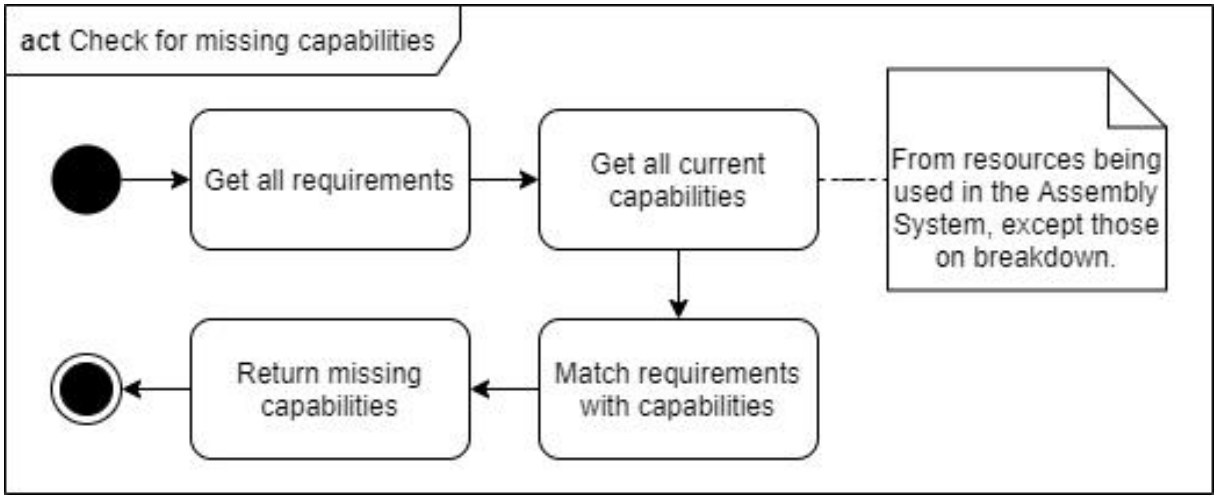
There are three stages to the loop: optimize, simulate, and adjust. Those stages are discussed in the following subsections.

4.7.1 Optimize

In the optimize phase, the optimization model is solved based on a set of constraints and parameters that might change from one loop to another due to the adjust phase. In the first execution of the optimization loop, there are no adjustments to be made. As such, the default model is influenced only by predefined parameters.

The goal of this phase is to return a set of at least 5 potential solutions to the problem, i.e., 5 different system configurations to be simulated in the simulate phase. The model should satisfy the requirements specified in section 4.2. Figure 21 illustrates

Figure 20 – Activity diagram for the “Check for missing capabilities” use case.

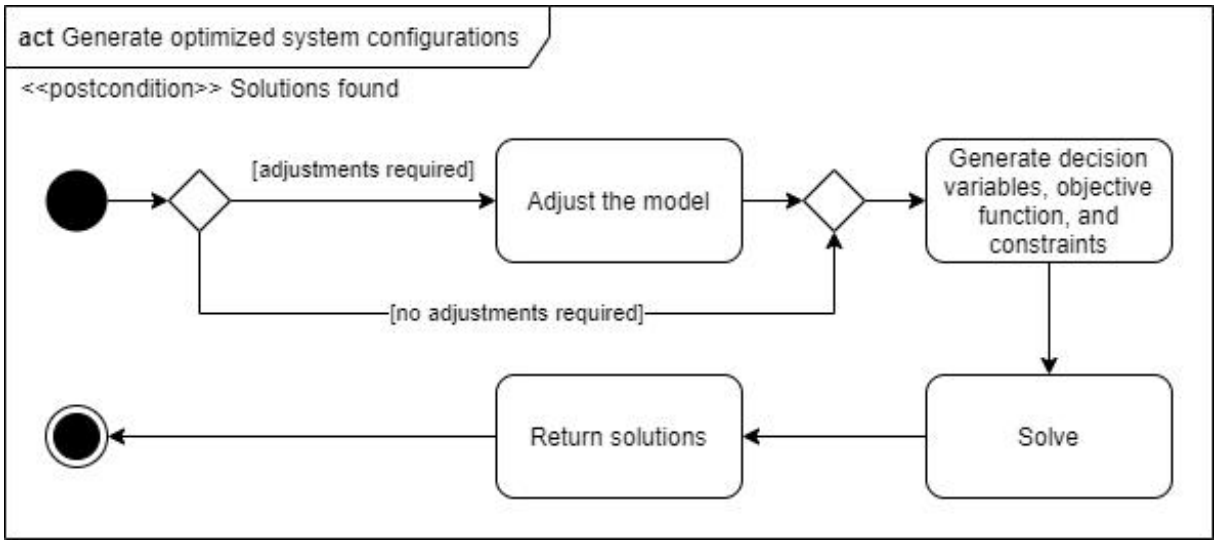


Source – Author.

the activity diagram for the use case “Generate optimized system configurations”.

Gurobipy was chosen as the solver, since it integrates with the Python programming language and is capable of solving linear programming problems efficiently.

Figure 21 – Activity diagram for the “Generate optimized system configurations” use case.



Source – Author.

Model formulation

An initial version of the model was formulated and implemented. It is considered "initial" as it solves only RQ-11 (the matching requirement) of the optimization-specific functional requirements, as well as RQ-17 (solvable in less than a minute) of the non-functional requirements. The model formulation is presented in the following subsections.

Sets

Let R be a set of requirements and C be a set of capabilities of the Assembly System.

Let M_c be a set of currently assigned resources and M_a be a set of resources available for assignment.

Let M_r be a subset of M_c, in which the resources have a capability that matches the requirement $r \in R$.

Let S be a set of stations.

Parameters

- Each station can have up to z resources assigned to it;
- new stations have a fixed cost of q;
- each resource $m \in M_a$ has a set of capabilities C_m and a cost c_m.
- for each requirement $r \in R$ we are given its quantity q_r (i.e., multiple products might have the same requirements);
- each station $s \in S$ has a set of resources assigned to it.

Variables

For each station $s \in S$ and for each resource $m \in M_a$ there will be an associated assignment variable $x_{s,m} \in \{0, 1\}$ describing if resource m is assigned to station s ($x_{s,m} = 1$).

For the station location constraint, there will be a set of station costs which increases in value as the station location becomes farther from the origin point.

