University of Applied Sciences and Arts Northwestern Switzerland
School of Business

UNIVERSITÀ DI CAMERINO

# Integrating Robotic Movements into a Business Modelling Environment

Prototyping Innovations as BPMN4MoPla

**Oliver Ruggli**

15-656-267 / 110130

*July 09, 2021*

Prof. Dr Barbara Re
Prof. Dr. Knut Hinkelmann
Dr. Emanuele Laurenzi

Master Thesis

**MSc in Business Information Systems (University of Applied Sciences Northwestern Switzerland)**
**MSc in Computer Science (University of Camerino)**

**UNIVERSITA' DEGLI STUDI DI CAMERINO**

***AUTHORIZATION TO GRADUATE ****

**DEGREE COURSE IN _____MSc Computer Science_____**

**Student n.** …110130…….

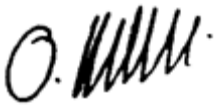
☐ compilation thesis ☐ sperimental thesis ☐ in-depth (analysis) thesis

☐ final paper (BSc) ☐ x project thesis ☐ theorical thesis

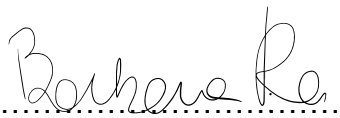
The Professor ……Barbara Re…… Relator/Unicam Tutor of the thesis, authorizes the student …….Oliver Ruggli……. to discuss it on ……21.07.2021….., academic year ……2020/2021…….., with a thesis/final paper entitled: …… Integrating Robotic Movements into a Business Modelling Environment…… ..related in the main subject: …Domain-specific Modelling, BPMN, CPS…….

Camerino, ……01.07.2021……

**Student** **Relator/ Unicam Tutor**



…….....
(signature)



……...
(signature)

**Co-supervisor**

……
(signature)

*** to be attached to the PDF of the thesis, the frontespiece and the substitutive declaration of certification.**

# ATTACHMENT: PDF THESIS

## SUBSTITUTIVE DECLARATION OF CERTIFICATION*

(D.P.R. 445/2000)

I undersigned Oliver Ruggli, born at Olten

in 17/04/1993 resident in Olten( Switzerland ),

address Rosengasse, n. 88- zip code CH-4600,

student n. 110130, enrolled at this University in the Degree course in:

Master Computer Science , having applied to discuss the thesis on

21 / 07 / 2021, aware of the current legislation (legislative refereneces: D.P.R. n. 445/2000; art. 1, D.P.R. n. 403/1998; artt. 1, 2, 3, L. 127/1997)

DECLARE

the PDF of the thesis in attach (and abstract in English if required) is in compliance with the final version of the thesis, approved and duly signed by the Relator on the frontespiece.

The Relator has already signed the authorization to graduate.

Olten , 01/07/ 2021

*(place)* *(date)* *(signature)*

* to be attached to the PDF of the thesis, the frontespiece and the authorization to graduate.

University of Applied Sciences and Arts Northwestern Switzerland
School of Business

UNICAM
Università di Camerino
1336

Università di Camerino

# Integrating Robotic Movements into a Business Modelling Environment

Prototyping Innovations as BPMN4MoPla

**Oliver Ruggli**

15-656-267 / 110130

*July 11, 2021*

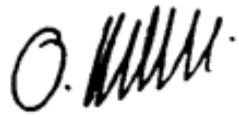
Prof. Dr Barbara Re
Prof. Dr. Knut Hinkelmann
Dr. Emanuele Laurenzi

Master Thesis

**MSc in Business Information Systems (University of Applied Sciences Northwestern Switzerland)**
**MSc in Computer Science (University of Camerino)**

# Statement of Authenticity

I confirm that this master thesis research was performed autonomously by myself using only the sources, aids and assistance stated in the report, and that quotes are readily identifiable as such.



11.07.2021, Olten, Oliver Ruggli

# Abstract

Mobility – the movement of people and goods – is a major expenditure in today's economy. That is not only the case monetarily spoken but true for pollution, more specifically, greenhouse gas emissions as well. Derived from that, the idea of a mobility planner emerged as the practitioner in mobility planning, that intends to efficiently address peoples transportation needs while reducing greenhouse gas emissions. This thesis intends to support the practitioner in a fictive scenario with a proof of concept BPMN4MoPla, which extends the standard BPMN for the purpose of supporting business decision making in mobility planning. Specifically, requirements for robotic car movements were derived and used for the modelling language extension The language extension is implemented through the meta-modelling development toolkit ADOxx. In result, a prototypical modelling tool is deployed allowing the design of Cyber Physical Systems-aware process models. For the execution, transformation steps are displayed to enable a feedback loop with a cyber-physical system provided. For this purpose two environments are presented, the ADOxx Bee-Up tool with an integrated ADOScript execution engine, and the Camunda workflow engine. The evaluation of the approach was carried by mapping the requirements derived from the mobility planning scenario with the developed proof of concept. The approach follows the Agile Modelling Method Engineering lifecycle.

At the University of Applied Sciences Northwestern Switzerland in Olten, the OMiLAB was introduced, a laboratory of a community that originally started from the University of Vienna and deals with all facets of modelling. One CPS component supplied with the lab is the mBot from Makeblock., which was used as the cyber-physical system within this thesis.

**Keywords**: DSML, BPMN, CPS, Mobility planning, Transportation CPS, ADOxx, Business Process, Camunda, Decision support, Modelling, Meta model

# Foreword / Acknowledgements

Many thanks to Knut, Emanuele and Barbara for their long-lasting patience, always fresh and helpful inputs that kept me from chasing my own tail and steered me in the right direction. Especially Emanuele pushed me with his view, approaches and challenges when transforming parts of the Thesis into a chapter for a new OMILAB book, kept me on the edge and let me question and reason a lot of the undertaken steps.

The ADOxx training days by Robert Woisch, Wilfried Utz and Damian, who are behind the ADOxx mailbox, also contributed significantly to the development phase. And as if the training days were not enough, they were extremely prompt and helpful in answering my questions about the OMiLAB and the mBot well after the training days.

My family environment has also contributed a lot of emotional support for which I am very grateful. In addition, I would like to thank my father Marco for taking the time to read through this work carefully and to take on the enormously important role of an outside observer to critically reflect on the content.

# Table of Contents

# 1 Introduction

Robots are fascinating inventions of humans, which come in all shapes, colours, heights, depths, visible, invisible, and other modes of perception while also being launched in space. They are contemplated with a wide variety of feelings and angles, everything from means to ends, to replacing and supporting any work, security and surveillance, awe, hate, concern, and objects of desire. Despite the many domains robots are applicable to, businesses and researcher of any kind tend to explain high complex (mal-) functions with visual aids like live experiments or models.

Within most domains, models serve in the most diverse ways; if logistics is considered, Niemueller, Karpas, Vaquero, & Timmons (2016) describe that models are used very intensively in planning and continuous improvement. This is especially true in the form of virtualisation and simulation of treadmills and other warehouse handling equipment. Especially through simulations, the effects of minimal changes, such as adjusting the treadmill speed, can be visualised. In certain domains, manual assembly of products is still a key success factor considering quality and flexibility. When thinking of flexibility traditional, fully automated assembly using specialized robot stations is mostly not feasible for small lot sizes due to high costs for programming and mechanical adaptations. In the last years, collaborative robots (cobots) entered the market to broaden the way for robot-assisted manual assembly (Froschauer & Lindorfer, 2019).

While also tackled in intralogistics, distribution logistics, more specifically (self-driven) vehicle transportation, models mimic a form of intelligence, referred to as Artificial Intelligence, that responds to the multiple types of environmental interactions. These can

include the behaviour of people in vehicles, bicycles, pedestrians, weather effects such as visibility, rain, heat, cold or even temporary signals used in road works or manual traffic diversion. Elvarsson (2017) shows many models that represent the subtleties of the braking process at intersections, which, even if only a very small part of such intelligence, are extraordinarily essential components. In general, research is mapping entire traffic models to enable self-driving and connected cars (Gora & Rüb, 2016)

While residing in the transportation domain and with the order of the Swiss Federal Council, the AWK Group (Wiederkehr, Frommenwiler, & Walti, 2018) publishes a study that states that Mobility as a Service (MaaS) plans to enable mobility to be fully tailored to the personal needs of the individual and can be purchased as a service. In the sense of a digital mobility assistant, the user is presented with all possible suggestions for the journey from A to B on a platform. These mostly multimodal combinations are tailored to the individual's preferences, the current traffic situation, and other general conditions. The prerequisite for this is the provision of real-time data from all existing mobility providers. MaaS, in its full complexity, also includes other elements: On the one hand, services are to be offered on a less schedule-based and routine-specific basis but will become increasingly dynamic and thus user-oriented. It should be possible to determine the time of travel and the route "on-demand". On the other hand, the user of a multimodal mobility service should not only have the information about his or her trip on a single platform but also be able to do so but also be able to book and pay for the trip without media discontinuity and with just one registration.

If one decides to found a MaaS, mobility planning is a practice that contributes creating conditions for such mobility. Within the EU (CIVITAS, 2021) mobility planning is called Integrated and Inclusive planning. It intends to create sustainable, efficient transport networks and ensures that mobility is incorporated into city development plans and sustainability goals. If mobility is the ability to move about and make transport choices, mobility planning is the practice that supports urban mobility managers in the creation of such transport choices towards more sustainable urban mobility (Lah, 2018).

Models are a valuable means for mobility planning. For example, a transport model supports to visualize the movement of people and goods in a transport network within an area having certain socio-economic and land use characteristics (Okraszewska et al., 2018). They serve as the basis for discussions, analysis, improvements, and for the support of decision-making. As discussed in Deka, Khan, Chowdhury, & Ayres (2018), to achieve higher support in decision-making, cyber-physical systems (CPS) can be employed. CPSs have the benefit of promoting mutual feedback loops between the cyber spaces and physical spaces, leading to quick and incremental improvements.

## 1.1 Background Information

To address this, miniature robotic cars that form a Cyber-Physical system provided by OMiLAB are used. This can consist of a small model car, a robot arm or a purchasable humanoid resembling robots like NAO or Pepper. These enable the Modeler to model and demonstrate their models in a real-life scenario, which provides a fast feedback loop. Often when these models are executed in a CPS, it is possible to tackle the domain of digital twins, especially if parallel to the execution, the model also showing the status of the robot, e.g., the current position of a miniature car or a robotic arm. On an important side note, it must be mentioned that even though OMiLAB provides the CPS and part of the software used later on, one can argue that the applications of this thesis could be conducted without having access to the OMiLAB infrastructure. The software components can be acquired free of charge (although a registration is necessary) and the hardware is also commonly available.

In their delphi study Becker, Vom Brocke, Heddier, & Seidel (2015) came up with 21 grand challenges for information systems. Bork et al. (2019) state that OMiLAB tackles the following three of these challenges:

- Supporting effective collaboration and learning through evolving media repertoires
- Raising collective consciousness

- Developing model-driven methods and tools for the full-scale automated generation of implementation-ready information systems

OMiLAB builds upon four pillars: (a) a granularly defined “Modeling method” with customizable building blocks; (b) an “Agile Modeling Method Engineering framework; (c) a “Digital Product Design Lab”; and (d) dissemination channels for reaching a global community (Bork et al., 2019). These aim to enable a new conceptual modelling research framework devised by (Sabegh, Lukyanenko, & Recker, 2017), which intends to push the conceptual modelling forwards as conceptual modelling has premises and promises that they are yet to foresee.

A distinction must be made between two different models or modelling types. On the one hand, there is the model that people in business create to represent a process in its final form. This may include building blocks that eventually lead to executions on the CPS. On the other hand, the elements, flows and relations must be specified on the metamodel level if an already existing modelling language is extended to support communication with a CPS. Alternatively, a modelling language can be created by designing a meta-model and further on deploy it to a modelling tool allowing Modelers to create models in a newly developed language.

This thesis arose from the intent to set up an OMiLAB room at the FHNW in Olten and to connect to a model and a CPS for the first time in Olten. This work should also serve to guide future student projects or examined deeper in potential master thesis topics.

## 1.2 Problem Statement

This section introduces the research problem. The statement is described below and composed of several aspects and references.

An approach of modelling mobility, respectively the movement of any actors, which supports the integration of an underlying CPS infrastructure is an interesting topic. The aim of this thesis and thus the problem to be tackled is the integration of CPS elements

into a modelling language. Whether this is done from scratch or in an existing language will be shown during the thesis, which is why the proof of concept shows both approaches. Two sides should be united here; on the one hand, the movement of an object should be able to be modelled and on the other hand, a modelling structure should exist where decisions can be made, and movement is then executed.

The above scenario can be abstracted to a variety of research for example, Graja, Kallel, Guermouche, & Kacem (2016) integrated ambulance drone elements into BPMN. The two Systematic Literature reviews by Compagnucci et al. (2020) and Zarour, Benmerzoug, Guermouche, & Drira (2019), allocate multiple extensions of BPMN. 30 of 51 describe the integration of IoT elements in various modelling languages, which evidentially shows a clear interest of the research community in any kind of integration of sensors, objects, or digital twins.

Frank (2013) addresses the challenges of Domain Specific Modelling Languages. Specifically, the effort required to create a language from scratch and the renunciation of already existing concepts that must be re-implemented in one or another way makes the creation even more expensive than it already is. An obstacle mentioned by Braun, Schlieter, Burwitz, & Esswein (2015) is, at least to some extent, ignored in this thesis is the highly required domain knowledge to develop an applicable DMSL. However, this is only possible because the chosen scenario for this thesis is very superficial and should show, that such a scenario is applicable to different domains such as aviation, robots in logistics and delivery planning, to name three examples.

OMiLAB (OMiLAB, 2021), which served as the inspiration for this thesis, support an active global community for conceptual modelling that benefits from open artefacts. To this end, they act as a facilitator to the development of scientific methods and technologies for all those who value models. In addition, they act as a platform where participants can bring in ideas related to modelling and engage in the exploration process. They follow a user-driven approach in their understanding of the term “model”, recognizing that there are useful models in widely different domains like information technology, biology, chemistry, or medicine, as well as various functional areas like procurement, marketing,

logistics and engineering. If we observe only the view of controlling an object, software development might often be the first solution that comes to mind. Bork et al. (2019) and Bork (2018) argue that software development also has an underlying concept that can be represented by modelling. This abstraction of software is then on a similar granularity as business processes and can be combined with them. Bork et al. (2019) explicitly state: "When we talk about software, we do not talk about code but use models to explain what they do." Karagiannis et al. (2016) describe that when designing languages for domain-specific modelling, a modelling method engineer will, on the one hand, consider the established experience and lessons learned from standard languages or notations and, on the other hand, will consider specializations and/or extensions with respect to modelling requirements raised for the addressed domain by the stakeholders who will either benefit from using models or work on the creation of models.

A proof of concept that either a new DSML or the existing BPMN language can be used to integrate CPS movement elements into a business and decision structure faces several challenges. With a glance at the healthcare domain, several researchers have also identified the need to reflect domain-specific concepts. Braun et al. (2016) have adapted the existing modelling notation BPMN to cover the needs of the clinical pathway in a newly created domain-specific modelling. This is also affirmed by (Mernik, Heering, & Sloane, 2005), who argues that a domain-specific modelling language is more expressive and easier to use than a general-purpose language in its domain of application due to its tailoring to a specific domain. While business-like activities and decisions about mobility planning can be specified in a BPMN model, the standard is not expressive enough to cover transportation movements, which hinders the feedback loop from the physical to the cyber space.

The literature review examines not only various CPS systems with application domains and then focus on mobility planning but also the constructs of modelling languages. For the development of domain-specific modelling languages, there are two broad directions that can be chosen: the creation of a language from scratch, and the adaptation of an existing one. This thesis provides an inside into both approaches.

## 1.3 Thesis Design

This section describes the different aspects of this research. It commences with the thesis statement, outlines the research objectives as well as the research question, followed by the delineations and limitations, and concludes with the significance.

The following thesis statement serves as a general guideline for this thesis:

***"It is possible to design a DSML that integrates the control of movements of physical transportation means into a business process."***

The following research question is in focus for this work:

*"How can modelling languages be designed for Mobility Planning, which allows to combine modelling of movements with the process- and decision logic of a business domain?"*

The goal of this thesis is to provide a modelling tool that incorporates car movements that can be modelled on the same layer with business worth decisions. Table 1 displays subdivided steps aiming to resolve the research question. First an understanding of the problem is required to then derive requirements (RO1) for a modelling and execution environment. In a second phase it needs to be analysed, how this problem can be solved (RO2). The concept shall then be implemented as a proof of concept (RO3).

| | |
|---|---|
| **Research Objective 1** | Elicitation of requirements concerning the modelling language, the modelling tool, and the CPS environment. |
| **Research Objective 2** | Conceptualization of the needed modelling language combinations/ extensions of the selected language, which are useful to meet the scenario requirements. |
| **Research Objective 3** | Implementation of the concept and executing movements in the Cyber-physical space. |

Table 1 Detailed research objectives

## 1.4 Delineation and limitations

A workflow engine development to run a customised BPMN model will not take place during this thesis. However, this can serve as a basis for another topic, which is more concerned with the discussion of executable models. This is outlined further in Chapter 6.

In the case of a commissioned project, a domain-specific requirements catalogue would first have to be conducted with a focus group to concretize the means as well as the evaluation of the prototype BPMN4MoPla. During this thesis, the requirements are derived based on the literature review and inputs from the supervisors.

This thesis will not focus dynamic behaviour of robots; the contribution is in the manner of modelling CPS to achieve a reduction of the effort to design them.

## 1.5 Significance

The significance can be summarised from the previous sub-chapters. The most important argumentation is how to bring together a business logic and a way to model and later control a CPS, or more generally, the movement of a robotic component. The feasibility has underlined a proof of concept applied to the mobility planning domain named BPMN4MoPla.

This thesis is one of the first OMiLAB projects at the University of Applied Sciences Northwestern Switzerland in Olten and should serve as an inspiration, basis, and reference for further projects.