Objective function

The objective is to minimize the total cost of assignments. Every assignment will have a cost defined by the sum of the resource and station costs. Equation (1) presents the objective function.

$$\min \sum_{s \in S} \sum_{m \in M_a} x_{s,m}(c_s + c_m) \tag{1}$$

Constraints

Matching constraint: the sum of assignments that have resources with a specific capability must be bigger or equal than the quantity of products with the matching requirement. This constraint is defined by Equation (2)

$$\sum x_{s,m_r} \geq q_r, \forall s \in S, \forall m_r \in M_r, \forall r \in R \tag{2}$$

Complete model

The complete optimization model is presented in Equation (3).

$$\begin{array}{ll} \min & \sum_{s \in S} \sum_{m \in M_a} x_{s,m}(c_s + c_m) \\ \text{s.t} & \sum x_{s,m_r} \geq q_r, \forall s \in S, \forall m_r \in M_r, \forall r \in R \\ & x_{s,m} \in \{0, 1\} \end{array} \tag{3}$$

4.7.2 Simulate

The simulate phase evaluates the potential solutions generated in the optimize phase by simulating the different scenarios under uncertainties (e.g., with machine breakdowns, delays, etc.). Göppert, Rachner, and Schmitt (2020) discuss this already-developed discrete-event simulation.

The Simulation Module is regarded as a black box, in which ReAssign will request a simulation by sending potential solutions as input. As a response, a result file containing a variety of statistics will be returned to ReAssign. The most important statistics for analyzing both feasibility and prospects of the potential solutions are:

- makespan: the time between the arrival of the first order and the completion of the last processing of the last order for a given order backlog. The order backlog corresponds to the given number of orders, which also limits the period of observation of the simulation. Since the simulation is terminated after completion of the last order, the makespan corresponds to the total simulation time;

- lead time: comprises the time between order release and the end of processing of an order. This is measured individually for each order and is determined by the time from the generation of the order to its arrival at the sink[2], which marks the end of processing. At the end of the simulation, the lead time of all orders in the simulation run is averaged. Thus, the arithmetically averaged lead time is used for evaluation. In addition, the lead time is divided into process time, waiting time, and transport time, all of which are also averaged in the same way. They're defined as:
  - process time: the summed time in which an order is in processing stations;
  - waiting time: the time in queues;
  - transport time: the time in which the order is in transport modules.
- utilization (stations): the capacity utilization of a processing station is the quotient of the times during which the station is in the status "in progress"[3] and the total simulation time. The utilization can be determined individually for each station to find out which stations are possibly overloaded or not needed. In addition, the averaged utilization is calculated for all stations;
- utilization (AGVs): the quotient of the time during which the AGVs are in motion and the total simulation time. An AGV can have the states "empty run", "transport" and "waiting". Empty run denotes the situation in which an AGV travels empty to a station to pick up an entity; transport means an AGV transports an entity from the current station to the next; and waiting indicates that the AGV is queued and available, being the only state in which the AGV is considered not in motion and, thus, "not utilized". To evaluate the AGVs, the averaged utilization is considered;
- availability: the proportion of the total simulation time during which a station is not in a breakdown. This refers to the time during which a station is available or in process and, thus, not down. In addition to the availability of all individual stations, the averaged availability for all stations is also calculated. In the simulation model, only technical failures or maintenance measures are taken into account; organizational downtimes are not. Therefore, the availability can be understood as technical availability[4] and depend on the specified values for Mean Time Between Failures (MTBF)[5] and Mean Time To Repair (MTTR)[6];

[2] The sink is an entity in the simulation that destroys incoming entities. It forms the last object in the process and marks an order as finished.

[3] "in progress" means that an order is being processed in this station. Alternatively, a station can have the states "available", "down", "blocked", and "in maintenance".

[4] In technical availability, only the technical failures are taken into account. They are caused by deficiencies in the design or construction of a machine, e.g., material defects, wear and breakage, failure of technical components such as sensors and motors, etc.

[5] MTBF is the average time that elapses between two failures.

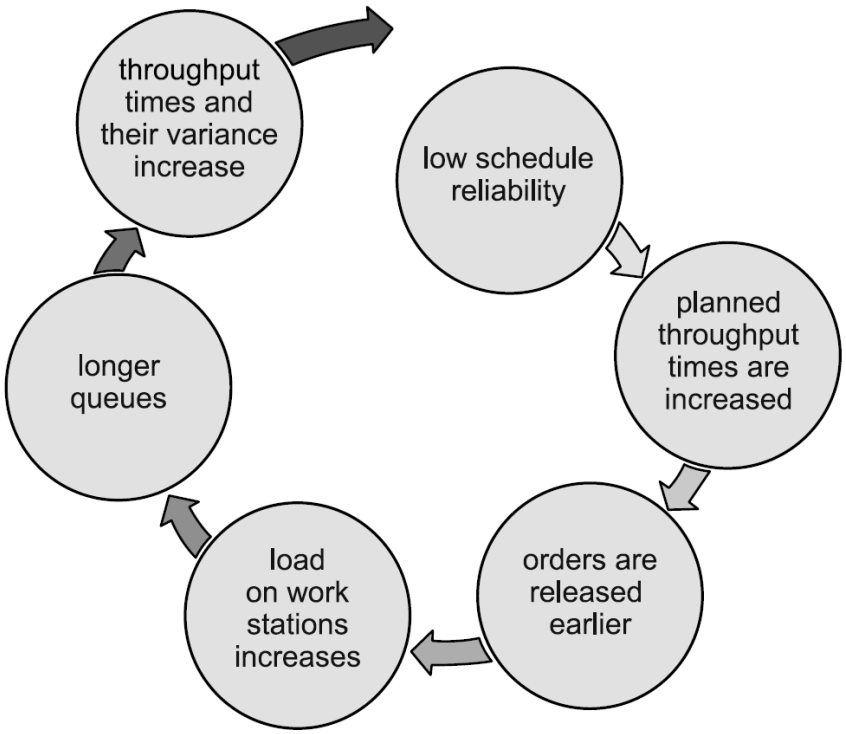
[6] MTTR is the average time needed for repairs.

- queue length: several key figures are retrieved to analyse the queues:
  - average queue length: indicates how many orders were in a queue on average over the entire period under consideration. This value is given individually for the queues of each station;
  - averaged average queue length: averages the average queue length of all stations;
  - maximum queue length: this key figure is given for each station individually and averaged. It indicates the maximum value of the queue length achieved in the period under consideration.
- blocked time: the summed time in which an order is in a station after the process has finished because none of the possible next stations are available. Thus, the order is blocking the station;
- status: indicates if the simulation was successful. The status can be “success”, “deadlock”, or “failed”, of which “deadlock” indicates that there is an unsolvable problem in the scheduling, and “failed” means the simulation failed for unknown reasons;
- throughput: the amount of orders that have finished during a given time (the simulated time).

The aforementioned statistics are to be analyzed in order to decide if the potential solutions are feasible. A potential solution is feasible when it is capable of satisfying throughput and utilization requirements under uncertainties. Although in this work these two key figures are considered the targets for determining feasibility, maximizing just one or the other is not enough. Rather, their impact on the various key figures must be simultaneously considered, as conflicts between objectives may lead to a variety of problems, resulting in uncertainties that often lead to the so-called “vicious cycle of production, planning, and control” depicted in Figure 22 (NYHUIS; WIENDAHL, 2008).

Nonetheless, a possible objective for the selection of feasible solutions is that all orders are completed before their deadlines. An example logic for feasibility can be that the solution is feasible only if the throughput is high enough to complete all orders before their deadline. Throughput can be increased by incrementing the number of available resources and, thus, stations, but this would cause utilization to lower. As such, a second rule could be defined as: “only check throughput if utilization is above X%”, where “X” is the desired minimum utilization. Then again, 100% utilization could indicate a bottleneck. These key figures must be balanced in order to not only avoid the previously mentioned problems, but also to allow the assembly system to efficiently handle the simulated uncertainties.

Figure 22 – Vicious cycle of production, planning, and control.



Source – Nyhuis and Wiendahl (2008).

A potential uncertainty is a machine breakdown, which would cause single-route assembly systems to be blocked while the breakdown is being resolved, thus reducing throughput. A configuration with multiple possible routes for product assembly would handle this issue better, as the assembly system would not be blocked while the issue is resolved. This would result in either a maintained or reduced throughput instead of a complete block. In the context of production ramp-up, this is essential to shorten the ramp-up phase or, in ideal scenarios, to avoid it altogether.

For the problem of product integration, the focus is on modifying the assembly system in a brownfield approach, i.e., changing as little as possible and still achieving production goals. A new order with a new product might have requirements that are already satisfied by the existing assembly system, but a simulation is required to verify if this integration would disturb the system throughput, as shared resources would have higher utilization and possibly wait times. When a simulated product integration results in a maintained throughput, even under uncertainties, it means that the product can be confidently integrated into the system and production ramp-up would be virtually non-existent.

As such, feasible solutions are tagged as such and will eventually be sent to the DT. To support more flexibility in decision-making on the DT side, the simulation

statistics of each feasible solution are to also be sent to the DT. The optimization loop ends when ReAssign finds enough[7] feasible solutions.

If not enough feasible solutions are found, the results are sent to the Optimizer class so it can adjust the optimization model. Thus, the adjust phase of the optimization loop begins.

4.7.3 Adjust

The lack of enough feasible solutions evaluated in the simulate phase triggers the adjust phase. To search for new possible solutions, adjustments to the optimization model are required. Which adjustments are made depends on the values of key figures contained in the results file from the simulate phase, and they might include constraint relaxation, parameter adjustments, etc.

As discussed in subsection 4.7.1, the implemented optimization model only considers the matching requirement RQ-11. Other restrictions (e.g., time restriction in RQ-12; max number of resources per station in RQ-15; or max number of stations in RQ-16) are not implemented, but are of importance to contextualize the adjust phase and guide future works. It is also worth mentioning that the stations hereby mentioned refer to a physical space in the assembly system that could contain resources and in which products are worked on. Thus, adding and removing stations in the physical assembly system is only costly because of the resources associated to it, as well as the space they occupy. This is not the case in the simulation, as the simulated stations have the capabilities and all associated costs. Nonetheless, the analysis presented in this subsection refers to the physical assembly system.

In the case of low throughput, a solution can be the addition of more resources and stations. Although this is a straighforward way of increasing the system throughput, it is highly costly and not aligned with the brownfield approach. Alternatively, it'd be better to analyse utilization and wait times to check for bottlenecks. If the bottleneck is located in a station because multiple product routes must go through it, then an alternative could be to move some of its resources to other stations, distributing the capabilities throughout the system and creating more possible product routes. Transfering a resource from a station to another is less costly than buying and installing a new resource, thus this solution is favored over the previously mentioned one.

For low utilization with a satisfying throughput, there is an indication that the assembly system might have too many stations or resources. This is not necessarily a negative indication: if this situation occurs from evaluating the current assembly system, it means that no changes are necessary for it to successfully produce all orders, thus the best solution is to simply follow the brownfield approach and not change the state. Although this is a good indicator for the current state, it is a negative for suggested

[7] According to RQ-14 defined in section 4.2, five feasible solutions is considered enough.

solutions. A low utilization may indicate that the optimization model is adding more resources and/or stations than necessary, needlessly increasing costs. Zero blocked and waiting times, as well as no queues, are indicators that can help identify the stations and product routes that are unnecessary to maintain throughput.

With the previously mentioned analysis in mind, the Optimizer class can adjust the model accordingly. Optimization parameters such as the max number of stations, or max number of resources, can be relaxed. New contraints can be added to force the addition of new resources, or to keep a specific station from being altered. It is also possible to consider delaying orders by increasing the time constraint to allow a more relaxed order deadline. Such decisions have to consider not only how the different key figures interact with one another, but also how much relaxation is allowed by the System Planner. Thus, although solutions are automatically generated, evaluated, and then sent to the DT, it is important to be precautious regarding the decision of which solution to apply and which actions to take in order to change the assembly system state. This motivates a manual or semi-automatic approach to the decision-making on the DT level, as the System Planner makes the final decision of what is acceptable.

5 RESULTS

In this chapter, results of this work are discussed. Specifically, the system modeling, open points, and the verification through implementation of software modules.

For verification of the model, the following software modules were implemented: communication modules (Simulator and MqttClient), Parser, and a optimization model in the Optimizer class. Furthermore, Scenario API was developed to allow communication with the Simulation Module. These modules were evaluated by unit and integrations tests.