## 1.6 Chapter overview

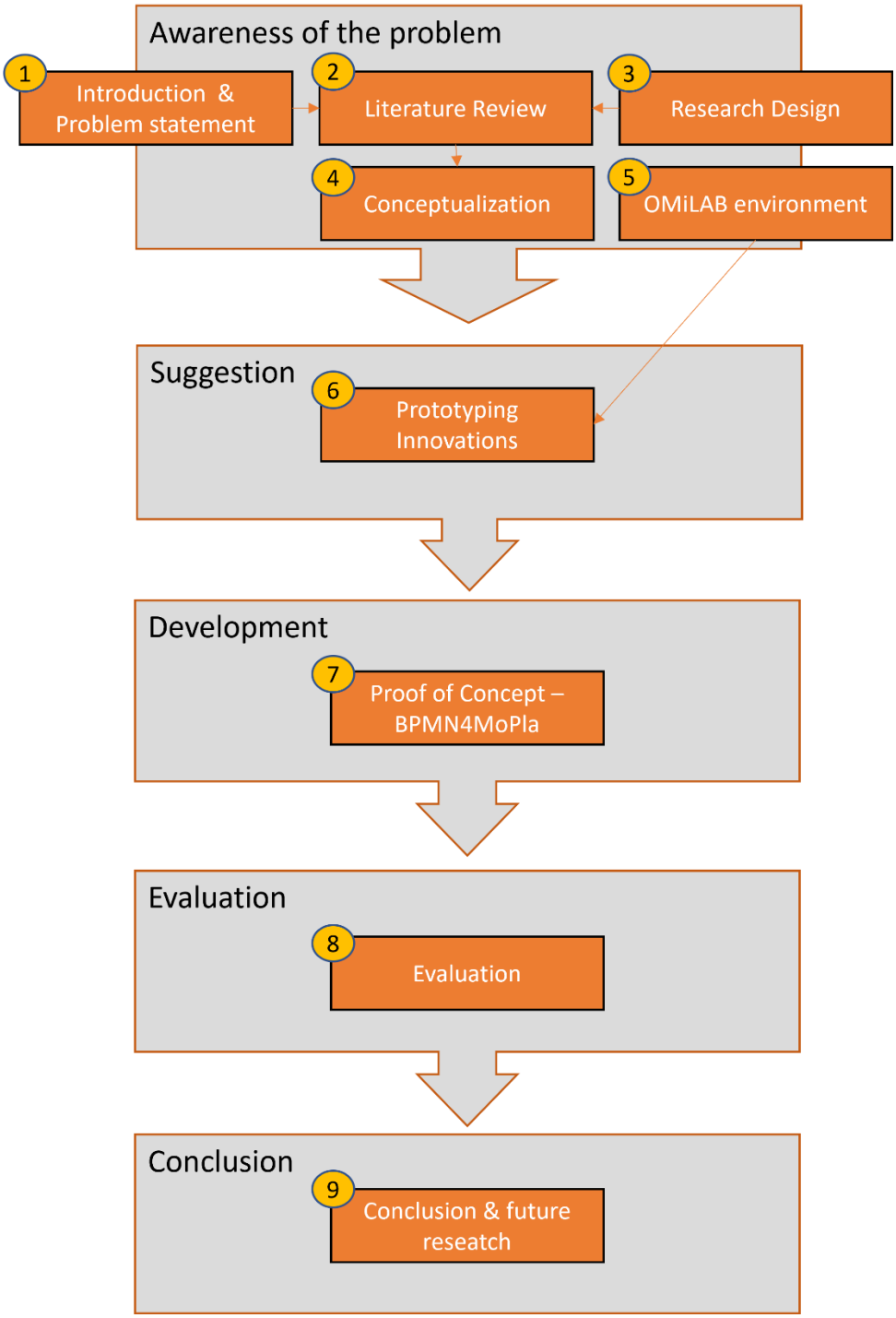



Figure 1 Thesis chapter overview

# 2 Literature Review

The aim of this literature review chapter is to provide a theoretical background and to outline the limits of this thesis.

The literature review provides information on the underpinning concepts of this thesis. The first part deals with the description of robots, CPS, and the domain.

In the beginning, the importance of robots in logistics is explained by means of autonomous mobile robots and their movements. This leads to Cyber-Physical systems, which are described and differentiated from IoT and Digital Twins. After an overview of the application examples of CPS, the mobility planning domain is explained as this is the application domain in which the thesis scenario takes place.

The second part of the literature review describes modelling methodology and is introduced by the description of metamodels. After a distinction between general-purpose-modelling languages and domain-specific-modelling languages, extensions are discussed, and the impressive effort in IoT extensions for BPMN is presented with two systematic literature reviews.

In the summary, the central findings are mentioned again, in particular, Graja et al. (2016), from which this thesis differs in that in addition to BPMN4MoPla also the physical feedback in the cyber-physical space shall be presented.

## 2.1 Autonomous mobile robots

Robot parts have long been in use in warehouses or assembly lines. In logistics, automated guided vehicles (AGV) have evolved into autonomous mobile robots (AMR). Although AGVs are still in use, the DHL Customer Solutions & Innovation report (2016) referred

to a study[1] that reported in 2010 that 35 million more logistics workers will be needed in the next 30 years, which will thrive the will to upgrade or evolve current logistic systems. Figure 2 represents the evolution from mechanical guidance to laser guidance and the next step, which combines multiple technological advances in vision-based guidance (Fragapane, de Koster, Sgarbossa, & Strandhagen, 2021).

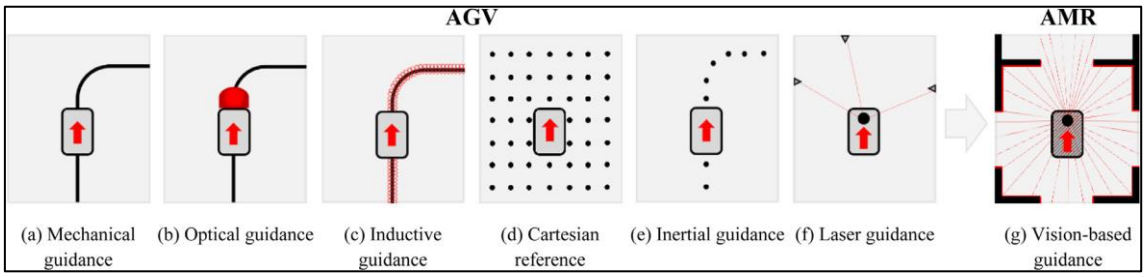



Figure 2 Guiding systems for AGVs and AMRs (Fragapane et al., 2021)

As in other industries, there is no single factor driving technology. Fragapane et al. (2021) distinguish between advances in hardware and software and further sub-categorise them, namely: sensors, robotic locomotion mechanism, batteries, equipment manipulation, processing devices are sub-categorised as hardware while simultaneous localisation and mapping (SLAM), motion planning and artificial intelligence are categorised as software.

AMRs are typically equipped with a wide array of small, low-cost, and power-efficient sensing technologies providing input data for autonomous navigation. Figure 3 provides an overview of the applications areas. Integrated laser scanners such as Light Detection and Ranging (LiDAR), 3D cameras, accelerometers, gyroscopes, and wheel encoders, which provide information on wheel positions to calculate the distance that the robot has driven or turned, and capture and transmit enormous amounts of data about the AMR's immediate, extended and anticipated environments, along with its internal condition (De Silva, Roche, & Kondoz, 2018). A limited battery capacity and long charging times were

[1] https://gbr.pepperdine.edu/2010/08/preparing-for-a-future-labor-shortage/

weak points of AGVs and reduced performance, utilization, and computational power. In addition, traditional lead-acid high-capacity batteries required increased vehicle size. The high-capacity batteries enable a longer operational time and provide more power for the calculations needed for autonomous navigation and operations. They also allow the AMRs to be smaller (this also holds for the newest AGVs) and thus to be deployed in narrow-aisle areas, or even directly underneath multiple loads stored closely in deep lanes (Lamballais, Roy, & De Koster, 2017). With these technological improvements, the importance of battery management has declined somewhat, although it may still be relevant in 24/7 operations (Zou, Xu, Gong, & De Koster, 2018).

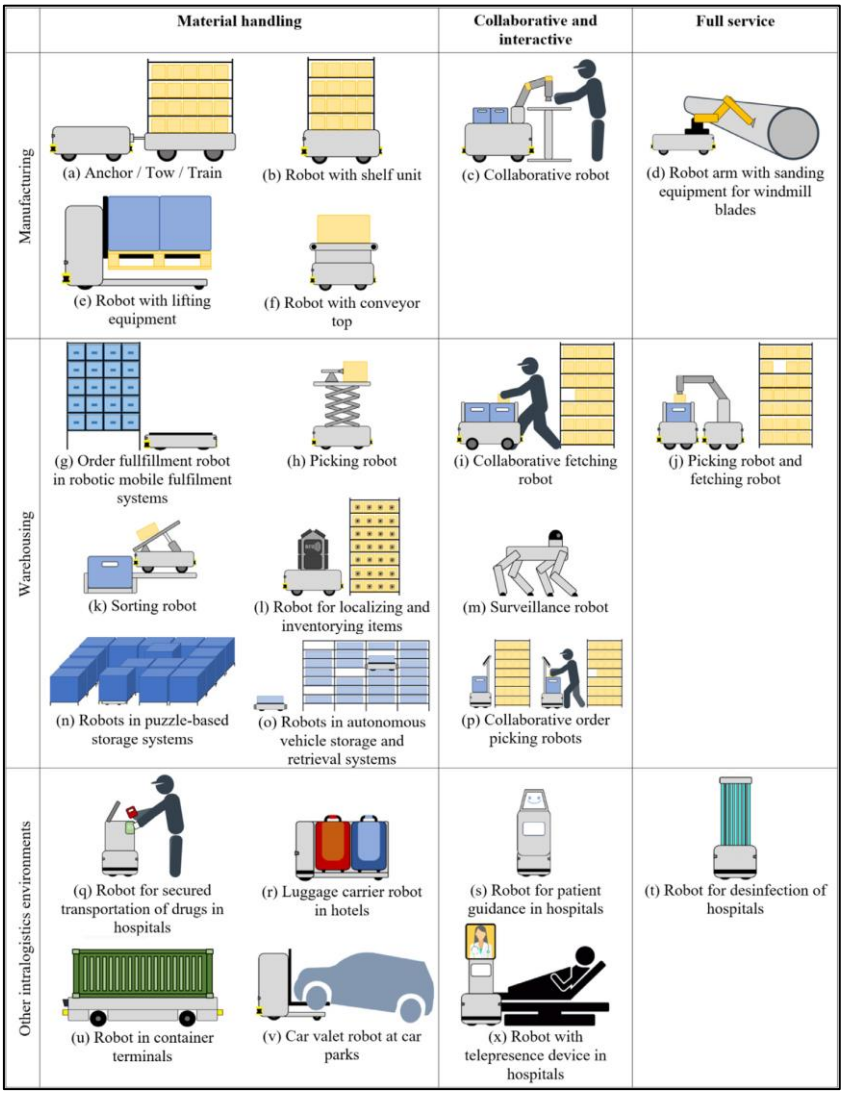



Figure 3 Types of AMRs and examples of applications (Fragapane et al., 2021)

## 2.2 Modelling robotic movements

First, it should be mentioned that when the term "modelling" is searched in conjunction with either "intralogistics robots" or "autonomous vehicle" in the Google Scholar search, the most popular results are about mathematical behaviour and models.

One example is published by the ETH Zurich (Elvarsson, 2017) on Modelling Urban Driving and Stopping Behaviour for Automated Vehicles, which explains formulas for desired driving behaviour, braking behaviour, and average directional behaviour at

intersections using formulas and describing them in a signal simulation. In the introduction and the

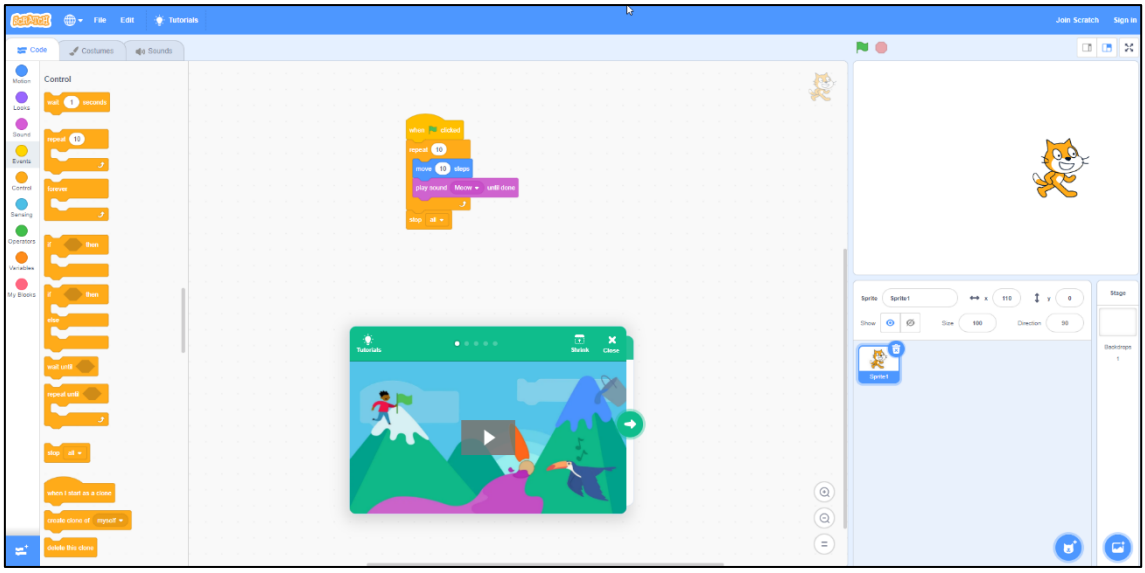


Figure 4 Excerpt of the Scratch UI

When searching for robotic applications, communities of robotic engineers are more often discussing the use of specific programming languages than the application method of providing movement. Some examples are as follows: C/C++, Python, Java, C#/Net, Matlab, Hardware Description Language, LISP and Prolog, Industrial Robot Languages, Scratch (excerpt displayed in Figure 4), Pascal.

For example, with Scratch (Scratch, 2021), it is possible to programme own interactive stories, games and animations and share the creations with others in the community online. Scratch helps young people to think creatively, reason systematically and collaborate with each other, which are, according to the Scratch homepage[2], essential skills for life in the 21st century. A similar experience is also provided from the launched

[2] https://scratch.mit.edu/

Roberta Initiative[3] in Germany, where you can choose to model for existing robots like EV3 LeJOS, NXT (Lego Mindstorm), micro: bit, Bot'n Roll, NAO and additional to eleven other robots, there is the mBot by Makeblock, which is also used in the further course of this work.

Tesla, Google, Uber, and GM are all creating their own self-driving cars that can run on real-world roads. Analysts' predictions vary when it comes to estimating a time span until fully self-driving cars are widespread. In comparison to intralogistics, the absolute majority in the field of self-driving cars is trying to rely on AI and deep learning. Above all, programming environments in connection with Python offer cost-effective variants for training your own network. In an article published by towards data science, it is demonstrated how a Raspberry PI, a SunFounderPiCar kit and Google's Edge TPU, which are shown in Figure 5, can be used to build an environment for such a project (Towards Datascience, 2019).

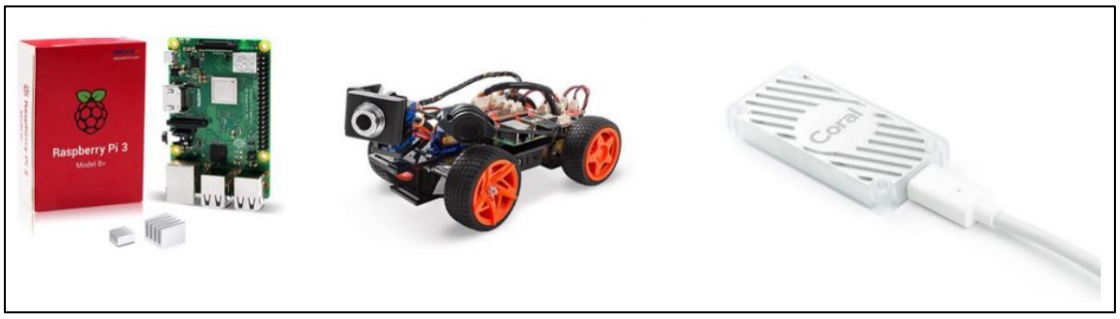



Figure 5 Raspberry Pi 3 B+ (left), SunFounder PiCar-V (middle), Google Edge TPU (right)

[3] https://www.roberta-home.de/impressum

## 2.3 Cyber-Physical Systems

When narrowing the scope, an entity of an autonomous robotic system can be described as a Cyber-Physical System. Cyber-Physical Systems are systems that integrate computing elements with the physical components and processes. The computing elements coordinate and communicate with sensors, which monitor cyber and physical indicators, and actuators, which modify the cyber and physical environment. Cyber-Physical Systems use sensors to connect all distributed intelligence in the environment to gain a deeper knowledge of the environment, which enables more accurate actions and tasks. Physical resources can be categorized as either static (medical devices, sensor networks) or mobile (e.g., robots) (Huang et al., 2009). Confirmed by (Boulila, 2019) Cyber-Physical Systems consist of computation, communication and control components tightly combined with physical processes of different domains such as mechanical, electrical, and chemical. On the other hand, , also stated by Boulila (2019), IoT is the technology enabling the interconnection of all types of devices through the internet to exchange data, optimize processes, monitor devices to generate benefits for the industry, the economy, and the end-user. IoT is described as a network of physical objects that are equipped with sensors and other technologies to connect and exchange data over the Internet (Kumar, C, A, & Anjali, 2021). It is composed of a network of sensors, actuators, and devices, forming new systems and services. (Boulila, 2019). IoT is elaborated further in chapter 2.10.2 because the amount of effort in conjunction with modelling (especially BPMN) is accomplished.

Deka et al. (2018) define Cyber-Physical System perceived in Europe as follows:

> *"Cyber-Physical System are systems with embedded software (as part of devices, buildings, means of transport, transport routes, production systems, medical processes, logistic processes, coordination processes and management processes), which:*
>
> - *directly record physical data using sensors and affect physical processes using actuators;*
> - *evaluate and save recorded data, and actively or reactively interact both with the physical and digital world;*

- *are connected with one another and in global networks via digital communication facilities (wireless and/or wired, local and/or global);*
- *use globally available data and services;*
- *have a series of dedicated, multi-modal human-machine interfaces".*

In the description of the OMiLAB Physical Objects (OMiPOB) environment (Karagiannis & Muck, 2017) the term CPS explicitly stands for the robotic actor. In order to differ CPS's, the following classes have been introduced:

- Class1: RobotCar with basic functionality (OMiPoC2)
- Class2: RobotCar with enhanced functionality (OMiPoC1)
- Class3: RobotArm (OMiArm - DoBot)
- Class4: Car (OMiCar – mBot)
- Class5: DrawBot (OMiSpd – mDrawBot)
- Class6: Rover (OMiRov – mBot Ranger (Land Raider))
- Class7: Transporter (OMiMec – Makeblock Mecanum Wheel Robot Kit)
- Class8: Humanoid (OMiNAO)

These different classes describe the capabilities of the CPS's. For the remainder of the thesis, the term OMiPOB is adopted, and CPS referred to as a robotic actor. As a side note, the used equipment in this thesis is the mBot from Makeblock found in Class 4 in the bullet list above.

### 2.3.1 Correlation and Comparison of CPS and Digital Twin

CPS and Digital Twins (DT) have similar features, and both are about the fusion of the cyber and physical worlds. However, CPS and DTs are not identical. CPS and DTs were proposed around the same time. However, the DT did not receive much attention until Glaessgen & Stargel (2012) published the digital twin paradigm for their future vehicles. Lee (2015) states that CPS received attention from academia and governments, with Industry 4.0 listing CPS as its core. Through cyber-physical interaction and control, both enable precise and better management of the physical world. DTs focus more on virtual models, which enable the one-to-one correspondence in a DT, while CPS emphasize 3C

capabilities, which lead to one-to-many correspondence. Mathematic and data models play an important role in a DT to help interpret and predict the behaviour of the physical world based on various data. Thus, sensors and actuators can be considered as the core elements in CPS, while models and data are the core elements in a DT. CPS are more akin to a scientific category, whereas DTs are akin to an engineering category. (Tao, Qi, Wang, & Nee, 2019). DT relates to a virtual representation that serves as the real-time digital counterpart of a physical resource (Negri, Fumagalli, & Macchi, 2017).