For testing the modules, the following assembly system was assumed:

- resources 1 and 2 with capabilities 1 and 2, respectively;
- stations 1 and 2 with resources 1 and 2, respectively;
- products 1 and 2 with requirements 1 and 2, respectively;
- orders 1 and 2 with products 1 and 2, respectively;
- one AGV.

This assembly system was specified in the AS-DM, ReA-D, and SA-I formats and added as test fixtures. For the Parser unit tests, these fixtures were used in testing the parsing functions from AS-DM to ReA-D and vice-versa. The tests passed if the result of the parsing method was equal to the expected format defined by the fixture.

For the Optimizer class, the test for the match() method asserts that the return value (a Counter object of missing capabilities) contains whichever capability is missing. It also checks if the count is correct. A second test for the match() method was developed specifically for parsed inputs (i.e. assembly state parsed from AS-DM to ReA-D), which does the same checks. The test for the solve() method asserts that the returned solution is equal to what is expected. This test in particular is problematic, since the optimization model is supposed to return multiple different solutions, but for this implementation works because the model always returns the same optimal solution for this specific test scenario. An improved test will be necessary for more complex implementations of the optimization model.

Testing for the Simulator class is done with and without parsed input. In the former, the assembly state is parsed from ReA-D to SA-I before being used as input to the simulate() method. In the latter, the fixture with the scenario analysis input is used as input to the simulate() method. For this test to pass, the Scenario API server must be running, and the response (simulation results) is checked for the “SimulationStatus” parameter. If its value is “Success”, then the simulation was successful and the test passed.

Although no tests were developed for testing the MqttClient class and its methods, it was manually tested with a public MQTT broker and the MQTT Explorer software (NORDQUIST, 2022). It was verified that the MqttClient class is able to connect to the broker, receive messages, and process them according to the rules defined in its methods.

The focus for the verification of the model was in the data models and the communication modules, as they are necessary for the system to operate, i.e. the communication with the DT for getting real-time data, and the communication with the Simulation Module for evaluating generated results. The other software modules can vary greatly in terms of complexity, i.e., the complexity of the algorithms for the optimization phase, or for the processing and decision-making in the adjust phases.

In terms of the objectives for this work, both modeling and verification were achieved. The verification of the communication modules aided in the verification of sufficiency for the defined data models, as they were used in different modules of the system. The system model is thus considered sufficient for the specified requirements.

6 CONCLUSIONS AND OUTLOOK

For the system planning problem of LAS, a proposed solution was modeled. It applies concepts of DSS, DT, simulation, and optimization to aid the System Planner in deciding how the assembly system should be changed in order to achieve sufficient throughput in a mass customization co-production scenario. This is achieved by means of real-time assembly system state data from the DT, which is fed into an optimization loop that generates possible solutions, evaluates them in a discrete-event simulation, and analyses the results to either adjust the optimization model to continue the loop, or feed back feasible solutions to the DT. The model satisfies all of the system functional requirements, and was verified by the implementation of 3 software modules.

The designed software structure is promising for future research in LAS ramp-up management, as it not only employs modern engineering concepts (e.g., DTs, predictive simulation), but also has a modular architecture that permits continual improvements. The Optimizer class, for example, can be replaced by virtually any algorithm that has the assembly state in the ReA-D format as inputs and outputs. Nonetheless, new parsing functions can also be added to the Parser class and used to parse the state into new data models.

Furthermore, a more generic approach to the architecture would see both the optimization and the simulation modules as replaceable. The focus of such an architecture would be to have the DT providing real-time data of the system state, while new states that satisfy whichever objectives were set are searched for by the optimization loop through an algorithm that generates potential solutions (optimize phase), a simulation software capable of predicting the potential solution influence on the system and how it would behave (simulate phase), and another algorithm, model or set of rules to make proper adjustments in the search for solutions.

Implementation was focused on the data models, parsers, and the communication modules in order to provide a baseline for future works. With this, integration with external systems has already been developed, and the focus can be shifted into improving internal modules, such as the Optimizer. Those can vary greatly in terms of complexity and are good starting points for future works.

The implemented modules were verified through unit and integration tests. It is worth mentioning that certain unit tests are difficult to write due to the random nature of, for example, the Simulation Module. The discrete-event simulation applies randomization to the simulated scenarios, as such the resulting statistics vary from one execution to the other. An alternative is to assume repetitive tests and set the random seeds to a constant value just for testing the Simulation Module, making it so the resulting statistics are always the same.

It is suggested that future works focus on research and development of models

and algorithms for the optimize and adjust phases, as they are responsible for suggesting possible solutions. Improvements in these modules will lead to better suggestions and, thus, a more efficient optimization loop in terms of finding feasible solutions. Another possible focus area is the implementation of the DT in terms of its lifecycle, data exchange and interactions with the assembly system, as well as decision-making in terms of deciding which solution should be chosen and which actions should be taken to transition the assembly system to the desired state.

As mentioned in section 3.3, this work and the developed model seek to be the framework for future research in production ramp-up of LAS. With the implementation of previously mentioned suggestions, the research gap in the use of real-time data for solving production ramp-up problems can be reduced. As advances in the area are made and confidence in digital systems decision-making is increased, a consensus amongst researches can be formed, further solidifying the fourth industrial revolution, Industry 4.0.

REFERENCES

ADAM, John. **Was ist der V-modell Ansatz in der Softwareentwicklung und -prüfung?** 2021. Available from: https://kruschecompany.com/de/v-modell-softwareentwicklung/. Visited on: 23 July 2022.

BASSE, Isabel; SAUER, Alexander; SCHMITT, Robert H. Scalable Ramp-up of Hybrid Manufacturing Systems. **Procedia CIRP**, v. 20, p. 1–6, Dec. 2014. DOI: 10.1016/j.procir.2014.05.024.

BRUCCOLERI, Manfredi; PASEK, Zbigniew J.; KOREN, Yoram. Operation management in reconfigurable manufacturing systems: Reconfiguration for error handling. **International Journal of Production Economics**, v. 100, n. 1, p. 87–100, 2006. ISSN 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2004.10.009. Available from: https://www.sciencedirect.com/science/article/pii/S0925527304004190.

BRUEGGE, Bernd. **Object-oriented software engineering : using UML, patterns, and Java**. Boston: Prentice Hall, 2010. ISBN 0136061257.

DOLTSINIS, Stefanos; FERREIRA, Pedro; MABKHOT, Mohammed M.; LOHSE, Niels. A Decision Support System for rapid ramp-up of industry 4.0 enabled production systems. **Computers in Industry**, v. 116, p. 103190, 2020. ISSN 0166-3615. DOI: https://doi.org/10.1016/j.compind.2020.103190. Available from: https://www.sciencedirect.com/science/article/pii/S0166361519306876.

DOMBROWSKI, Uwe; WULLBRANDT, Jonas; KRENKEL, Philipp. "Industrie 4.0 in production ramp-up management". **Procedia Manufacturing**, v. 17, p. 1015–1022, 2018. 28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and Smart Industry for Good of Humanity. ISSN 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2018.10.085. Available from: https://www.sciencedirect.com/science/article/pii/S2351978918312022.

GLOCK, Christoph; GROSSE, Eric. Decision support models for production ramp-up: A systematic literature review. **International Journal of Production Research**, v. 53, p. 1–15, July 2015. DOI: 10.1080/00207543.2015.1064185.

GÖPPERT, Amon; GRAHN, Lea; RACHNER, Jonas; GRUNERT, Dennis; HORT, Simon; SCHMITT, Robert H. Pipeline for ontology-based modeling and automated deployment of digital twins for planning and control of manufacturing systems. **Journal of Intelligent Manufacturing**, Springer Science and Business Media LLC, Nov. 2021. DOI: 10.1007/s10845-021-01860-6.

GÖPPERT, Amon; HÜTTEMANN, Guido; JUNG, Sven; GRUNERT, Dennis; SCHMITT, Robert. Frei verkettete montagesysteme. **Zeitschrift für wirtschaftlichen Fabrikbetrieb**, De Gruyter, v. 113, n. 3, p. 151–155, 2018.

GÖPPERT, Amon; RACHNER, Jonas; SCHMITT, Robert H. Automated scenario analysis of reinforcement learning controlled line-less assembly systems. **Procedia CIRP**, v. 93, p. 1091–1096, 2020. 53rd CIRP Conference on Manufacturing Systems 2020. ISSN 2212-8271. DOI: https://doi.org/10.1016/j.procir.2020.04.116. Available from: https://www.sciencedirect.com/science/article/pii/S2212827120307514.

GRAHN, Lea; RACHNER, Jonas; GÖPPERT, Amon; SAEED, Sazvan; SCHMITT, Robert H. Framework for Potential Analysis by Approximating Line-Less Assembly Systems with AutoML. In: ANDERSEN, Ann-Louise; ANDERSEN, Rasmus; BRUNOE, Thomas Ditlev; LARSEN, Maria Stoettrup Schioenning; NIELSEN, Kjeld; NAPOLEONE, Alessia; KJELDGAARD, Stefan (Eds.). **Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems**. Cham: Springer International Publishing, 2022. P. 423–430.

HADDAWAY, Neal; PAGE, Matthew; PRITCHARD, Chris; MCGUINNESS, Luke. PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis. **Campbell Systematic Reviews**, v. 18, June 2022. DOI: 10.1002/cl2.1230.

HANSEN, Klaus R.N.; GRUNOW, Martin. Modelling ramp-up curves to reflect learning: improving capacity planning in secondary pharmaceutical production. **International Journal of Production Research**, Taylor & Francis, v. 53, n. 18, p. 5399–5417, 2015. DOI: 10.1080/00207543.2014.998788. eprint: https://doi.org/10.1080/00207543.2014.998788. Available from: https://doi.org/10.1080/00207543.2014.998788.

HÜTTEMANN, Guido; BUCKHORST, Armin F.; SCHMITT, Robert H. Modelling and Assessing Line-less Mobile Assembly Systems. **Procedia CIRP**, v. 81, p. 724–729,

2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019. ISSN 2212-8271. DOI: https://doi.org/10.1016/j.procir.2019.03.184. Available from: https://www.sciencedirect.com/science/article/pii/S2212827119304895.

HÜTTEMANN, Guido; GÖPPERT, Amon; LETTMANN, Pascal; SCHMITT, R. Dynamically interconnected assembly systems–concept definition, requirements and applicability analysis. **WGP-Jahreskongress**, v. 7, n. 1, p. 1–25, 2017.

IVANOV, Dmitry; TANG, Christopher S.; DOLGUI, Alexandre; BATTINI, Daria; DAS, Ajay. Researchers' perspectives on Industry 4.0: multi-disciplinary analysis and opportunities for operations management. **International Journal of Production Research**, Taylor & Francis, v. 59, n. 7, p. 2055–2078, 2021. DOI: 10.1080/00207543.2020.1798035. eprint: https://doi.org/10.1080/00207543.2020.1798035. Available from: https://doi.org/10.1080/00207543.2020.1798035.

KLETTI, Jürgen. **MES - Manufacturing Execution System: Moderne Informationstechnologie unterstützt die Wertschöpfung**. [S.l.: s.n.], Jan. 2015. ISBN 978-3-662-46901-9. DOI: 10.1007/978-3-662-46902-6.

KRITZINGER, Werner; KARNER, Matthias; TRAAR, Georg; HENJES, Jan; SIHN, Wilfried. Digital Twin in manufacturing: A categorical literature review and classification. **IFAC-PapersOnLine**, v. 51, n. 11, p. 1016–1022, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. ISSN 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.08.474. Available from: https://www.sciencedirect.com/science/article/pii/S2405896318316021.

KUHN, Axel; WIENDAHL, Hans-Peter; EVERSHEIM, Walter; SCHUH, Günter. Fast ramp up: schneller Produktionsanlauf von Serienprodukten. **Verlag Praxiswissen, Dortmund**, v. 6, 2002.

MAKINEN, Simo; MÜNCH, Jürgen. Effects of Test-Driven Development: A Comparative Analysis of Empirical Studies. In: DOI: 10.1007/978-3-319-03602-1_10.

MARSNER TECHNOLOGIES. **Why Test-Driven Development (TDD)**. Available from: https://marsner.com/blog/why-test-driven-development-tdd/. Visited on: 23 July 2022.

MQTT.ORG. **MQTT: The Standard for IoT Messaging**. 2022. Available from: https://mqtt.org. Visited on: 25 July 2022.

NORDQUIST, Thomas. **MQTT Explorer**. Available from: http://mqtt-explorer.com. Visited on: 8 Aug. 2022.