Digital Twin itself is then often confused with Digital Shadow or Digital Model. The main difference is described and drawn by Fuller, Fan, Day, & Barlow (2020), with the data flow shown in Figure 6.

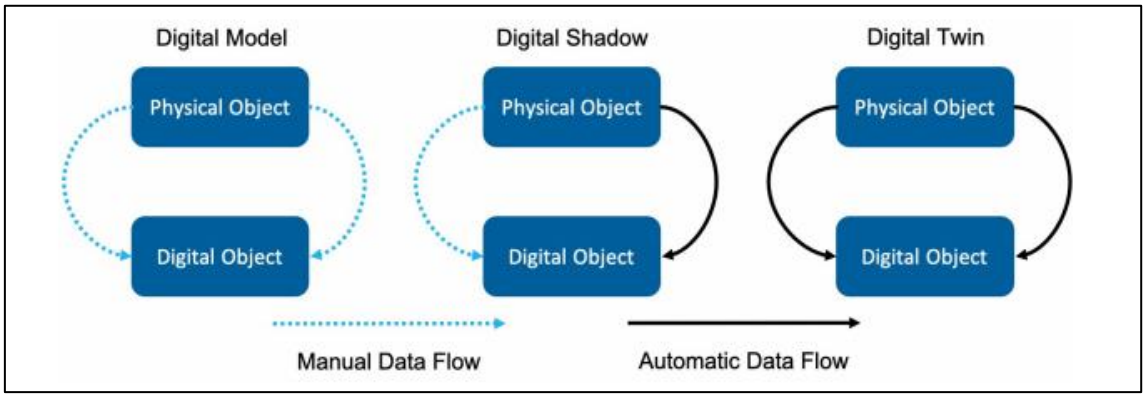



Figure 6 Misconceptions of digital twins (Fuller et al., 2020)

A digital model is described as a digital version of a pre-existing or planned physical object to correctly define a digital model; there is no automatic data exchange between the physical model and the digital model. Examples of a digital model could be but not limited to plans for buildings, product designs and development. The important defining feature is there is no form of automatic data exchange between the physical system and digital model. This means once the digital model is created, a change made to the physical object has no impact on the digital model either way. A digital shadow is a digital representation of an object that has a one-way flow between the physical and digital object. A change in the state of the physical object leads to a change in the digital object and not vice versa (Fuller et al., 2020).

If the data flow between an existing physical object and a digital object, and they are fully integrated with both directions, this constituted the reference “Digital Twin”. A change made to the physical object automatically leads to a change in the digital object and vice versa (Fuller et al., 2020).

## 2.4 Applications of Cyber-Physical Systems

Cyber-physical systems (CPSs) are engineered systems that offer close interaction between cyber and physical components. The field of CPS has been identified as a key area of research and CPSs are expected to play a major role in the design and development of future systems. Khaitan & McCalley, (2015) published recent advancements made in the development and applications of cyber-physical systems. Existing research work has been classified according to its characteristics and future challenges have been identified. The aim is to grant an overview of CPS applications and motivate researcher and system designer to propose novel solutions for making wide-scale adoption of CPS a tangible reality.

In an overview, Khaitan & McCalley, (2015) shows that before applications are discussed that on the one hand side, Design, which includes architecture and modelling, simulations, tools and programming frameworks, as well as verification. On the other side, aspects and issues concerning security, resiliency, reliability are major topics to research on their own before a CPS can be applied to a domain.

Table 2 shows a wide range of applications scenarios where research papers have been published.

| **Domain** | **Examples of published papers** |
|---|---|
| **Vehicular systems and transportation** | • Data fusion in distributed CPS<br>• Transportation<br>• Design of CP vehicles |

| | |
|---|---|
| | • Road monitoring |
| **Medical and healthcare systems** | • Implantable/life-support medical devices<br>• Robot-assisted operation<br>• Development of medical application platform |
| **Smart homes and buildings** | • Electronic equipment and HVAC control<br>• Healthcare in smart homes<br>• Energy/ power management in smart homes |
| **Social Network and gaming** | • Integrating physical inputs<br>• Single player/ one-to-one<br>• Multi player/ social network |
| **Power and thermal management** | • Minimizing electricity costs<br>• Workload placement<br>• CP interaction based thermal power management |
| **Data centre** | • Power management and energy costs for computation |
| **Electric power grid and energy systems** | • Modelling<br>• Renewable energy<br>• Security related aspects |
| **Networking systems** | |

| **Surveillance** |
| --- |

Table 2 Applications of CPS adopted from (Khaitan & McCalley, 2015)

As a closing remark, Dibaji et al. (2019) state the CPS has been attracting more and more attention lately. A key concern that is pervasive in CPS is the need to ensure security in the face of cyber-attacks. The review of systems, and control methods have been proposed for CPS security. A good effort is currently made into researching the security CPS due to its wide range of application domains.

## 2.5 Modelling in Cyber-Physical Systems

For modelling and simulating CPS, authors (Zeng et al. (2014), Taha, Taha, & Thunberg (2021)) uses the approach of virtualising sensor with mathematical models. For hands-on activity Zeng et al. (2014) and Taha et al., (2021) underline exercises and research with results generated in the Tool Acumen. An intuitive tool for simulating mathematical models of cyber-physical systems and for visualizing them as plots or in 3D. Researchers use it to develop better CPS technologies. It is distributed as free, open-source software under a BSD licence (Acumen, 2016). In order to achieve a simulation displayed in Figure 7. A mathematical model addressing the dynamics of a common quadcopter model have been constructed. The derived equations for the dynamics are then expressed in a core language (Zeng et al., 2014).

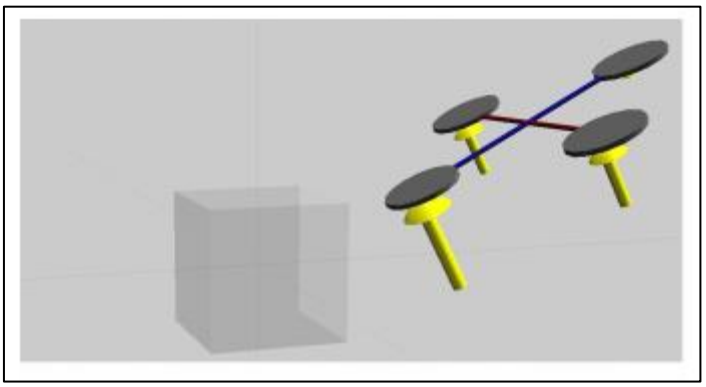


Figure 7 : The simulation results of the quadcopter model (Zeng et al., 2014)

UML 2 (Bock, 2006) is widely used for modelling software systems. Its derivative SysML (SysML, 2017) has many features that are useful for CPS. The internal block diagram notation of SysML, which is based on the UML 2 composite structure diagrams, particularly with the use of flow ports, has great potential for modelling complex systems, such as aircraft fuel systems (Derler, Lee, & Sangiovanni Vincentelli, 2012).

## 2.6 Transportation Cyber-Physical System

With the thesis scenario in mind, this section focus on Transportation Cyber-Physical Systems (TCPS). Acatech (2011) cites that ageing populations, climate change, the advent of megacities, increased energy requirements and the overarching need for smart, green, and integrated transport have been identified as global challenges faced by our society. With the advancement in the field of embedded intelligence systems promising technological solutions to address the before-mentioned major challenges are brought up. With the simultaneous rapid advancement in IoT (confirmed in (Kumar et al., 2021)) CPS are equipped with the power of collective intelligence as opposed to individual entities. Compared to a traditional transportation system, a TCPS can enable higher efficiency and reliability by enabling increased feedback-based interactions between the cyber and the physical system in transportation (Deka et al., 2018).

## 2.7 Mobility planning

In accordance with Acatech (2011), WEF (2021) publishes the importance of Mobility – the movement of people and goods – the provision of access to jobs, education, healthcare and trade. In the face of the rapid population growth that is expected to raise to 10 billion by 2050, the mobility systems cannot meet this demand without increasing congestion and pollution. While megacities might be overstretched, rural areas will fall further behind. Also by 2050, a study (Fuglestvedt, Berntsen, Myhre, Rypdal, & Skeie, 2008) estimates that 30-50% of the total $CO_2$ missions originate from the transport sector wherein 2008 it was around 20-25%, and in 2021 within the EU it is accounted for a quarter of the total emissions (CIVITAS, 2021). A conducted study (European Commision, 2013) estimates the external costs of transportation in urban areas consisting of congestion, air quality, accidents, noise and $CO_2$ about EUR 230 billion annually.

In order to tackle these, Eltis (2021) reasons the relevance of the Sustainable Urban Mobility Plan (SUMP) concept with the need for more sustainable and integrative planning processes as a way of dealing with the complexity of urban mobility and new approaches to urban mobility planning are emerging rapidly in an ever-changing urban mobility climate. Due to people its eagerness to adopt new modes of transport (for example, Mobility as a Service and shared transport), the urban mobility dialogue is steadily evolving. As tackled in the introduction mobility planning is a practice that contributes to creating conditions for such mobility of the future. If mobility is the ability to move about and make transport choices, mobility planning is the practice that supports urban mobility managers in the creation of such transport choices towards more sustainable urban mobility (Lah, 2018).

## 2.8 Modelling- Methods and Metamodel definition

According to Karagiannis & Kühn (2002a), modelling methods consists of two components: a modelling technique, which is divided into a modelling language and a modelling procedure, and mechanisms & algorithms (shorten: mechanisms) working on the models described by the modelling language  The modelling language contains the

elements, with which a model can be described. A modelling language itself is described by its syntax, semantics, and notation. The modelling procedure describes the steps applying the modelling language to create results, usually models. Figure 8 grants an overview.

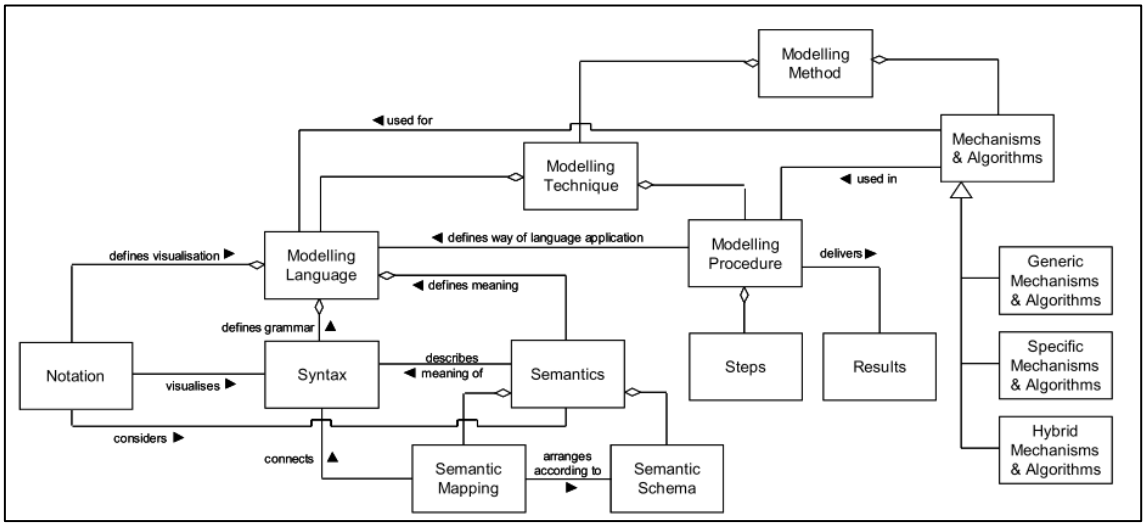



Figure 8 Components of modelling methods (Karagiannis & Kühn, 2002a)

To create a metamodel, a metamodeling language is used. The model defining the metamodeling language is the meta-metamodel or meta2-model (Geisler, Klar, & Pons, 1998).

The amount of meta-metamodel is depending on a useful level of abstraction and not limited to a certain level. To use concepts such as “thing”, “property”, and “relation” may be helpful but lack semantics, especially if the language of the “finishing” level should provide the foundation for implementing the lower levels. According to different authors (Karagiannis & Kühn, 2002a; Laurenzi et al., 2017), the three levels proposed in Figure 9 are sufficient.

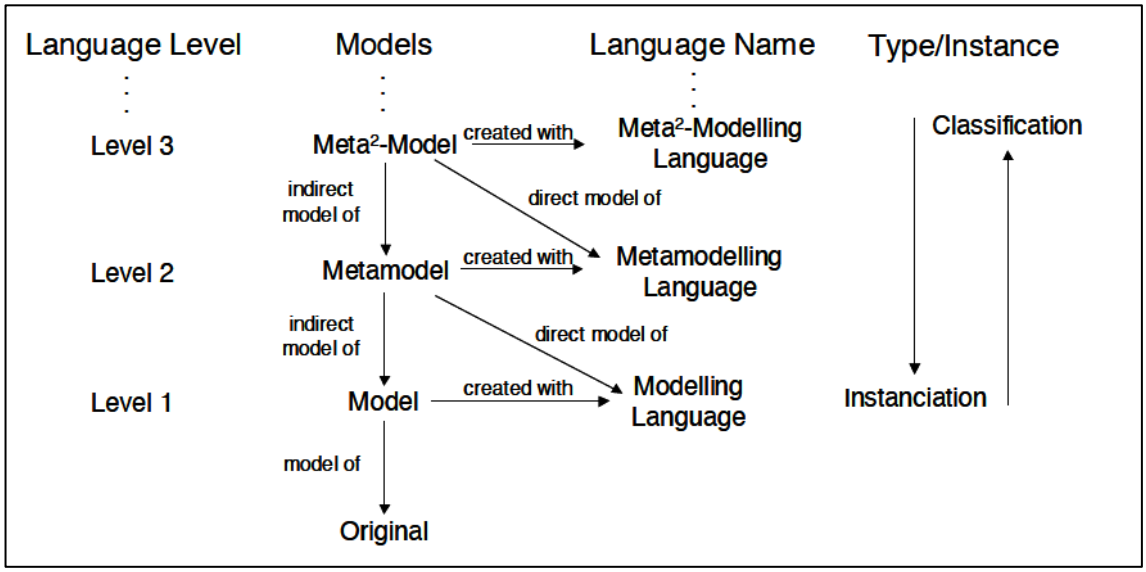



Figure 9 Metamodeling Hierarchy (Karagiannis & Kühn, 2002a)

*"Metamodels are at the heart of any conceptual modelling language as they establish the abstraction level to be applied while creating models. This abstraction level is realized by means of the available concepts of a modelling language and the valid combinations thereof."*(Bork, 2018)

## 2.8.1 Metamodel visualisation

Modelling languages such as BPMN and UML are widely used in industry and academia. Such modelling languages are usually introduced in specification documents maintained by standardization institutions in the example of the afore-mentioned languages Object Management Group (OMG). Being the primary and often even the single source of information, such specifications are vital for modelers, researchers, and tool vendors. The most important terms are visualized in Figure 10.

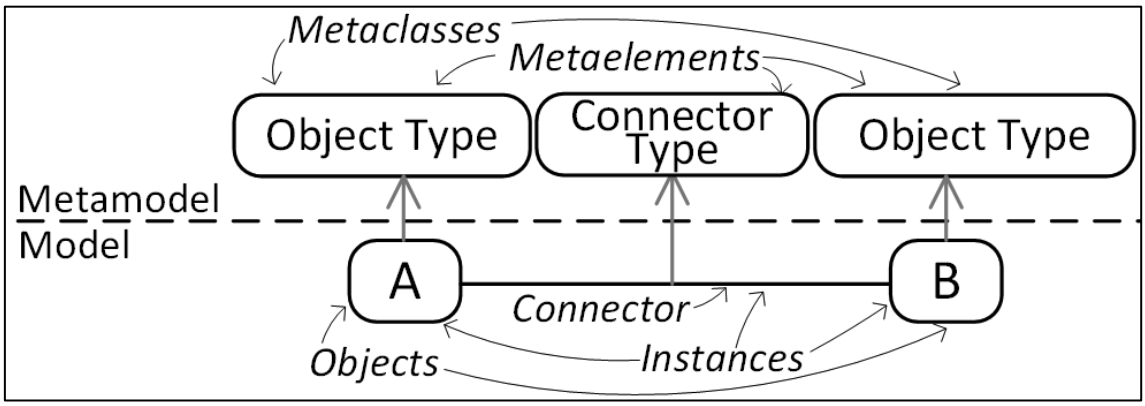



Figure 10 Terminological foundation metamodel and model elements (Bork, Karagiannis, & Pittl, 2018)

Bork et al. (2018) describe 7 different metamodel visualisation techniques where multiple visualisations are not only permitted but desirable due to multiple facets of a modelling language. In the case of BPMN, OMG (2013) applied the following 5 out of 7 described techniques to visualize their metamodel:

- Visual Metamodel
- Slicing Metamodel
- Referencing Metamodel
- Matrix
- Table

Enterprise modelling languages (Braun, 2015) like Archi-Mate (The Open Group, 2019), Business Process Model and Notation (OMG, 2013), Case Management Model and Notation (OMG, 2016), Decision Model and Notation (OMG, 2020) are typically modelled as UML class diagrams in level two. Figure 11 shows that most of the industry modelling standards from well-known standardisation organisations have a UML-class diagram as a graphical mean to represent the abstract syntax.

| | | Abstract Syntax | | |
|---|---|---|---|---|
| | | Graphical | Textual (formal) | Textual informal) |
| OASIS | Universal Business Language (UBL) | UML-Class Diagram | BNF | Natural Language |
| | The eBusiness eXtensible Markup Language (ebXML) Business Process Specification Schema (BPSS) | | XSD | Natural Langauge |
| | Web Services Business Process Execution Language (WS-BPEL) | UML-Class Diagram | XSD | Natural Language |
| OMG | Business Process Model and Notation (BPMN) | UML-Class Diagram | XSD | Natural Language |
| | Case Management Model and Notation (CMMN) | UML-Class Diagram | XSD | Natural Language |
| | Decision Model and Notation (DMN) | UML-Class Diagram | XSD; BNF | Natural Language |
| Open Group | ArchiMate | UML-Class Diagram | | Natural Langauge |

Figure 11 Representation of standard modelling from (Efendioglu, Woitsch, Utz, & Falcioni, 2017)

## 2.9 Modelling languages

Zečević et al. (2017) classify modelling languages into two categories: General Purpose Modelling Languages (GPMLs), which are applicable in every domain; and Domain-Specific Modelling Languages (DSML), which are designed for a specific domain, context, or industry.

A GPML provides rudimentary concepts such as “class”, “relation”, “attribute”. Frank (2010). defines a GPML as “a GPML is a modelling language that is thought to be independent of a particular domain of discourse. Instead, it should be suited to cover a wide range of domains. It consists of generic modelling concepts that do not include any specific aspects of a particular domain of discourse”.