NYHUIS, Peter; WIENDAHL, Hans-Peter. **Fundamentals of production logistics: theory, tools and applications**. [S.l.]: Springer Science & Business Media, 2008.

PAGE, Matthew J et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. **BMJ**, BMJ Publishing Group Ltd, v. 372, 2021. DOI: 10.1136/bmj.n71. eprint: https://www.bmj.com/content/372/bmj.n71.full.pdf. Available from: https://www.bmj.com/content/372/bmj.n71.

POWER, Daniel J. **Decision support systems: concepts and resources for managers**. [S.l.]: Greenwood Publishing Group, 2002.

REHKOPF, MAX. **User stories with examples and a template**. Available from: https://www.atlassian.com/agile/project-management/user-stories. Visited on: 23 July 2022.

SABIONI, Rachel; DAABOUL, Joanna; LE DUIGOU, Julien. Joint optimization of product configuration and process planning in Reconfigurable Manufacturing Systems. **International Journal of Industrial Engineering and Management**, v. 13, p. 58–75, Mar. 2022. DOI: 10.24867/IJIEM-2022-1-301.

SCHMITT, Robert H.; HEINE, Ina; JIANG, Ruth; GIEDZIELLA, Felix; BASSE, Felix; VOET, Hanno; LU, Stephen. On the future of ramp-up management. **CIRP Journal of Manufacturing Science and Technology**, v. 23, p. 217–225, 2018. ISSN 1755-5817. DOI: https://doi.org/10.1016/j.cirpj.2018.03.001. Available from: https://www.sciencedirect.com/science/article/pii/S1755581718300105.

SCHUH, Günther; GARTZEN, Thomas; WAGNER, Johannes. Complexity-oriented ramp-up of assembly systems. **CIRP Journal of Manufacturing Science and Technology**, v. 10, p. 1–15, 2015. ISSN 1755-5817. DOI: https://doi.org/10.1016/j.cirpj.2015.05.007. Available from: https://www.sciencedirect.com/science/article/pii/S1755581715000334.

TAO, Fei; ZHANG, Meng; NEE, Andrew Yeh Chris. **Digital twin driven smart manufacturing**. [S.l.]: Academic Press, 2019.

WIENDAHL, Hans-Peter; GERST, Detlef; KEUNECKE, Lars. **Variantenbeherrschung in der Montage**. [S.l.]: Springer Berlin Heidelberg, Jan. 2004. ISBN 3540140425. Available from: https://www.ebook.de/de/product/2438765/variantenbeherrschung_in_der_montage.html.

WIKAREK, Jarosław; SITEK, Paweł; NIELSEN, Peter. Model of decision support for the configuration of manufacturing system. **IFAC-PapersOnLine**, v. 52, n. 13, p. 826–831, 2019. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019. ISSN 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.11.232. Available from: https://www.sciencedirect.com/science/article/pii/S2405896319311838.

WZL. **Chair of Production Metrology and Quality Management**. 2022a. Available from: https://www.wzl.rwth-aachen.de/cms/wzl/Forschung/Fertigungsmesstechnik/~suqy/Uebersicht-MQ/lidx/1/. Visited on: 4 Aug. 2022.

WZL. **Model-based Systems**. 2022b. Available from: https://www.wzl.rwth-aachen.de/cms/wzl/Forschung/Fertigungsmesstechnik/~suqz/Model-based-Systems/lidx/1/. Visited on: 4 Aug. 2022.

APPENDIX A – PRISMA AUTOMATOR

This program was developed as a means of automating the initial steps of the PRISMA2020 Statement (PAGE et al., 2021), hence the nickname “Prisma Automator”. Those steps are normally done manually, and the process becomes more repetitive as the number of keyword combinations increases. As such, the goal is to ease the user’s burden by allowing him to skip formulating search strings, searching and downloading document metadata, and filtering the results. Essentially, the user only has to input desired keywords, wait for the program to execute its tasks, and screen the resulting document pool.

Prisma Automator contains two classes: “Splitter” and “Collector”. The Splitter class is responsible for generating all possible search strings (splits) from the defined keyword groups. The Collector class is responsible for interacting with the Scopus API and retrieving results, as well as cleaning them up before saving them to a local directory.

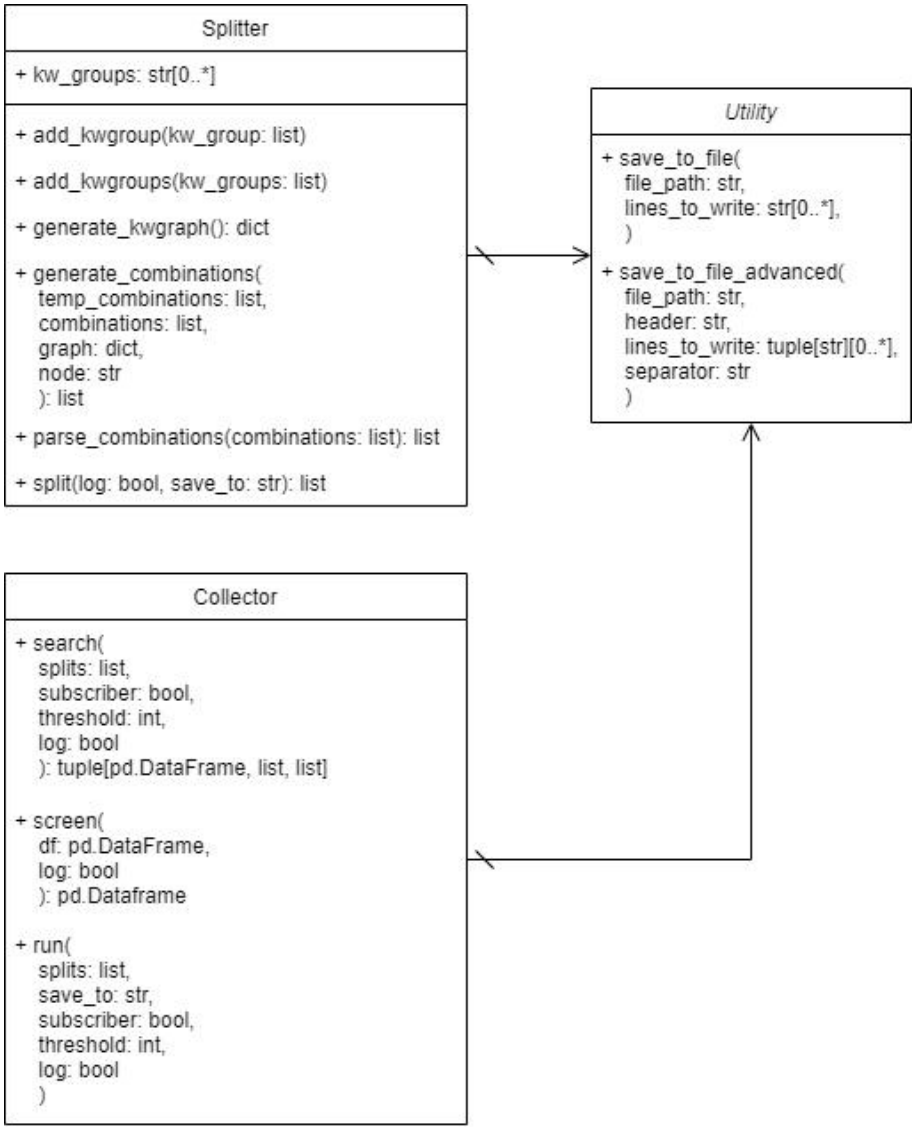
Both classes were implemented with methods to wrap all of their functionality and streamline the process of acquiring search strings and Scopus results: Splitter has the “split()” method; and Collector has the “run()” method. Nonetheless, it is also possible to use the other methods and change the default functionality to suit any particular needs. Figure 23 illustrates the Prisma Automator class diagram.

SPLITTER

The Splitter class uses a recursive depth-first search to find all possible keyword combinations. Before that, it is necessary to generate an adjacency graph to represent the tree. The found combinations are then parsed to generate splits that are searchable in Scopus.

- add_kwgroup() and add_kwgroups(): add keyword groups to the Splitter. These keyword groups are then used in generating combinations and splits;
- generate_kwgraph(): generates an adjacency graph from the added keywords for use in depth-first search;
- generate_combinations(): uses recursive depth-first search to generate all possible keyword combinations;
- parse_combinations(): parses keyword combinations into searchable strings;
- split(): streamlines the split generation process by calling all other methods, as well as saving generated data to the local directory.

Figure 23 – Class diagram for the Prisma Automator.



Source – Author.

COLLECTOR

The Collector class contains 3 methods: search(), screen(), and run().

- search(): takes the generated splits as input and searches Scopus. Results are saved in 3 different objects: a pandas dataframe containing all data from search results (DOI, title, etc.), and two lists containing the number of search results and their associated split;
- screen(): takes the generated dataframe as input and screens it for duplicates, unnecessary columns (e.g. funding data), conference reviews, and rows without a DOI;

- run(): streamlines the whole process by calling ‘search()‘ and ‘screen()‘, as well as saving the generated data to the local directory.

The repository for the Prisma Automator project can be accessed in the following url: `https://github.com/Fabulani/prisma-automator`.

APPENDIX B – SCENARIO API

INTRODUCTION

The Scenario API is responsible for handling communication between Scenario Analysis and ReAssign, as well as any other systems that might need access to its methods and data. It is capable of receiving requests for creating, reading, and executing simulations, as well as saving the resulting data to a database and returning it when requested. A web documentation page is available at `https://documenter.getpostman.com/view/7261554/TzRLmAf4`.

The endpoints used by ReAssign were described in section 4.6.2, and in this chapter the remaining ones are discussed.

OVERVIEW

The main requests of Scenario API are presented in a way similar to the workflow of Scenario Analysis, and can be seen in Figure 24. The following steps are followed:

1. create a simulation;
2. execute the created simulation;
3. access the data generated by Scenario Analysis.

Other important requests are also presented, but request types such as PUT and PATCH have not been included, as they are not important for the Scenario Analysis workflow.

Tests were written for all documented requests. All the requests have a test that checks if the status code is correct, while some have tests for checking the response content and data types.

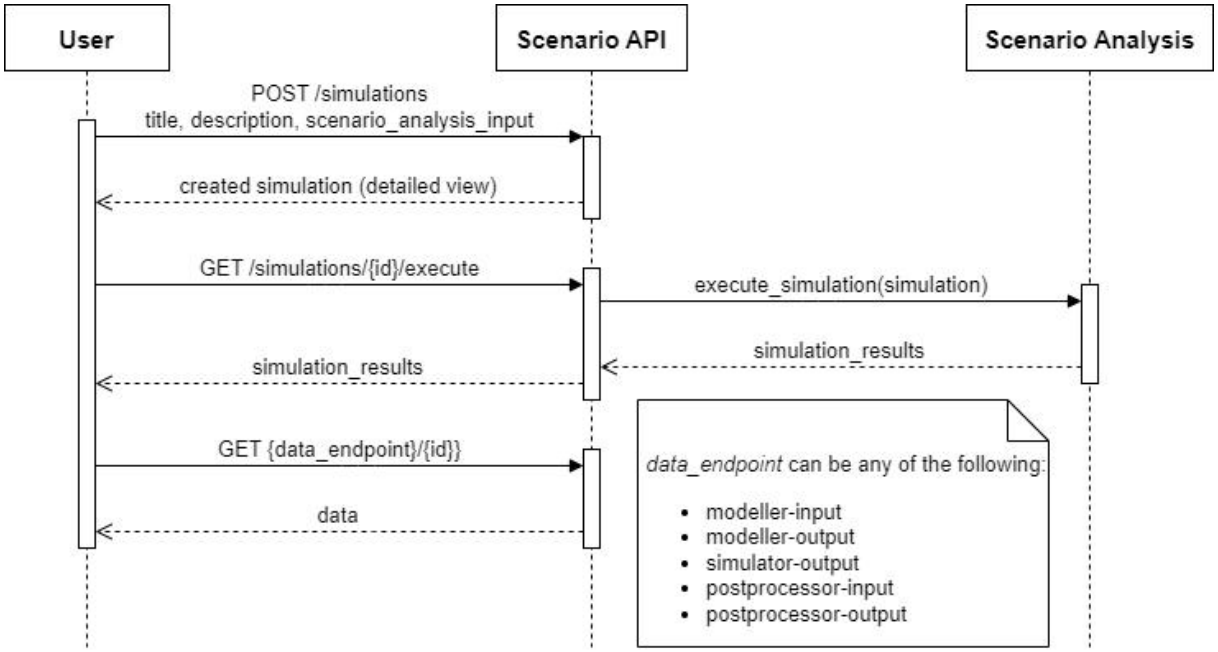
The chosen method of Authentication was *BasicAuthentication*. This requires a “Authorization: Basic” header with a valid username and password combination to be sent in order to access the API endpoints. This method was chosen to shorten development time, but will eventually be replaced with a more secure method.