In contradiction, Karagiannis et al. (2016) state that the exact boundary of what "domain-specific" means and where it differentiates from "cross-domain" or "general purpose" is not fixed in an absolute way. Some languages are more specific than others, and some domains are narrower than others. The notion of the domain itself may have different interpretations—it could be a business sector, a community-driven paradigm, a narrow application area or even a single (typically virtual) case of an enterprise that is not interested in model interoperability or understanding outside its environment.

### 2.9.1 General-purpose modelling language (GPML)

Van Deursen, Klint, & Visser (2000) mentions: "In all branches of science and engineering one can distinguish between approaches that are generic and those that are specific". Frank (2010) describes a GPML as a modelling language that is thought to be independent of a particular domain of discourse. Instead, it should be suited to cover a wide range of domains. It consists of generic modelling concepts that do not include any specific aspects of a particular domain of discourse.

Many system developers are likely to consider a general-purpose modelling language GPML such as UML as a suitable tool for system design. An instrument for system design due to the provision of rudimentary concepts such as "class", "relation", "attribute". As obvious as such a choice may seem, it is questionable: In everyday life, Frank (2014) pointed out, we would consider it completely unacceptable if we had to communicate with a language that was limited to a few primitive concepts such as class or attribute. Instead, we expect of a language that it provides concepts that allow differentiated communication without forcing us to explain used terms ourselves. These considerations have led to the development of modelling languages that are designed for specific domains. A DSML is a modelling language that addresses specific domains of discourse. It is based on concepts derived from terms of the corresponding domain.

A report (Erickson & Siau 2013) about the history of UML to help deal with the dual problems of the general nature of UML, and the necessity to make it also domain-specific,

UML 2.X was developed to create a modified version of the language that can extend the language so that it also covers specific domains, rather like SAP's "industry solutions" for customizing their ERP product to specific industries.

## 2.9.2 Domain-specific modelling languages (DSML)

Frank (2010) defines a DSML as "a DSML is a modelling language that is intended to be used in a certain domain of discourse. It enriches generic modelling concepts with concepts that were reconstructed from technical terms used in the respective domain of discourse. A DSML serves to create conceptual models of the domain it is related to".

Sprinkle, Rumpe, Vangheluwe, & Karsai (2010) publish about the State of the Art and Research Challenges of Metamodeling (literally, "beyond Modelling" is the Modelling of models) that despite the many tools available for metamodeling, the underlying mechanism of object-orientation has fostered that most of the metamodeling tools use a common set of abstractions with only slight variations. Using these abstractions, it is possible to raise the specification of a language and its tooling far above the implementation layer. Additional capabilities increase the power of metamodeling by permitting the synthesis of languages as well as automated or semi-automated analysis and synthesis techniques.

Braun et al. (2015) built a DSML by extending BPMN. They provide insights on the approach adopted by comparing it to their previous work in Burwitz, Schlieter, & Esswein (2013), in which a dedicated DSML was built from scratch addressing the same application domain. The comparison includes eleven criteria. The criteria and the comparison results are depicted in Figure 12, where a filled dot refers to a fulfilled criterion, the dot within a dot refers to partially fulfilled criteria, and the hyphenation represents a not fulfilled criteria.

| Criterion | Extension Approach | | DSML Approach (see [5]) | |
|---|---|---|---|---|
| *Domain Representation* | Nearly complete<br>* Discussion on conditional equivalences necessary (see Table 1) | ● (*) | Complete | ● |
| *Meta Model* | Level M2 / $M2_{profile}$ | ● | Level M2 | ● |
| *Abstr. Syntax* | Limited capabilities (missing element constraints), no views possible | ⊙ | Complete specification<br>Views are possible | ● |
| *Concr. Syntax* | Limited by BPMN style ([24], p. 8) | ⊙ | Not limited | ● |
| *Procedure* | Limited guidance [1, 36] | ⊙ | Limited guidance [10] | ⊙ |
| *Concept Reuse* | Broad reuse (e.g., task type)<br>Focus on domain concepts | ● | Re-design of basal process concepts<br>Focus on domain and process concepts | – |
| *Cost of Design* | Depends on relations to BPMN | ⊙ | Depends on language | ⊙ |
| *Tool Support* | Good (range of tools) | ● | Limited | – |
| *Dissemination* | Very good | ● | Limited | – |
| *Integration* | Integration with other extensions<br>Reuse of other BPMN interfaces | ● | Dedicated interfaces required<br>No integration reuse | – |
| *Execution* | BPEL integration with adaption | ⊙ | Requires dedicated design | – |

Figure 12 Comparison of the Extension and the DSML approach extracted from Braun et al. (2015)

From the comparison results, one can conclude that a domain-specific adaptation approach presents more advantages in terms of the reusability of concepts, tool support, dissemination, integration, and execution.

### *2.9.2.1 Requirements gathering*

Whenever creating a DSML either from scratch or when extending an already existing language, the beginning consists of high-intensity meetings of domain experts (Bork et al., 2018; Frank, 2013). Especially Frank (2013) differs between generic and specific requirements. There are not many catalogues of generic requirements available. Also, with respect to the fact that the field has not reached a mature state yet, not all proposals need to be convincing. Therefore, the analysis of available catalogues should pay special attention to the rationale given for each requirement (Frank, 2013). Each requirement should be described and justified with respect to the purpose of the DSML.

### *2.9.2.2 Specific requirements*

Each scenario is related to a certain diagram type. To get an idea of what information should be represented in respective diagrams, one can start with a rudimentary graphical representation and then develop a list of questions that are related to the diagram. With respect to preparing for a corresponding modelling tool, it is helpful to specify for each

question whether it can be answered by a machine, by a human only, or in a partially automated way (P). Note that the stages ’develop use scenarios’, ’design exemplary diagrams’ and ’refine scenarios are interweaved. Each diagram type should be clearly described with respect to its purpose and its key concepts.

#### *2.9.2.3 DSML user*

The modelling expert or DSML user is any person who is skilled in modelling and is applying one or more modelling languages. The modelling expert uses modelling languages to create models and might provide guidance or explanations to domain experts. They should be involved in the evaluation of a modelling language to provide feedback (Laurenzi, 2020). Hence they will also be in direct contact with the language engineer. Karagiannis & Kühn (2002b) refer to this role as the process engineer, while in industry, it is likely to be called a modelling (senior) consultant. Specifically for this thesis, the urban mobility manager is regarded as a domain expert and DSML user. As such, this person is skilled in the mobility planning domain. The mobility manager can cooperate with a modelling expert to create models pertaining to the underlying domain (Völter, Stahl, Bettin, Haase, & Helsen, 2013). In the case of business-like modelling languages (e.g., BPMN, DMN or DSMLs) this role could intervene in the design of models or even creating them without the assistance of the modelling expert.

## 2.10 Model language extension

Zarour, Benmerzoug, Guermouche, & Drira (2019) describe that Business Process Model and Notation (BPMN) is a generic language that is often extended by researchers, either for dealing with processes of specific domains or for improving the language itself.

In a systematic literature review, they determine the current state of the art of BPMN extensions and identify the gaps that should be filled in this research area. After the collection and filtering of papers, 52 extensions were retained to be thoroughly examined and compared according to a set of criteria, including objective, targeted domain, conformity to the extension mechanism, demonstration, implementation, etc. It evaluates

and compares all BPMN extensions over the last four years, based on several criteria covering different aspects. Furthermore, the authors were able to verify the conformity of extensions as they were published after the introduction of the BPMN extension mechanism by the Object Management Group.

Braun, Schlieter, Burwitz, & Esswein (2016) describe BPMN as one of the very few languages that explicitly provides capabilities for its extension and provides multiple examples like one for Clinical Pathways.

Since there is a lack of modelling concepts required to represent CPS aspects in a business process model, Graja, Kallel, Guermouche, & Kacem (2016) presented BPMN4CPS, an extension of the BPMN 2.0 standard. They introduced a process logic using three parts: the cyber part, the controller, and the physical part. Each part has its own type of activities that can be performed. In addition, the extension included the CPS device roles, the properties of the real-world environment and the physical entities. Our future work will add support for a set of specific CPS temporal constraints and define profiles for domain-specific extensions.

In a case study, Graja et al. (2016) consider an example of a future health-care CPS using an ambulance drone. This system consists of three types of devices: 1) e-health sensors like the blood pressure sensor, the body temperature sensor, the glucometer sensor, and the electrocardiogram sensor (ECG), 2) the ambulance drone that transport a defibrillator, and 3) the ambulance vehicle. Additionally, this system needs to collaborate with the cyber components, which are the emergency service and the medical service. A specific instance of such a CPS would be a wireless network.

Due to the natural similarity of CPS and IoT, the next chapters display that a lot of effort towards integrating IoT in BPMN has been elucidated.

### *2.10.1.1 BPMN Task*

BPMN (OMG, 2013) differs between eight different types of tasks shown in Figure 13.

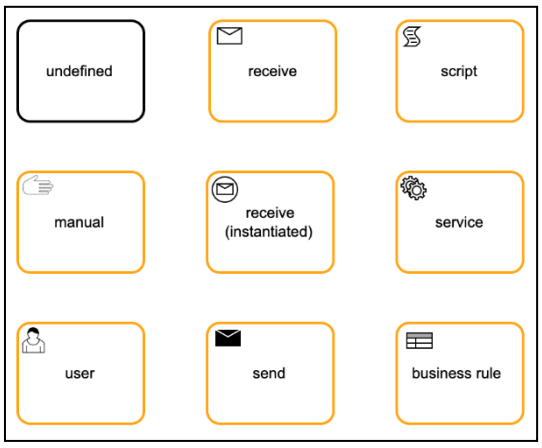



Figure 13 Different BPMN tasks implemented in Camunda.

When inspecting the different tasks above, to extend a task with predefined scripts, either the service or script tasks crystalise out. The OMG (2016) specifications provide the following information.

A Service Task is a Task that uses some sort of service, which could be a Web service or an automated application. A Service Task object shares the same shape as the Task, which is a rectangle that has rounded corners. However, there is a graphical marker showing two gears in the upper left corner of the shape that indicates that the Task is a Service Task.

A business process engine executes a Script Task. The modeller or implementer defines a script in a language that the engine can interpret. When the Task is ready to start, the engine will execute the script. When the script is completed, the Task will also be completed. A Script Task object shares the same shape as the Task, which is a rectangle that has rounded corners. However, there is a graphical marker in the upper left corner of the shape that indicates that the Task is a Script Task.

### 2.10.2 Recognisable effort in connecting IoT and BPMN

A Systematic Literature Review (SLR) conducted by Compagnucci et al. (2020), which shows, with 30 out of 47 (64%), that an extension of BPMN 2.0 is the most common approach to model IoT-aware business processes. A systematic mapping study (Torres, Serral, Valderas, Pelechano, & Grefen 2020) addressing some important issues for the near future, such as the lack of standardisation, further analysed 36 different solutions. Figure 14 presents an overview of the analysed paper. The colours indicate overlapping extensions.

| Source | Modelling Notation | Modelling Tool | Notation Usage |
|---|---|---|---|
| Forbig & Buchholz 2017 | CoTal | CoTaSE | D |
| Song et al. 2018 | CAPN | Not specified | D |
| Venkatakumar & Schmidt 2019 | S-BPM | Metasonic Build | D |
| | | | |
| Caracas & Bernauer 2011 | BPMN 2.0 | Oryx | D |
| Brouns et al. 2011 | BPMN 2.0 | Oryx | D |
| Ferreira, Martinho & Domingos 2014 | BPMN 2.0 | Not specified | D |
| Cherrier & Deshpande 2017 | BPMN 2.0 | Eclipse Modeler | D |
| Song et al. 2018 | BPMN 2.0 | Signavio | D |
| Martins & Domingos 2017 | BPMN 2.0 | Not specified | E |
| Domingos & Martins 2017 | BPMN 2.0 | jPBM | E |
| Mass et al. 2017 | BPMN 2.0 | Not specified | E |
| Friedow, Völker & Hewelt 2018 | BPMN 2.0 | bpmn.io | E |
| Meroni, Baresi, Montali &Plebani 2018 | BPMN 2.0 | Not specified | E |
| Schönig et al. 2018 | BPMN 2.0 | Camunda | E |
| Panfilenko 2018 | BPMN 2.0 | Camunda | E |
| Xu, Xu, Li, Lv & Liu 2012 | BPMN 2.0 | Not specified | E |
| Ruiz-Fernandez et al. 2017 | BPMN 2.0 | Bonita | E |
| | | | |
| Meyer, Sperner, Magerkurth & Pasquier 2011 | IAPM | Not specified | D |
| Meyer 2012 | IAPM | Activiti | D |
| Meyer, Ruppen & Magerkurth 2013 | IAPM | Signavio | D |
| Ruppen & Meyer 2013 | IAPM | Signavio | D |
| Martinho & Domingos 2014 | IAPM | Not specified | D |
| Meyer, Ruppen & Hilty 2015 | IAPM | Not specified | D |
| Chiu & Wang 2015 | IAPM | Not specified | D |
| Chen & Wang 2017 | IAPM | Not specified | D |
| Petrasch & Hentschke 2015 | IAPM | MagicDraw | D |
| Petrasch & Hentschke 2016 | I4PML | MagicDraw | D |
| Suri, Gaaloul, Cuccuru & Gerard 2017 | IoT-BPO | Signavio Core C. | D |
| Suri, Gaaloul & Cuccuru 2018 | IoT-BPO | Signatio Core C. | D |
| Yousfi, De Freitas, Dey & Saidi 2016 | uBPMN | Not specified | D |
| Yousfi, Bauer, Saidi & Dey 2016 | uBPMN | Not specified | D |
| Yousfi, Hewelt, Bauer Weske 2018 | uBPMN | Not specified | D |
| Graja et al. 2016 | BPMN4CPS | Not specified | D |
| Lee & Ma 2016 | BPMN-MDM | Not specified | D |
| Sperner, Meyer & Magerkurth 2011 | Sperner et al. | Not specified | D |
| Ramos-Merino et al. 2019 | BPMN-E2 | ARIS | D |
| Gao, Zaremba, Bhiri & Derguerch 2011 | Gao et al. | Not specified | D |
| Cheng, Zhao, Cheng, Chen & Chen 2019 | Cheng et al. | jBPM | D |
| Kozel 2010 | Kozel T. | Not specified | D |
| Sang & Zhou 2015 | Sang et al. | Activiti | D |
| Schönig, Ackermann & Jablonski 2018 | Schonig S et al. | Not specified | D |
| Grefen et al. 2019 | Grefen P. et al. | Not specified | D |
| Casati et al. 2012 | BPMN4WSN | Signavio Core C. | D |
| Tranquillini et al. 2012 | BPMN4WSN | Signavio Core C. | E |
| Sungur, Spiess, Oertel & Kopp 2013 | BPMN4WSN | Signavio Core C. | E |
| Mottola & Picco et al. 2019 | BPMN4WSN | Signavio Core C. | E |
| Appel et al. 1014 | SPU | Not specified | E |

| Source | Modelling notation |
|---|---|
| Al-alshuhai 2015 | Context-aware AD |
| Albreshne 2015 | GPL4SRE |
| Appel 2014 | SPUs |
| Baresi 2015 | |
| Bocciarelli 2017 | PyBPMN |
| Breitenbücher 2015 | SitME4BPEL |
| Bucchiarone 2009 | APFoL |
| Caracas 2011 | |
| Cheng 2019 | |
| Chiu 2015 | |
| Dar 2015 | |
| Domingos 2014 | |
| Domingos 2017 | |
| Dörndorfer 2018 | Context4BPMN |
| Friedow 2018 | |
| Gao 2011 | |
| Graja 2016 | BPMN4CPS |
| Kefalakis 2011 | APDL |
| Kim 2014 | |
| Kim 2016 | Process-aware IoT |
| Lee 2016 | BPMN-MDM |
| Maamar 2018 | PoT |
| Mandal 2017 | |
| Meyer 2013 | |
| Mottola 2018 | makeSense |
| Petrasch 2016 | I4PML |
| Sasirekha 2016 | |
| Schönig 2018 | |
| Seiger 2015 | |
| Serral 2015 | CAPN |
| Sperner 2011 | |
| Suri 2017 | IoT-BPO |
| Tu 2018 | IoTPM |
| Wang 2014 | CWfMS |
| Wehlitz 2017 | |
| Yousfi 2019 | UDABP |

Figure 14 Extracted papers adopted from Compagnucci et al. (2020) left and; Torres et al. (2020) right

Highlighted in light green Graja 2016 with BPMN4CPS is listed in both SLRs about IoT-aware business processes, which implicitly confirms the strong relationship between these two topics.

In case of Yousfi, Bauer, Saidi, & Dey (2016) a "ubiquitous" BPMN is suggested. A ubiquitous business process is a location-independent business process that turns its business environment into a source of data and/or a target of outcome with the least of human interventions (Yousfi, Freitas, Dey, & Saidi, 2015). Ubicomp capabilities include, for instance, Automatic Identification and Data Capture (AIDC). Yousfi et al. (2016) elaborate a case study three distinct examples are tackled: (a) model the usage of a Radio Frequency Identification tag on the car windshield in order to pass any toll plaza without stopping. (b) Model a location-based sensor gathering information and assign a taxi to the customer based on the customers current location. (c) Model the capturing of a music sample, identify the song and add the song to the customers music order. A proposed approach for case b is depicted in Figure 15.

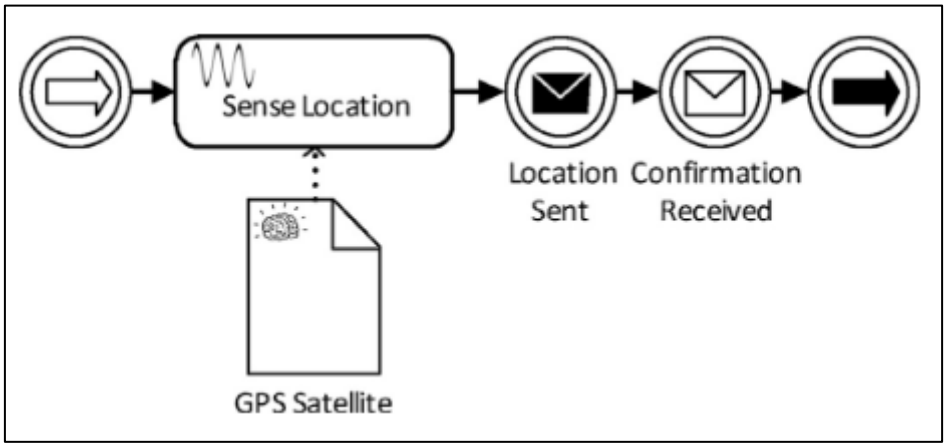



Figure 15 Sense location suggestion adopted from (Yousfi et al., 2016)

Coloured orange in Figure 14, Domingos & Martins (2017) propose to model and execute IoT behaviours by firstly parse BPMN XML files, then perform a syntactic and semantic validation, and to complete, translate BPMN into the programming language Callas. A similar approach, but for wireless sensor networks, is also suggested by Caracaș & Bernauer (2011), highlighted in grey colour in Figure 14.

Janiesch et al. (2017) state that the Internet of Things (IoT) refers to a network of connected devices collecting and exchanging data over the Internet. These things can be artificial or natural and interact as autonomous agents forming a complex system. In turn, Business Process Management (BPM) was established to analyse, discover, design, implement, execute, monitor, and evolve collaborative business processes within and across organizations.

The IoT provides many opportunities for the industry as well as for personal use through the meaningful yet dynamic interaction of humans, software, machines, and things. BPM is a well-established discipline that deals with the discovery, analysis, (re-)design, implementation, execution, monitoring, controlling and evolution of business processes. So far, both areas have been considered separately. (Janiesch et al., 2020) have formulated several challenges for the amalgamation of the IoT and BPM, which is deemed important to be tackled soon for the IoT to benefit from business processes and vice-versa. Before concluding, we would like to highlight a cross-issue, i.e., dealing with security and privacy issues. For example, privacy levels that exist at the sensors level might be different with respect to those at the BPM side. Janiesch et al. (2020) suggest that a full-disclosure approach should be avoided, especially in contexts where sensitive (i.e., personal) information is collected. The most relevant challenge, in this case, is the communication between the two worlds, each of them with corresponding privacy/security levels and policies.

While the IoT and BPM have been regarded as separate topics in research and practice, Janiesch et al. (2017) strongly believe that the management of IoT applications will strongly benefit from BPM concepts, methods and technologies on the one hand; on the other one, the IoT poses challenges that will require enhancements and extensions of the current state-of-the-art in the BPM field.

## 2.11 Summary and research contribution

An exciting aspect of this thesis is the intermingling of CPS, which has its own rapidly growing pool of researchers in those domains, especially engineering, and the use of

modelling languages in the business world, which puts a lot of emphasis on decision-making and more and more on the digitalisation of business processes.

Section 2.2 displays that when searching for robotic movements multiple domains are found as it is a very high-level search term. The results range from Advanced modelling of human movements using numerical optimisation (Colloud, 2016) to lab initiatives like RobertaLab and Scratch. Section 2.5 shows that while searching for CPS modelling, it is very likely to find research in electrical engineering or complex systems like fuel tanks of an aircraft (Derler et al., 2012).

CPS applications are extremely widespread (Khaitan & McCalley, 2015) and are rarely modelled within a business context. The systematic literature reviews (Compagnucci et al., 2020; Zarour et al., 2019) show that BPMN has been extended in various ways, and Graja et al. (2016) shows that this has already successfully been attempted with ambulance drones. Section 2.10.2 describes multiple variants of including sensors and data into BPMN, also the concept of using webservices is suggested in multiple instances (Domingos & Martins, 2017; Schönig, Ackermann, Jablonski, & Ermer, 2018). But during the search through this list no example has been found that executes physical movement.

For further development, the literature review shows that BPMN is a very solid starting ground as far as modelling languages go. To execute physical movement within a CPS, but excluded in the literature review, OMiLAB provides example projects mentioned in the introduction and further elaborated in chapter 5. After discussions with the supervisors, the Home to airport scenario firstly introduced in chapter 4 has been selected as a scenario residing in the mobility-planning domain. To establish a proof of concept requirements from scenario with regards to mobility have been derived and further explained in chapter 4, that then concludes the awareness of the problem phase

This thesis shall contribute a proof of concept – BPMN4MoPla – in order to support a mobility planner with a modelling tool and a feedback loop in the form of a cyber-physical system.

# 3 Research Design

This chapter contains a description of the design of this thesis. The goal is to prove or refute the thesis statement, find proposals to the given research questions in the form of artefacts and contribute to scientific research literature.

## 3.1 Research Philosophy

The way in which a research question is answered. Pragmatism research philosophy accepts concepts to be relevant only if they support action. Pragmatics "recognise that there are many different ways of interpreting the world and undertaking research, that no single point of view can ever give the entire picture and that there may be multiple realities" (Saunders, Lewis, & Thornhill, 2019). According to pragmatism research philosophy, the research question is the most important determinant of the research philosophy. Pragmatics can combine both, positivist and interpretivism positions within the scope of a single research according to the nature of the research question. This philosophy emphasizes on practical solutions and outcomes (Saunders et al., 2019) which is the case in this thesis as well in form of a proof of concept called BPMN4MoPla.

## 3.2 Research Approach

An inductive approach is chosen. Inductive reasoning is based on learning from experience. Patterns, resemblances and regularities in experience (premises) are observed in order to reach conclusions, or to generate theory (often in the form of a conceptual framework) (Saunders et al., 2019). This thesis starts by collecting data to explore the phenomenon of modelling CPS movements and builds a conceptual model that than is evaluated as the proof of concept BPMN4MoPla.

## 3.3 Research Strategy ï Design Science Research

A research strategy generally represents a bridge between a research philosophy and a choice of methods (Saunders et al., 2019). Hevner & Chatterje (2004) describe design-science as a strategy aiming to create new and innovative artefacts. This is also confirmed by Chatterjee & Hevner (2010), which consider the design science research strategy as useful to create a new model or tool. Moreover, design science research allows a flexible structure and qualitative data collection techniques (Chatterjee & Hevner, 2010). During this thesis, multiple artefacts are generated, including a DSML and other components in order to execute the DSML, which led to the decisions of using a Design Science Research strategy.

DSR is characterised by three cycles and includes an environment as well as a knowledge base, as Figure 16 demonstrates:

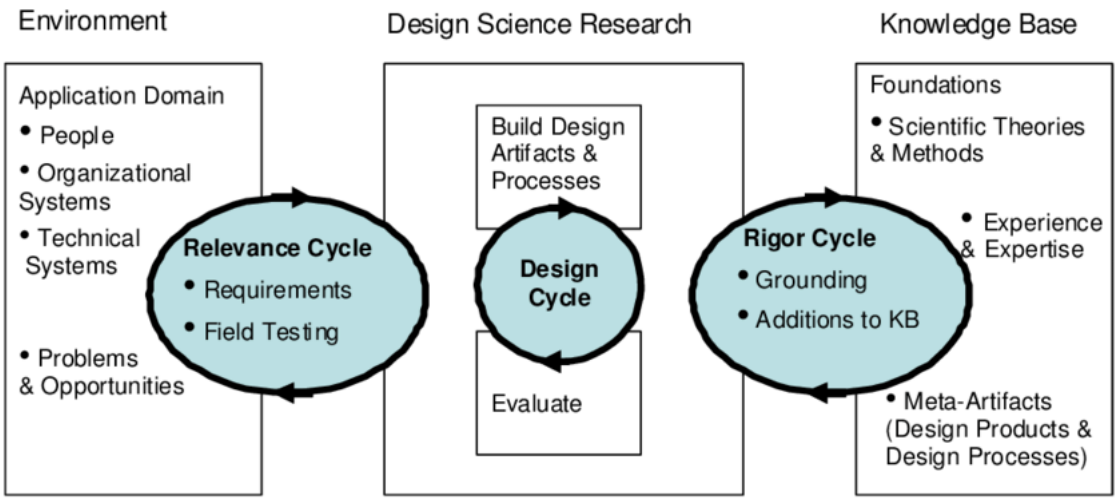



Figure 16 DSR cycles from (Chatterjee & Hevner, 2010)

On the left side, the environment contains the application domain, where different actors play together. Problems & opportunities are prevalent in the application domain. Hevner & Chatterjee (2010) explain that identifying and representing these opportunities and problems in an actual application environment is a main aspect of good design science research. Thus, the relevance cycle initiates DSR with requirements from the application domain and defines acceptance criteria for the DSR results. On the right side, the

knowledge base provides the foundation for DSR. Existing theories and artefacts are reused to develop new artefacts. The rigor cycle thus provides this past knowledge to ensure that innovation is happening. Newly developed artefacts will be added to the knowledge base by the rigor cycle. Lastly, the design cycle is the heart of any design science research project (Hevner & Chatterjee, 2010). In DSR, a new artefact or process is designed and evaluated. If needed, the design will then be further refined, or alternatives will be created. The design cycle can be explained in additional details with the DSR framework, which shows the reasoning in the general design cycle.

### 3.3.1 Design Science Research Framework

Vaishnavi and Kuechler (2007) extended the general design cycle, which was published by Takeda et al. (1990) in analysis by applying the cycle specifically to design scientific research and making the generated knowledge of design work explicit. Figure 17 visualizes the process steps within the DSR Framework.

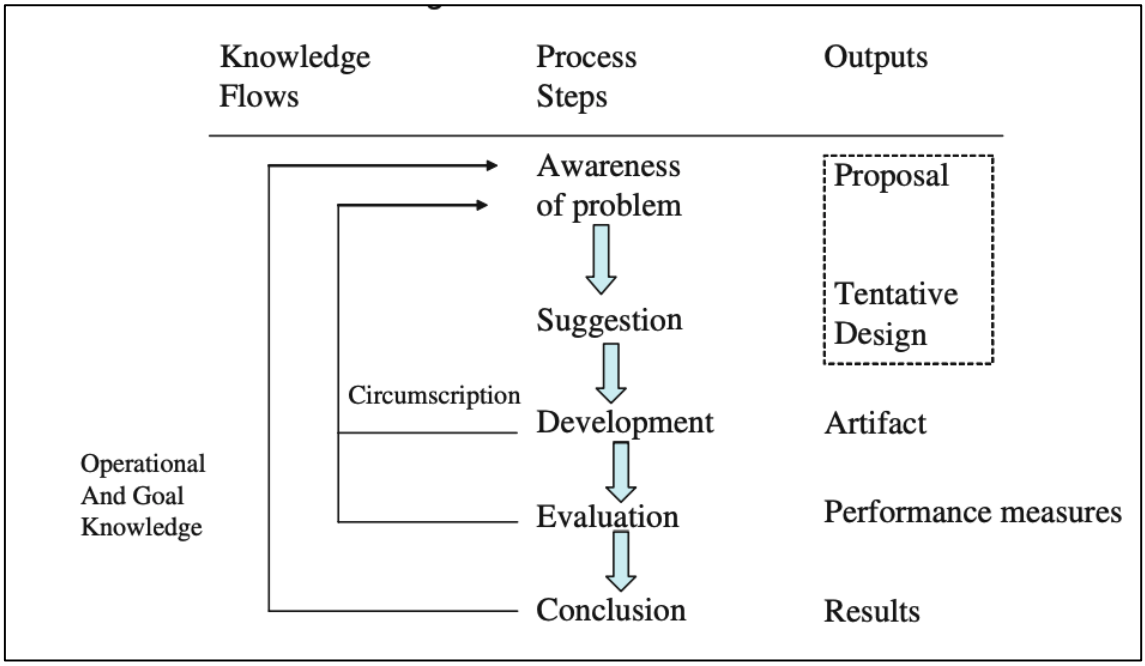



Figure 17 Reasoning in the general design cycle (Chatterjee & Hevner, 2010)

### Awareness of Problem

The first process step of the problem awareness is represented with the literature review in Chapter 2. Before literature review has been conducted, a thesis assignment has been discussed with all supervisors. To envision the scientific environment of designing a model which interacts with a Cyber-Physical system, an imaginary travel scenario has been fabricated. Chapter 4 Conceptualization further highlights key aspects extracted from the literature review, defines requirements categorised for the DSML, the modelling tool, and the CPS environment, which are then used to evaluate the produced artefacts and derive conclusions. Before the developing started the concepts of the used tools had to be analysed, especially the OMILAB environment and is then described in chapter 5.

### Suggestion

The suggestion phase determines which artefact should contribute to the fulfilment of specific requirements and defines the steps that should lead to the proof of concept. This means an extension of existing modelling languages is defined. Hevner & Chatterjee (2010) define the outcome of this phase as the tentative designs. Thus, the outputs created in chapter 6, Prototyping Innovations, include the specifics of the applicable scenario, the expected requirement coverage for BPMN, the planned metamodel extensions with details of the created elements and an idea of how a CPS can then be executed. To achieve this multiple tools ae tested in a first step. Then the BPMN metamodel is inspected and based on the findings of the literature review a possible extension of a service task is suggested. This shall allow the fulfilment of the previously defined requirements and allows the modeller to model and execute the defined scenario.

### Development

The development phase describes the implementation of the proof of concept in its details. It starts with the exploration of modelling the scenario in BPMN with four different tools to compare not only the tools but also the correspondence of BPMN with the requirements to gain a focus on the unfulfilled requirements of the DSML. In the next step, the metamodel was extended accordingly with the help of the ADOxx environment and made

available in a modelling tool. In the final proof of concepts, two solutions proposed in the suggestion phase are implemented to run the scenario in a CPS environment.

In chapter 7, development phase, the suggested outputs of chapter 5 are implemented and described. The implementation is taking place in the ADOxx environment that includes the tools: Scene2Model, Bee-Up, ADOxx Developing Toolkit, and the ADOxx Modelling Toolkit.

**Evaluation**

In chapter 8 The Proof of concept is then evaluated against requirements that are defined during the problem awareness of the DSR. Any deviations from these requirements must be noted and explained. Strengths and weaknesses of the artefact are recorded.

**Conclusion**

The results of the DSR are written down in the last phase. Gained knowledge is described and further needed research can be proposed. In chapter 9, the results of this work are summarized in a conclusion. An outlook for further research is described as well.

## 3.4 Modelling method

The AMME Lifecycle (Karagiannis, 2015), also referred to as the OMiLAB Lifecycle (Bork et al., 2019), introduces the application of agile principles in DSML engineering. The AMME Lifecycles aims to tackle evolving modelling requirements from the narrow domain and sophisticated model stakeholders. (Karagiannis, 2015) describes the applied agile principles as follows:

- Iterative development – to allow work cycle revisiting the same work items.
- Incremental development – successive usable versions are built upon previous versions.
- Version control – the output of each iteration is traceable across multiple versions.
- Team control – small groups of people are assigned to backlog items with shared accountability.

A proposed AMME framework and architecture by the work at hand introduces a support architecture of methodological, conceptual, and technological enablers with the modules **AMME Project Tracking System** to track modelling requirements and refer them to building blocks, **Reusable Asset Repositories** to ensure reusability across this and multiple projects, **Prototyping Environment** to support rapid prototyping, which is necessary for incremental and development. And the fourth module, **Deployment Channels** which supports granting stakeholders' access to developed prototypes (Karagiannis, 2015).

AMME is envisioned by the OMiLAB and aims to support meta-modelling research projects and communities. The following presented OMiLAB lifecycle defines a cycle of method iterations and is illustrated with five phases (Bork et al., 2019; Karagiannis, 2015).

The Creation phase – is a mix of knowledge acquisition and requirements elicitation activities that capture and represent the modelling requirements.

- The Design phase – specifies the metamodel, language grammar, notation, and functionality.
- The Formalize phase – aims to describe the outcome of the previous phase in non-ambiguous representations with the purpose of sharing results within a scientific community.
- The Develop phase – produces concrete modelling prototypes.
- The Deploy/ Validate phase – involves the stakeholders in hands-on experience and the evaluation process.

Figure 18 visualizes the above-mentioned phases.

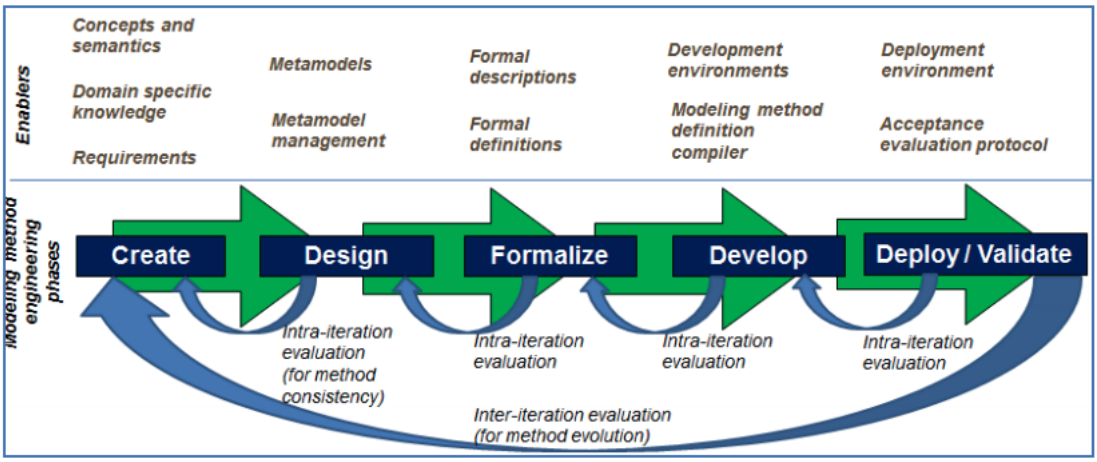



Figure 18 OMiLAB iterative lifecycle (Karagiannis, 2015)

Figure 19 graphically depicts the instantiation the AMME methodology (including the evaluation phase for the feedback loop) for the creation of BPMN4MoPla.

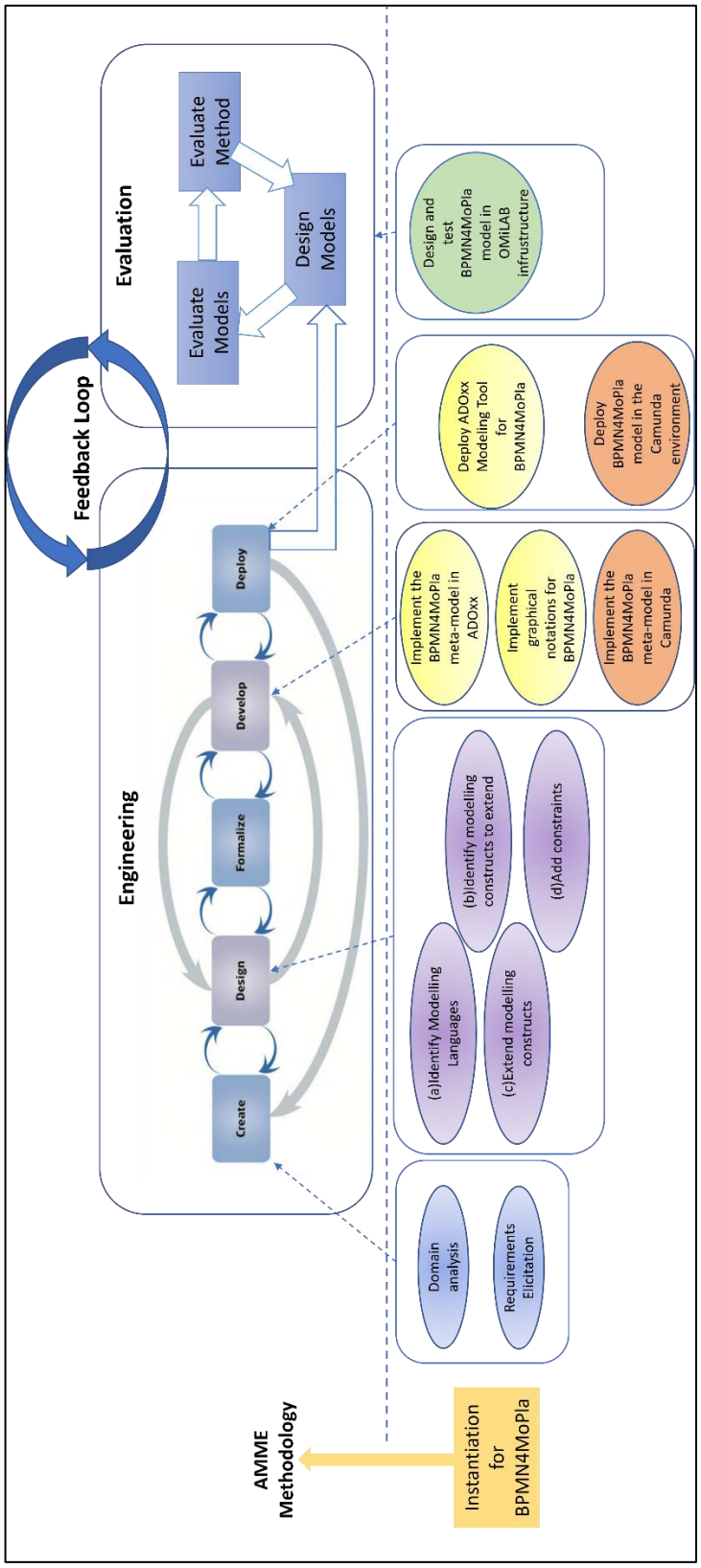



Figure 19 AMME instantiations for BPMN4MoPla adopted from (Laurenzi, 2020)

# 4 Conceptualization

This section defines the main abstractions and concepts of the addressed mobility planning domain and adapt them to a specific mobility planning scenario. When this thesis started, multiple discussion took place subjecting the search for a domain scenario.