Pagination was used for list views. The pagination properties are:

- count: total number of results received;
- next: url of the next page;
- previous: url of the previous page.

The urls are null when the number of results is less than the minimum required for a page.

Figure 24 – Usual workflow for Scenario API.



Source – Author.

The user receives a 200 OK status code when the following conditions are met:

1. the request is authenticated;
2. the accessed endpoint is allowed for the authenticated user;
3. the authenticated user is the owner of the data being accessed (e.g. a simulation).

201 Created is returned when a simulation is successfully created through a POST request. Other possible codes are:

- 401 Unauthorized is returned if condition (1.) is not met;
- 403 Forbidden is returned if condition (2.) is not met;
- 404 Not Found is returned if condition (3.) is not met;
- 500 Internal Server Error might be returned if the simulation is executed with a bad input, or if an unknown error happens in Scenario Analysis.

ENDPOINTS

This section presents the available endpoints in Scenario API. For ease of understanding, the different endpoints were grouped into the following:

1. miscellaneous;
2. simulations;
3. integration with Scenario Analysis;
4. simulation data after execution.

Miscellaneous

Some endpoints are only for informational purposes and are presented in this subsection.

The default basic root view of the API is located in the "/" (root) endpoint. By sending a GET request, the user receives a response containing the available endpoints in the API. It is also the homepage of the API browser view.

The */users* endpoint returns a list of the registered users. The response contains the following properties:

- pagination properties;
- results: an array of users.

Each user contains the following properties:

- username: the user's username;
- email: the user's email address;
- url: the url to the user detailed view;
- groups: an array of groups that the user belongs to;
- simulations: an array of simulations that the user owns.

The */groups* endpoint returns a list of the created user groups. The response contains the following properties:

- pagination properties;
- results: an array of groups.

Each group contains the following properties:

- name: the group name;
- url: the url to the group detailed view.

Both the */users* and the */groups* endpoints require the user to have admin access to the API. It is also possible to obtain a detailed view of specific users or groups by providing their id in the endpoint, e.g. */users/{id}*.

Simulations

The */simulations* endpoint handles everything related to simulations: create, read, and execute.

A simulation contains the following properties:

- id: the id of the simulation;
- owner: the username of the user that owns the simulation;
- title: a user defined title for the simulation;
- description: a user defined description for the simulation;
- date_created: the date that the simulation was created (GMT+0);
- url: the url of the simulation;
- execute: the url to execute the simulation;
- modeller_input: the url of the simulation modeller input;
- modeller_output: the url of the simulation modeller output;
- simulator_output: the url of the simulation simulator output;
- postprocessor_input: the url of the simulation postprocessor input;
- postprocessor_output: the url of the simulation postprocessor output.

By sending a POST request to the */simulations* endpoint, the user is capable of creating a new simulation instance. The *scenario_analysis_input* property sent with the POST request is saved in the simulation modeller input, while other input and output data properties have null values until the simulation is successfully executed. The input property must be in the format accepted by Scenario Analysis. The response is an object containing the created simulation.

The */simulations/{id}* endpoint handles access to specific simulation instances. This detailed view contains all the simulation properties, which can be accessed with a GET request.

A GET request to the */simulations* endpoint returns an object with pagination properties and a results array of simulations owned by the user. This list view is more simplified than the detailed view, as such it'll only show the following properties:

- id;
- owner;
- title;

- date_created;
- url.

If the user is an admin, then all simulations in the database are returned.

Integration with Scenario Analysis

The */simulations/{id}/execute* endpoint executes Scenario Analysis for the specified simulation. The input is collected from the simulation modeller input, and after a successful execution, the generated files data is saved inside the other input and output properties. The response includes an object with the following properties:

- errors: an array of serialization errors that occurred when executing the simulation;
- postprocessor_output: the simulation resulting postprocessor output object, i.e. the simulation results.

This request might take some minutes to finish, depending on the contents of the modeller input. As the complexity of the simulation increases (higher number of products, resources, scenarios, etc.), so does the time required for Scenario Analysis to return a result, and for the API to respond with it.

Simulation data after execution

The endpoints henceforth presented are used to retrieve the data generated by Scenario Analysis.

The */modeller-input/{id}* endpoint contains the data for the simulation modeller input. When creating a new simulation, the scenario_analysis_input property is saved here. The response contains the following properties:

- simulation: the id of the related simulation;
- url: the url of the modeller input;
- scenario_analysis_input: the input for Scenario Analysis.

The */modeller-output/{id}* endpoint contains the data for the simulation modeller output. The response contains the following properties:

- simulation: the id of the related simulation;
- url: the url of the modeller output;
- generated_scenarios: an object in which the keys are the names of the generated files, and the content inside these keys is the JSON data for the scenarios generated by Scenario Analysis.

The */simulator-output/{id}* endpoint contains the data for the simulation simulator output. The response contains the following properties:

- simulation: the id of the related simulation;
- url: the url of the simulator output;
- cfg_data: an object in which the keys are the names of the generated files, and the content inside these keys is the string data of the .cfg files.

The */postprocessor-input/{id}* endpoint contains the data for the simulation post-processor input. The response contains the following properties:

- simulation: the id of the related simulation;
- url: the url of the postprocessor input;
- rep_data: an object in which the keys are the names of the generated files, and the content inside these keys is the string data of the .rep files;
- error_reports: an object in which the keys are the names of the generated files, and the content inside these keys is the string data of the .txt files. If an error occurs in the Scenario Analysis simulation, the .cfg files might be empty, and the .txt files will have a traceback of the error. Otherwise, the .txt files will be empty.

The */postprocessor-output/{id}* endpoint contains the data for the simulation post-processor output. The response contains the following properties:

- simulation: the id of the related simulation;
- url: the url of the postprocessor output;
- scenario_analysis_result: a JSON object with the results of the Scenario Analysis simulation.