The idea of a trip to the airport (mobility planning) came up, which has been hand drawn and shown in Figure 20.In a fictitious task, the role of a mobility planning manager is to model and demonstrate this with a CPS. This initial idea was to enable a rapid prototype and let the possibility open to adding different conditions and attributes further on.

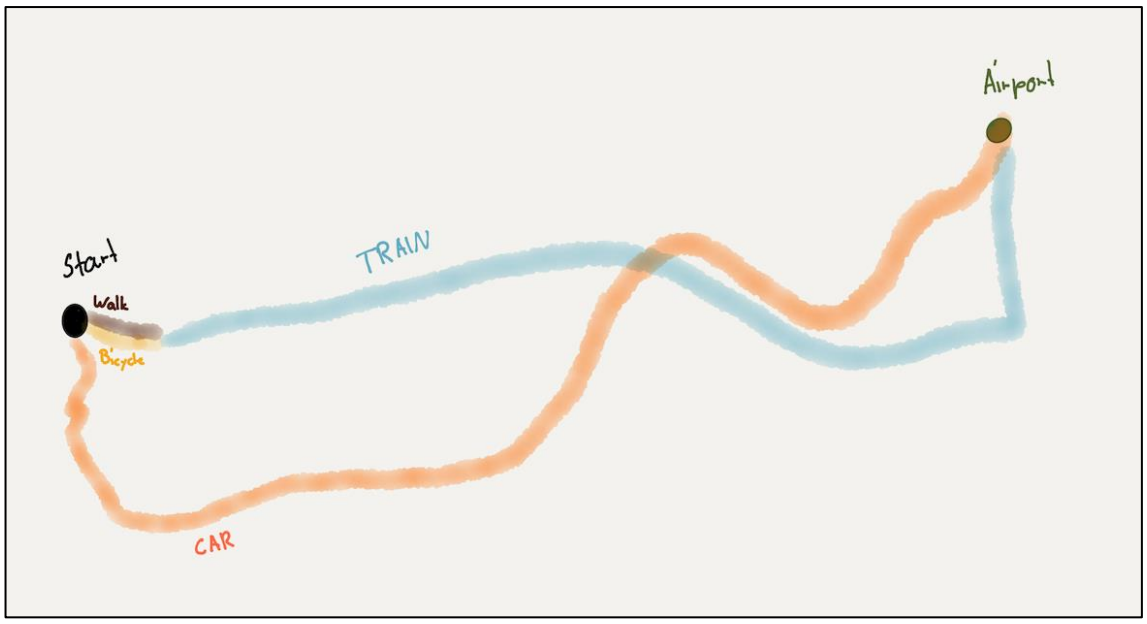



Figure 20 First draft of the scenario

For the first attempt following objects, representations and dependencies are intended:

- 1 Person intends to go to the Airport.
- The person has 4 transport modes:
  - Car
  - Train

    - Bicycle to the train station
    - The walk to the train station.
- Start: The house
- Destination: Airport

The scenario was created based on a real-world situation of the author that faces any time he goes from his home to the airport from Olten to the Zurich airport. The scenario is depicted in Figure 21 and consists of a plan with a home, an intermediary train station, and a destination. From the starting point, there are three different options of transportation modes, i.e., car, bicycle, or walk. Choosing the car implies going straight to the airport with no changes in the transportation mode. In contrast, when choosing either the bicycle or walking one can reach the train station, in which a change in the transportation mode is foreseen to reach the airport, i.e., by train. Additionally, based on certain user inputs, a transportation mode is chosen. Specifically for business trips, the car is taken to drive to the airport. For leisure trips. the user shall specify whether he or she has light or heavy luggage. In case of heavy luggage, the suggestion is to go on foot until the intermediate station (the train station), else by bicycle also until the train station. The variables Purpose and Luggage have been suggested by the author for this first scenario. Although other options, for example the weather, the duration of the travel, the distance to the train station, the availability and location of the car, would have also been suitable variables and might be integrated in a later point in order to enable more sophisticated decisions. As soon as a decision is taken (regardless of what the decision is), the robotic car shall perform actions by following one of the arrow paths illustrated in Figure 21.

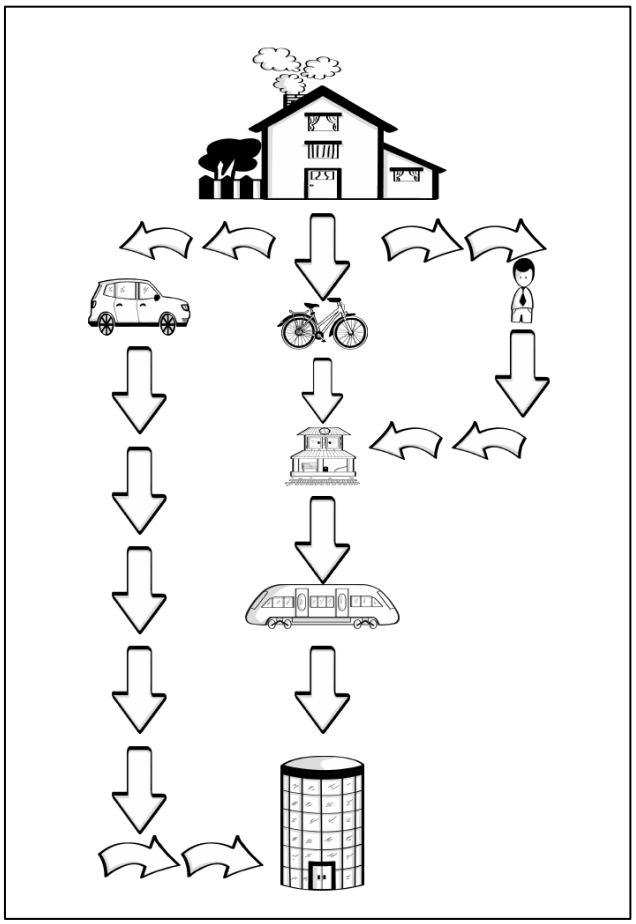


Figure 21 Scenario overview created with Scene2Model

The presented mobility planning scenario was analysed to derive requirements, which were grouped in three categories:

- *requirements for the DSML,*
- *requirements for the modelling tool,*
- *requirements for the CPS infrastructure.*

The requirements for the DSML are listed in Table 3. In the "Name" column, the numbers indicate subgroups within the DSML requirements: REQ-01X elaborates structural conditions and elements, REQ-02X groups the input of information that needs to be provided in order to complete a trip from home to the airport, and REQ-03X addresses the precautions for executing movements.

| Name | Description | Rationale |
|---|---|---|
| **REQ-010** | The DSML should accommodate business-like activities and decisions. | In order to decide which transportation method is chosen, a destination based on the variables purpose and luggage is required. |
| **REQ-011** | The DSML should accommodate model elements that are triggered by a process flow. | Decisions and Tasks, such as car movements, should be triggered automatically and subsequently along the process flow. |
| **REQ-012** | The DSML should accommodate the ability to represent different transportation movements | In order to reach the airport (destination) different movements need to be modelled. For example, drive straight, turn left or right. |
| **REQ-013** | The DSML should accommodate mechanisms such that a group of model elements can be incorporated into a model element having a higher abstraction. | Creating routes in sub-processes ensures consistent abstraction layers, and. reusability of these routes. |
| **REQ-020** | The DSML should accommodate constructs for specifying the travelling purpose. | In order to collect a basis for decisions the purpose for the trip has to be specified. |

| | | |
|---|---|---|
| **REQ-021** | The DSML should accommodate attributes to define the luggage that is brought to the trip. | In order to collect a basis for decisions, it has to be specified if light or heavy luggage is brought. |
| **REQ-022** | The DSML should accommodate the ability to differentiate between the transportation modes. | After the decision, it needs to be specified if the user travels by car, train, bicycle, or walk. |
| **REQ-030** | The DSML should accommodate scripts for the control of the car movement. For example, a script that execute the robotic car movement task. | Writing scripts in the model is time consuming and error prone, therefore the requirement of embedding scripts to execute functions, like follow the black line, arose. |
| **REQ-031** | The DSML should accommodate custom scripts for the control of the car movement. | In order to experiment with additional futuristic movements, for example a script that allows the car to turn for 180 degrees, an option to develop a script or add an external script should exist. |

Table 3 Requirements for the DSML

The requirements for the modelling tool were derived based on the *ease-of-use* (Tullis & Albert, 2013) criteria that a DSML shall ensure (see also (Laurenzi et al., 2017)) and is listed in Table 4.

| Name | Description | Rationale |
|---|---|---|
| **REQ-101** | The modelling tool should display (through the palette) all the set of modelling constructs used for the mobility-planning scenario, especially those for the transportation movements. | Frank (2010) claims that the more specific a DSML is the higher the productivity of the modeller is. |
| **REQ-102** | The modelling tool should provide interoperability with other modelling languages. | This requirement was added after the implementation of the metamodel in ADOxx as it was learned that a BPMN4MoPla is not supported with a workflow engine that could execute the model directly. |

Table 4 Modelling tool requirements based derived on the ease of use (Tullis & Albert, 2013)

In order to intermingle movements with the modelling languages, requirements regarding the robotic environment have been derived. Fulfilling these shall enable the model to connect to a CPS environment and execute physical movement. The requirements for the CPS environment are listen in

| Name | Description | Rationale |
|---|---|---|
| **REQ-201** | The CPS environment should provide connectivity to the cyber-physical space and be able to execute physical car movements. | After starting a movement task, the model needs to be informed the task is either completed or aborted in order to continue. |
| **REQ-202** | The CPS environment should foresee a robotic car with sensors capable of detecting the physical space and follow predefine routes | A vehicle to fulfil the robotic car actions is required. |
| **REQ-203** | The CPS environments should provide indicators to differentiate among the different transportation modes. | While defined as car, train, bicycle and walk the robotic car shall represent the chosen transportation mode. |

Table 5:

| Name | Description | Rationale |
|---|---|---|
| **REQ-201** | The CPS environment should provide connectivity to the cyber-physical space and be able to execute physical car movements. | After starting a movement task, the model needs to be informed the task is either completed or aborted in order to continue. |
| **REQ-202** | The CPS environment should foresee a robotic car with sensors capable of detecting the physical space and follow predefine routes | A vehicle to fulfil the robotic car actions is required. |
| **REQ-203** | The CPS environments should provide indicators to differentiate among the different transportation modes. | While defined as car, train, bicycle and walk the robotic car shall represent the chosen transportation mode. |

Table 5 Requirements for the CPS environment

# 5 OMiLAB environment

This chapter introduces software hardware and design principles adapted from the OMiLAB environment focussing on the used modelling platform ADOxx, the Bee-up tool, as well as a introduction to the OMiLAB infrastructure and an example project.

## 5.1 ADOxx modelling platform

ADOxx has been successfully used in academia and industry for over two decades. The platform comes with a rich set of domain-independent functionality like model management, user management, and user interaction. What is left to be done for metamodel developers is to 1) configure the specific metamodel by referring its concepts to the meta-metamodel concepts of ADOxx; 2) provide a visualisation for the concepts and combine them into logical chunks, i.e., ADOxx model types; and 3) realize additional functionality like model transformations, queries, or simulations on top of the modelling language (Bork, 2018).

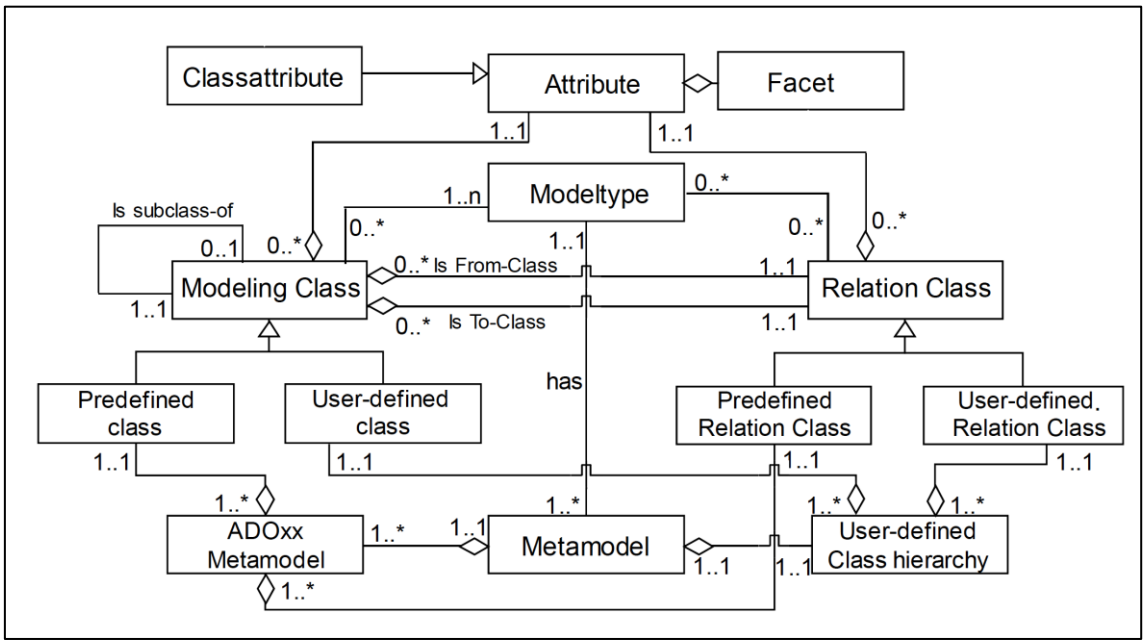



Figure 22 Excerpt of ADOxx meta2model adapted from (Bork, 2018)

A metamodel realized with ADOxx is composed of model types which themselves comprise predefined and user-defined modelling classes and relation classes depicted in Figure 22. Following a graph-based structure, modelling classes refer to nodes and relation classes to edges between nodes. Attributes define the semantics of all ADOxx classes. Functionality in ADOxx is attached to predefined abstract meta classes of the ADOxx meta-metamodel, depicted in Figure 22. When defining an inheritance relationship between domain-specific concepts and predefined abstract meta classes, the functionality is inherited. Consequently, the metamodel design decisions determine the functionality of the resulting modelling tool.

Figure 23 briefly introduces the most important ADOxx meta classes. Bork (2018) describes ADOxx meta classes are either static (prefix 'S') or dynamic (prefix 'D'). The former employs a tree-based structure for hierarchies between static classes, while the latter employs a graph-based structure for realizing simulations.

| Meta Class | Description |
|---|---|
| D_Aggregation | Every modeled object 'a' having its x/y coordinates within the drawing area of any container 'b' has the relation *'a' is-inside 'b'*. Moreover, subclasses come with a self-defined "drawing area" by means of resizeable rectangles. |
| D_Swimlane | Also provides the "is-inside" relation but the "drawing area" is limited to strict horizontal or vertical rectangles. |
| D_Event | Encapsulates all nodes of a graph necessary for its simulation. Subclasses are e.g., D_Start, D_Subgraph, D_Activity, D_Decision. |
| S_Group | This class represents a node in a tree structure. |
| S_Aggregation | Special kinds of nodes in a tree structure. Similar semantics as for the dynamic counterparts. |
| S_Swimlane | |
| S_Person | Implements person-dependent aspects like wages and working hours. |

Figure 23 Most important ADOxx meta classes extracted from (Bork, 2018)

## 5.2 OMiLAB Bee-Up tool

Karagiannis et al. (2016) focus on the metamodeling approach for the hybridization of BPMN, ER, EPC, UML, and Petri Nets within a single modelling method identified as Fundamental Concept Modelling Language (FCML), with a proof of concept named Bee-Up implemented in OMiLAB.

Bee-Up hybridizes several modelling languages in one prototypical implementation/tool. It allows the creation of models in modelling languages commonly used in different domains: Business Process Model and Notation (BPMN), Event-driven Process Chains (EPC), Entity-Relationship models (ER), Unified Modelling Language (UML) and Petri Nets. Processing capabilities, like process simulation or SQL generation, are also available in Bee-Up. They provide additional showcases on how models can be used.

The "IMKER" Case Study - Practice with the Bee-Up tool (Karagiannis, Burzynski, & Miron, 2017) reports on a project developed by the NEMO Summer School, where a text written by beekeepers was used to derive different views such as process, data, system and production views and represent them in different models. The languages used were

BPMN, UML, ER, EPC and Petri Nets, as mentioned above. Figure 24 provides an overview of examples of some model types.

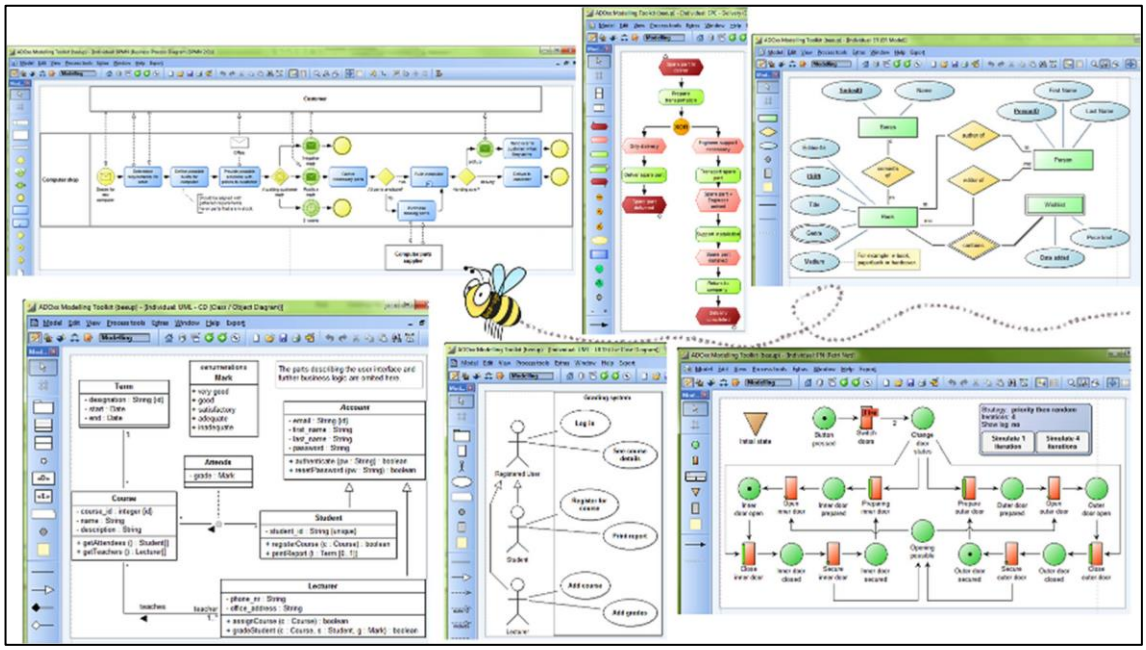


Figure 24 Examples of some model types (Karagiannis et al., 2017)

The entire bee-up environment is implemented through different artefacts and resources. At its heart is the modelling tool, which is realised on the ADOxx platform due to its free availability and ease of use. Thus bee-up uses the ADOxx platform's meta²-model to define the available concepts and inherits the architecture of the ADOxx platform for deployment as a standalone tool, parts of which are shown in Figure 25.

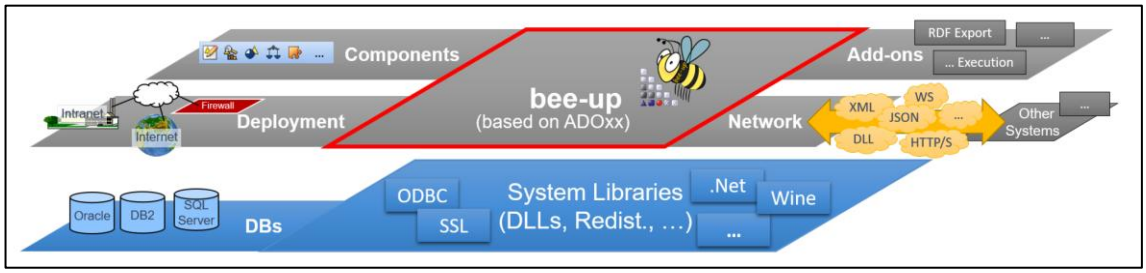



Figure 25 bee-up architecture as an ADOxx based tool (Burzynski & Karagiannis, 2020)

The specific types of models are realised through extending the ADOxx platform its core metamodel with the necessary classes, relations, and attributes, leading to an integration

of the languages through shared super-classes. The current bee-up implementation has around 200 classes and around 50 relations in total and an even larger number of attributes. The Bee-Up import for ADOxx or the standalone tool is freely available[4].

### 5.2.1 The OMiLAB community

The central space of OMiLAB resource hosting is openly available[5] for multiple projects, events, services, tools, and downloadable content. The referred content is accessible without any form of interaction. For interested parties, OMiLAB prescribes three levels of commitment: Member, Associated Partner and Operator of an OMiLAB node (Bork et al., 2019). Table 6 presents the three brought up roles.

| | |
|---|---|
| Member | Mostly individuals that contribute with their knowledge and a modelling method, incentivized with a potential realization of a prototype supporting their research. |
| Associated Partner | An Associated Partner can consist of a research group or university department that actively advocates, motivates courses to use OMiLAB resources and contribute to OMiLAB events as well as to help to advance the international visibility by publishing the resources emerging from the community. |
| Operator | The Operator status signifies the establishment of the physical infrastructure and knowledge to perform training and host OMiLAB resources. |

Table 6 OMiLAB level of commitments (Bork et al. 2019)

[4] https://austria.omilab.org/psm/content/bee-up/download

[5] www.omilab.org

### 5.2.2 Roles within OMiLAB

The general concept is to have multiple roles, visualized as the innovation community carry out the conceptualization process, which leads from modelling to a fast prototype. The roles start with a domain expert to bring the knowledge and conclude with the Digital Product Engineer or the End-User. The succeeding list elaborates on the roles and their tribute to the conceptualization process.

A **Domain Exper**t is required to input the domain knowledge that must be embedded in all building blocks. A **Metamodel Designer** is responsible for the modelling language definition. A **Modelling Method Engineer** uses the AMME framework to build the modelling method by extending the language with a modelling procedure and model-driven functionality. **Digital Product Engineer** applies the modelling method for a selected domain or case towards the goal of realising a digital product that employs the modelling tool as its "knowledge engine". This is a knowledge acquisition enabler that interoperates with other artefacts, for example Internet of Things components or microservices either through direct interoperability mechanisms or through the model base. digital product prototypes that employ models as a knowledge source can thus be built or experimented upon. The **End-user** is the person who draws the model for its intended purposes, which can involve simulation, reporting, model queries or a diagrammatic representation of a real-world situation (Bork & Miron, 2017). Figure 26 provides a summarised overview of the OMiLAB Digital Ecosystem.

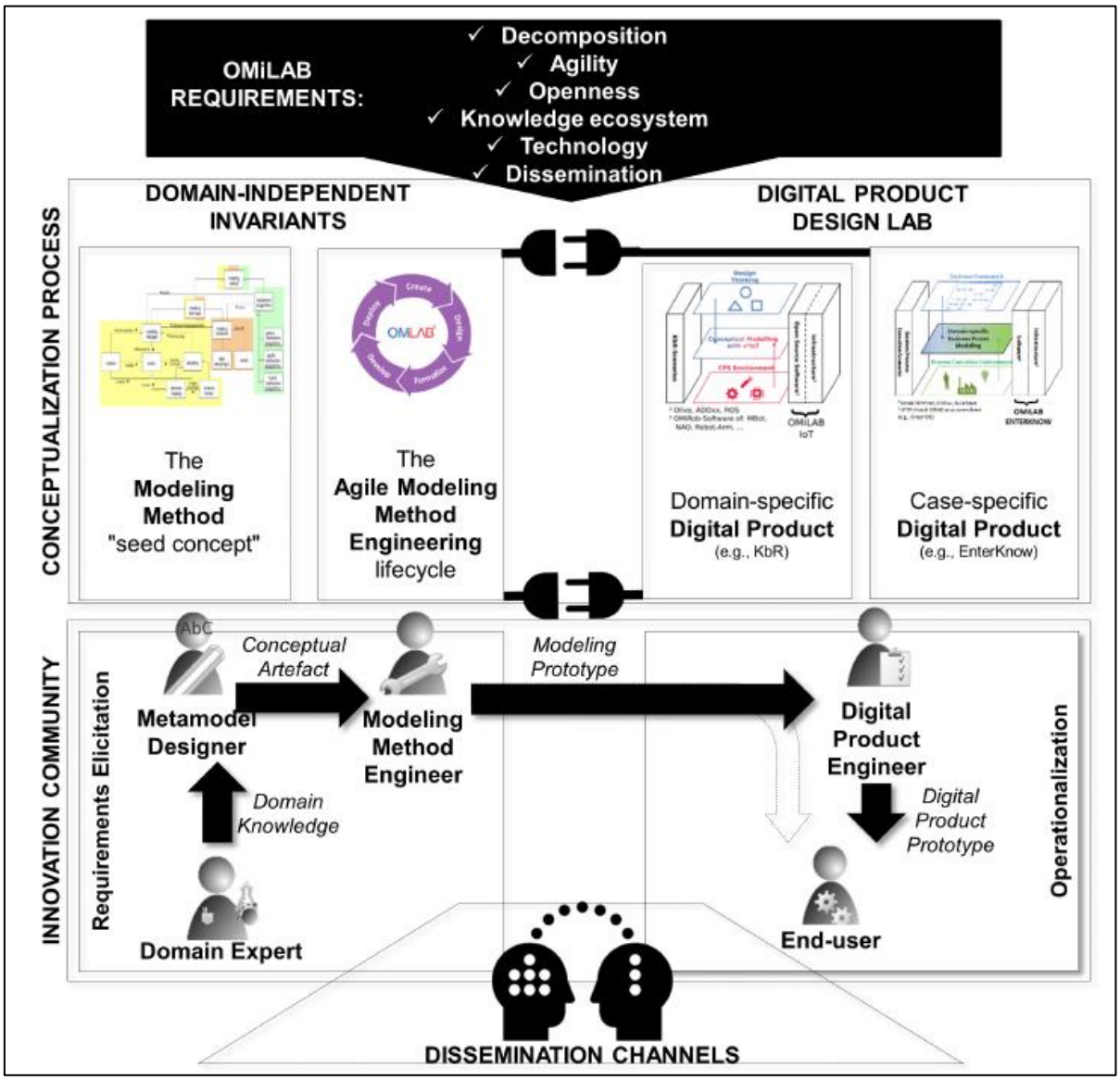



Figure 26 OMiLAB Digital Ecosystem according to (Bork et al., 2019)

### 5.2.3 mBot and the street layout

The Robot Car called mBot from the manufacturer Makeblock equipped with the corresponding hardware on top form the referred CPS (Karagiannis & Muck, 2017). The mBot has two bigger wheels and a small caster wheel. Each of the two larger wheels has an individual motor and can therefore be controlled separately. The standard version has the following sensors:

- ME Ultrasonic sensor allows measuring the distance to objects in front of the mBot.
- Me Line Follower Sensor allows the mBot to detect if it is currently on a dark or light background. The Me Line Follower Sensor also detects if the mBot is on the edge of different undergrounds. As an energy resource, a battery pack with 4 AA batteries is used.

For the mBot a REST interface is accessible via a Raspberry Pi, which allows a user to call REST functions from a WLAN connected device. These functions are implemented in JAVA and can control the engines, read sensor data, generate sounds, light LEDs and more. In addition to the before-mentioned sensors and the Raspberry Pi, this mBot is equipped with Raspberry cameras that can be used for streaming.

## 5.3 Other OMiLAB project

The following OMiLAB example project is published on their website[6] and is called XOMiAC1. The main objective of this example project is to simulate a delivery process in warehouses. For this purpose, the mBot represents the delivery item, and a robot arm simulates the pick-up from the correct project of the warehouse and then places it on the delivery item, depicted in Figure 27. The process consists of taking the correct ingredient,

[6] https://austria.omilab.org/psm/omirob

handing it over to the delivery part, which moves to a production area and back again to receive the next ingredient.

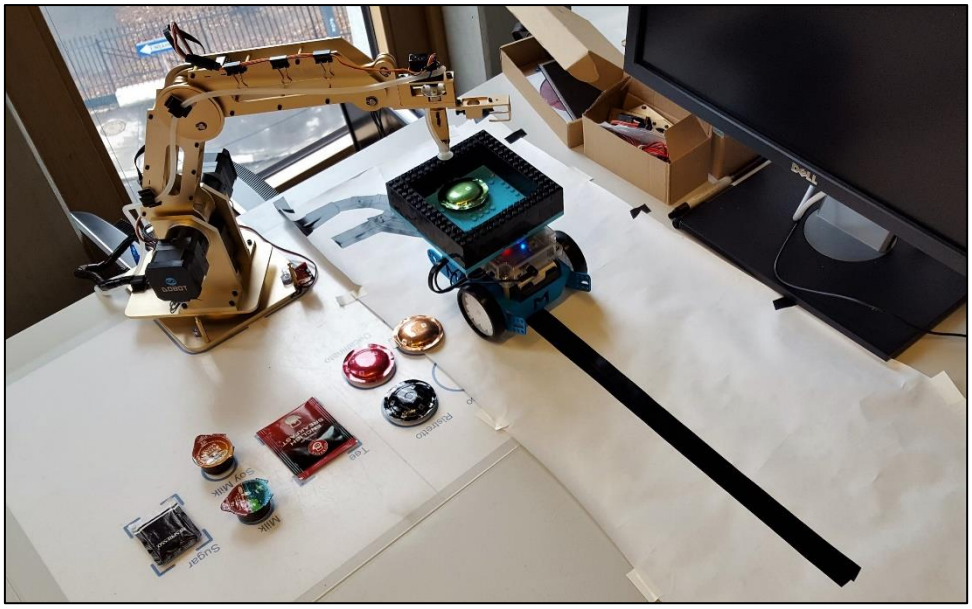


Figure 27 Warehouse process project from OMiLAB

The realisation of this sub-operation is based on the mBot Java project available here (https://gitlab.dke.univie.ac.at/OMiROB/MBotJava). The robot arm as well as the car were controlled by this program via REST requests.

## 5.4 Creating a modelling language with ADOxx

In a small Hands-On project, ADOxx refers to the following 15 steps to create a simple ER-Diagram model. This steps were later used in order to create a DSML from scratch handling robotic movement.

**1) Define model type:**

- Define the model type in the Modi text field, enter: 1MODELTYPE "ER-Diagram".

**2) Create a new (abstract) class, inherited from __D-construct__:**

- Select super class__D-construct__.and create a new class.
- Name new class _ER-construct_ (here we apply the name convention for abstract classes: class name starts and ends with an underscore sign)

**3) Make the class _ER-construct_ abstract (effect: class cannot be instantiated in the Modelling Toolkit):**

- Unfold the class_ER-Construct_ to see all its attributes. Double click the attribute ClassAbstract and check it.

**4) Create a new classes named Entity and Relation, inherited from _ER-construct_**

- Analogously to step 2)

**5) Create new class named Attribute, inherited from __D-construct__**

- Analogously to step 2)

**6) Configure GraphRep of Class Entity:**

- Open the GraphRep Editor of the class entity and type in the following code snippet:

```
GRAPHREP
```

```
PEN w:0.05cm
RECTANGLE x: - 2cm y: - .5cm  w:4cm  h:1cm
ATTR "Name"  x:0cm  y:0cm  w:c:3.5cm  h:c:1cm  line  - break :rigorous
```

### 7) Configure GraphRep of Class Relation:

- analogously to step 6)

```
GRAPHREP
PEN w:0.05cm
FILL  color:white
POLYGON  x1: - 1.9cm  y2:1.6cm  x3:1.9cm  y4: - 1.6cm
ATTR "Name"  w:c:2.5cm  h:c:0.8cm  line  - break :rigorous
```

### 8) Configure GraphRep of Class Attribute:

- analogously to step 6)

```
GRAPHREP
PEN w:0.05cm
ELLIPSE x:0.00cm  y:0.00cm  rx:3.00cm  ry:0.80cm
ATTR "Name"  y: - 0.2cm  w:c:2.8cm  h:t
# Advanced Attribute  Dependent Graphical  Notation
AVAL k: "key"
IF  (  k  =  "1" )
{
   LINE  x1: - 1.7cm  y1: - 0.5cm  x2: - 1.7cm  y2:0.5cm
   LINE  x1: - 1.5cm  y1: - 0.6cm  x2: - 1.5cm  y2:0.6cm
}
```

### 9) Add a new attribute (Integer) to class Attribute:

- Select class Attribute and add a new attribute “key” as an Integer-type.

### 10) Add a new attribute (Enumeration) to class Attribute

- Similar to step 9): attribute name: datatype, type: Enumeration
- Add the items String, Integer, Float to the value range and apply.

### 11) Configure AttrRep (Metamodel) attribute of class Attribute

- Double click AttrRep (Metamodel) & Set standard value.

```
NOTEBOOK
CHAPTER Description"
ATTR "Name"
ATTR "key"    ctrltype:check
ATTR "datatype"     ctrltype:dropdown
```

### 12) Include classes to model type ER-Diagram

- Go to the Modi attribute in the “Library attributes”.
- Add the following lines to include the classes Entity, Relation, Attribute in the model type ER-Diagram.

```
INCL "Entity"
INCL "Relation"
INCL "Attribute"
```

### 13) Create relation class "has"

- Go to Class Hierarchy and double click folder Relation class.
- Enter name: has, from-class: _ER-construct_, to-class: Attribute.
- Include relation class must model type ER-Diagram:

```
INCL "has"
```

### 14) Configure GraphRep of relation class "has"

```
GRAPHREP
EDGE
START
FILL   color:black
ELLIPSE x: - .1cm  y:0cm  rx:.1cm    ry:.1cm
```

### 15) Create relation class "relates"

- Runs analogously to step 13)
- name: relates, from-class: Entity, to-class: Relation.

These quick 15 steps lead to different classes, including an Entity a Relation and an Attribute class. Those can be connected with the relation classes has or relates. Figure 28 displays an overview of the achieved model tool.

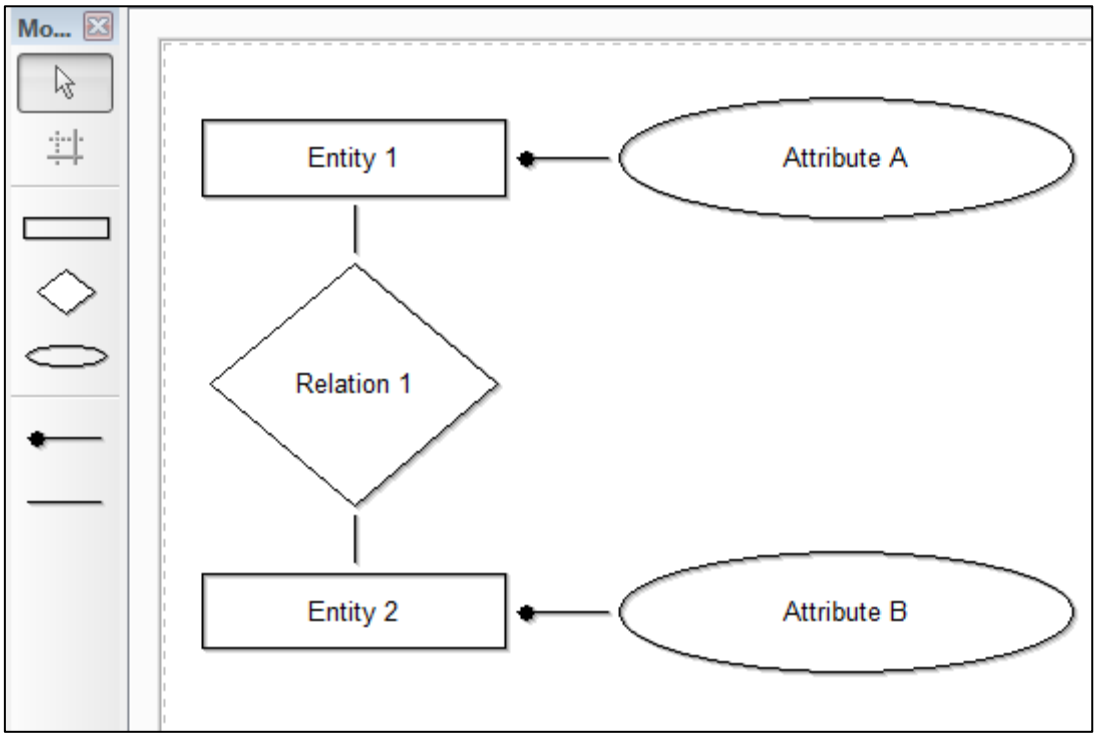



Figure 28 Quick ER model elements established as language from scratch (ADOxx, 2021)

## 5.5 Conclusion

This chapter is mostly separated from the literature review. The reason for this is that OMiLAB should not be the main part of this work but should be considered and explained as a supporting environment.

As a personal side note, the author highly recommends a visit to the OMiLAB site. The variety of projects may inspire future projects and display the variety of how small robots can be incorporated into diverse ventures. Regardless of whether the rest of the diverse ADOxx tool suite is used.

# 6 Prototyping Innovations

In this chapter, the planned steps that eventually lead to the proof of concept are presented. Conform with the DSR (Chatterjee & Hevner, 2010) this chapters produces a tentative design of the proof of concept that is then developed in the next chapter.

## 6.1 Scenario adaptation

After the examination of the OMiLAB infrastructure and examples, the scenario displayed in Figure 21 in Chapter 4 Conceptualization has been adapted further and is depicted in Figure 29. Road templates that can later be printed out and used as a road layout for the robotic car replaced the arrows. With the ulterior motive that one does not have to set the car manually from the target position to the start position, there is also a path back home. The colours red, yellow, green, and blue have been chosen to represent the transport mode differentiation, not least because the LED was chosen to represent the mode of transportation. As shown in Figure 29, yellow was assigned for the car, red for the walk, green for the bike, and blue for the train.

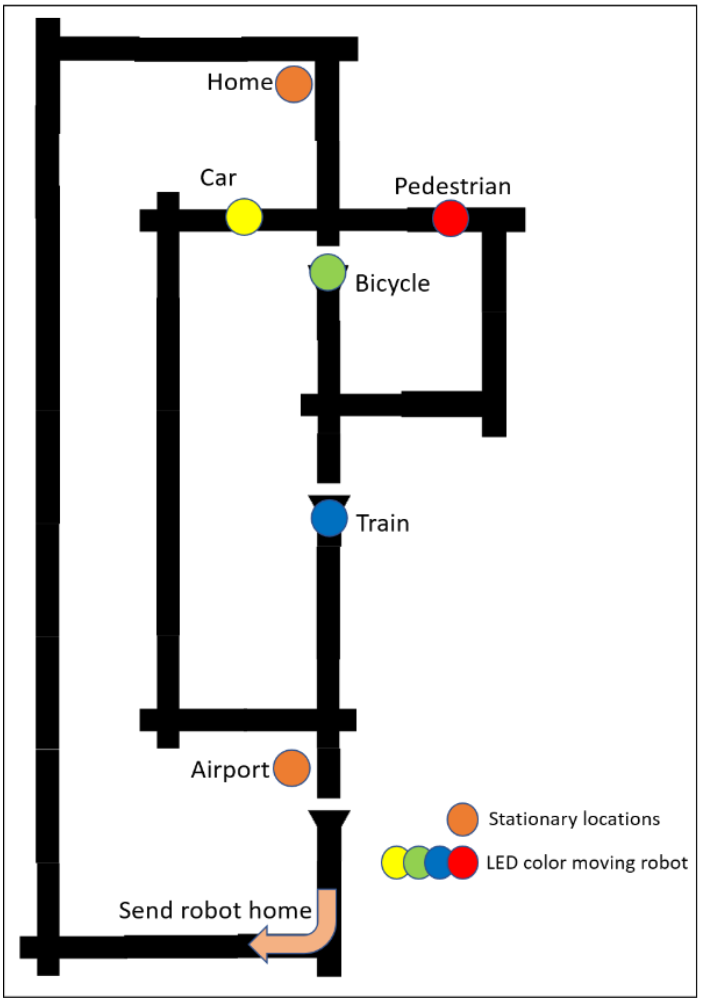



Figure 29 Mobility planning scenario

The proof of concept for BPMN4MoPla consists of multiple parts, including the creation of a DSML and the transformation to a format where a workflow-execution can be demonstrated. The street layout showed in Figure 29 that graces the floor contributes to the fulfilment of REQ-202 by providing an environment, used in example projects described in chapter 5, the sensors of the mBot by Makeblock recognize.

## 6.2 BPMN modelling Tool Comparison

The BPMN modelling standard was chosen as it addresses REQ-010 to REQ-013, REQ-020, and REQ-021. With intermediate events, BPMN allows to model foreseeable events.

In the mobility planning domain where events like accidents jam and more can interfere with a planned route, this might be very beneficial in the future. Along this, the support of complex decisions, which can be further described as multilevel decisions, e.g., in the form of decision tables might find good uses, which further underlines BPMN as a suitable starting candidate with a wide userbase.

In a first evaluation, a few tools shall be explored to go forward with the thesis and the modelling. The following two aspects have been explored:

- A tool that can model BPMN 2.0.
- If possible, one that supports a workflow engine to execute the model (REQ-021)

The modelling is planned to initially do with various tools, including Visual Paradigm, Bee-Up, Camunda Modeler and the ADOxx Modelling Toolkit. This choice resulted due to recommendations and usage of them at FHNW as well as Unicam and to indicate how the same notation can appear different with different tools. Whether workflow engines are also available for this is not taken into consideration when modelling in a first attempt but kept in mind. This step shall further assess to what extend the following requirements can be fulfilled with BPMN: REQ-010, REQ-011, REQ-012, REQ-013, REQ-020, REQ-021, REQ-022, REQ-030, REQ-031.

## 6.3 Adapt BPMN and create a DSML from scratch

The literature review, more specifically chapter 2.10, shows research that is conducted to adapt and use BPMN in all possible constellations. The goal of this step is to show that BPMN can be used as a good point of entry.

The aim is to propose solutions to avoid direct technical commands in the model. One of the main reasons is that REQ-012 cannot be fully fulfilled with the basic BPMN language since the transportation movement can only be displayed within the name of the element. This can be done by extending the modelling language or by the more radical approach of developing a custom modelling language.

## 6.3.1 BPMN Extension

With the focus of addressing REQ-012, REQ-022, REQ-030, REQ-031 BPMN shall be extended. Good candidates for it are the BPMN tasks that describe some sort of automation like the Service Task and the Script Tasks. A *Service Task* is designed to invoke external services, for example, a web service or an application. The *Script Task* typically incorporates executable scripts, e.g., for arithmetic calculations. In this case, a service task is rather more appropriate for extensions as the modelling constructs for the movement should rather invoke external services for the control of the robotic car. Figure 30 shows an excerpt of the BPMN metamodel with the newly added modelling construct named *CarMovement* as a sub-class of the BPMN Service Task.

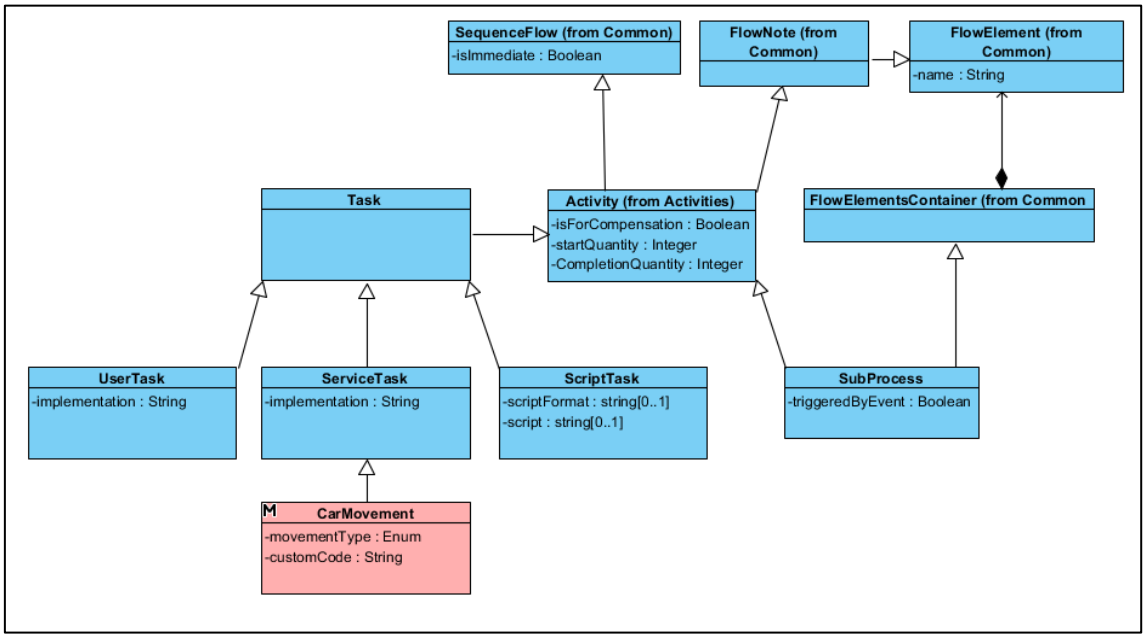



Figure 30 Excerpt of the extended BPMN metamodel

In line with REQ-022 class the task CarMovement needs to provide at least a movement type attribute as well as a transportation mode identifier attribute. To achieve that, the variable for movement type defines the type of movement in any direction as well as set the transportation mode. When choosing to set the transportation mode, another attribute must be provided to set the transportation mode to either car, train, bicycle, and walk.

Figure 31 is providing an overview and later specified in a more detailed view in Figure 33.

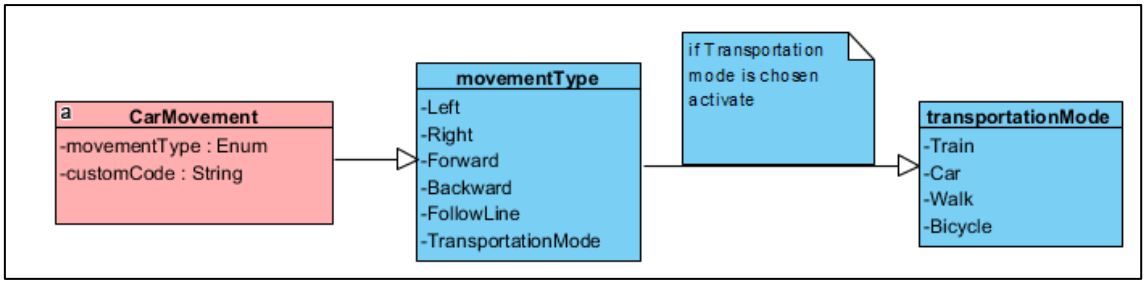



Figure 31 Conceptualization CarMovement element

### *6.3.1.1 Adding Attributes to the BPMN Task Element*

Alternatively, and in an experimental approach, attributes can be added to the standard BPMN task from the Bee-Up library without implementing a new class. However, the author is aware that this approach adds even more attributes to a task that is already clotted with attributes. Nevertheless, the following ideas of attributes came together concerning REQ-010, REQ-012, REQ-020- REQ-022, REQ-030, REQ-102 and the CPS environment requirements. During the development, different options of how to display and specify the following attributes have to be explored.

- A dropdown menu with the possible vehicle movements (REQ-012 and REQ-030).
- A dropdown to identify the transportation mode (REQ-022)
- A radio button to select the purpose of the trip (REQ-020) and the intended luggage to bring along (REQ-021).
- An option to insert custom functions that fulfils the later raised REQ-031.

Instead of adding attributes to the task element, there are other variants, such as the possibility to develop a completely new language next to the premises to add a new sub-class element as conceptualized in Figure 30.

## 6.3.2 DSML from scratch language creation

This section provides an insight into developing a DSML from scratch, which can have the advantage that every element is necessary for its existence if the due diligence before is done accordingly.

### *6.3.2.1 DSML planning*

The following main constraints were derived from REQ-012, REQ-022, REQ-030; REQ-031 and REQ-101 shall specifically be addressed during this sub-chapter.

- CPS elements are visually easy recognisable (REQ-012).
- The modeller must be able to choose quickly between the following basic movements (REQ-101):
    - Jump gap, follow line, turn left and right, turn the LED on and off.

To design a modelling language, the basic elements for a model were first worked out and shown in Figure 32. The basic construct is a start & end point connected through a sequence flow. To make decisions based on the scenario, an element must be defined in order to specify variables (REQ-020 and REQ-021) and a gateway that can split the sequence flow in different directions. The split of a sequence flow allows a modeller to display (at minimum) two possible outcomes of a decision. It is recognisable that the elements are relatable with BPMN and thus considered easier in the application, if BPMN is known beforehand.

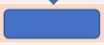
| Element | Description | Form |
|---|---|---|
| Variable input | Manual input |  |
| Exclusive Gateway | Adapt from BPMN/ Flowchart |  |
| Movement Task | Task with preset specifications |  |
| Sequence Flow | Adapt from BPMN |  |
| Start & End Point | Adapt from BPMN |  |

Figure 32 Basic construct of the from scratch DSML

Further on, a new element must be designed that covers the car movements (REQ-012, REQ-030) and handles the differentiation of the transportation mode (REQ-022) by

adjusting the LED colour. The reason why both actions have been allocated in one element is that the principle of the execution, calling an API (REQ-201), is consistent. Additionally, the appearance of the element shall change according to the selected movement types (REQ-012). For the graphical representation, arrows to the left, to the right and down are chosen to represent the actions turn left, turn right and moving forward. The arch (or rainbow) represents the movement “jump a gap”. This is important for the street layout, depicted in Figure 29, where gaps are used to bring the mBot to stop. For the LED control elements, a circle is chosen to turn on the LED and the circle with a cross symbolizes an LED off sign. A custom and a flowchart reference are visualized by a capital “C” for custom or the word “Flow” for a flowchart interreference. That planned when the LED Movement-type is chosen a second dropdown should then the modeller allows to choose the colour. In accordance with the ease of use (Tullis & Albert, 2013) this field should only be editable if the LED Movement-Type is chosen, else it should be locked. Similar principles are then applied to the field Flowchart process and Custom movement code that shall be only modifiable if “Custom” or “Flowchart” are selected as Movement-Type.

Adapted from Figure 31, Figure 33 displays the conditions for the planned Movement Task element.

Although displayed in Figure 33, the External AdoScript as well as the Flowchart Process are only listed due to provide the completeness of the end result. Both attributes were added later in development in order to tackle a detected weakness regarding the execution, elaborated further in section 6.4.

| Element | Attribute | Elaboration | Form |
|---|---|---|---|
| **Movement Task** | | |  |
| | **Movement-Type**<br>• Jump Gap<br>• Follow Line<br>• Turn Left<br>• Turn Right<br>• LED On<br>• LED Off<br>• Custom<br>• Flowchart | **Appearance**<br>• Dropdown<br>**Condition**<br>• none |  |
| | **LED-Control**<br>• Red<br>• Yellow<br>• Blue<br>• Green | **Appearance**<br>• Dropdown<br>**Condition**<br>• Only active when "LED On" is selected | |
| | **Flowchart Process** | **Appearance**<br>• Text field with a add-icon and a delete-icon<br>**Condition**<br>• Only active when "Flowchart" is selected<br>**Description**<br>• Interreference that allows calling an existing Flow Chart Model. | |
| | **Custom Movement Code** | **Appearance**<br>• Large text field<br>**Condition**<br>• Only active when "Custom" is selected<br>**Description**<br>• Text field to manually enter AdoScript code | |
| | **External AdoScript** | **Appearance**<br>• Text field with file system with a lens icon<br>**Condition**<br>• Only active if "Custom" is selected<br>**Description**<br>• Search for AdoScripts with the file browser | |

Figure 33 Conceptualization CarMovement element

### *6.3.2.2 Chosen Metamodelling Tool*

ADOxx was chosen as the platform for development, not least because OMILAB had its part in serving as inspiration during the thesis start. Another possible platform would be MetaCase, with their best-selling product MetaEdit+.

ADOxx is a meta-modelling development and configuration platform for implementing modelling methods. Implementation of full-fletched modelling methods can be realised using the platform, consisting not only of a modelling language but also of modelling procedure and the corresponding functionality in the form of mechanisms and algorithms (further elaborated in chapter 5.1).

## 6.4 Execute the model and move the mBot

Conform with the section title REQ-201, REQ-202, and REQ-203 are tackled with the previously planned constructs for REQ-030 and REQ-031 in consideration. For the movement of the Cyber-Physical system, the following physical material is derived:

- mBot or another mini vehicle with a control interface
- Access to the above-mentioned control interface and right to execute movement commands (REQ-201).
- LED or text board for vehicle type identification (train, bus, pedestrian, bicycle (REQ-203)).
- Optional: road map or other driving environments for the mini car to make the scenario more vivid. This would contribute to a more sophisticated fulfilment of REQ-202.

The mentioned material requirements might indicate to the mBot used within the OMILAB, but from the description given above, alternatives could be derived. In the example of this thesis, the mBot with adjustable LED colours and line sensors is used. It is described in more details in chapter 5.2.3. In parallel to the physical material, software is needed for development and modelling.

- Tool for modelling.
- A workflow engine to run the model.

In alignment with the scenario, the following factors can be presumed to achieve the first success:

- A scenario executed from start to finish.
- Move the mBot.
- Make decisions using if-else gateways.

First, the two input variables, what is the reason for the trip (leisure or work) and the kind of luggage must be defined (REQ-020, REQ-021). Based on the inputs, the mBot, with an indication of the transport mode, specifically the LED, will follow a different route.

To execute the model, all components must be connected to the same network or be available publicly. As far as the street layout design goes, it can be printed out and put to the ground, or the mBot can directly drive on the ground, and it can manually be checked whether the car takes different routes but ends up at the same destination.

Currently, the Bee-Up tool and the ADOxx modelling toolkit, are not supported with a workflow engine to execute BPMN or an extended, respectively newly created DSML. To provide a feedback loop with the CPS, two workarounds are proposed.

### 6.4.1 Bee-Up with Flowcharts and AdoScript execution

While becoming familiar with the OMiLAB infrastructure and achieve the above-mentioned first success, the Bee-Up tool is used to model flowchart elements that control the mBot by Makeblock. To execute the defined scenario with the Bee-Up tool, firstly the variables purpose (REQ-020) and luggage (REQ-021) have to be specified, the mode of transportation be decided (REQ-022) and reflected by the CPS (REQ-203). Then movement needs to be chosen (REQ-030) and performed in the physical space (REQ-201) along the layed out street map (REQ-202). For the movement itself, ADOScripts are used to send HTTP requests to the mBot API. To support the execution within the modelling tool instead of Bee-Up, an HTTP-extension DDL-File must be loaded within the development toolkit.

To do that the device that, contains the model has to be connected to the OMiLAB network in Olten since the IP is not publicly addressable. The network used during the development is not connected to the internet. In contrast to the laboratory in Olten, in the OMiLAB in Vienna, some spectator cameras and vehicle cameras are addressable via the Internet.

An alternative to Flowchart would be Petri nets, one reason being that Bee-Up already has a built-in engine for Petri nets. Furthermore, it was shown on the introduction day of OMiLAB in Olten, as well as in the OMiLAB description, that through Petri-Nets, the operation of the robotic arm Dobot Magician is possible. When comparing these two languages in perspective of the mobility planner Petri Nets requires a lot more cognitive effort to learn than flowchart.

The approach is to adapt an existing scenario that already uses ADOScripts for decisions and other elements. The individual code fragments and internet addresses can be derived from the API description. A possible disadvantage of this approach is the appearance of the model, which becomes very unfriendly for less technical-affine modellers.

### 6.4.2 Camunda with a java-based workflow engine

As a further opportunity, Camunda is used to display a possible realisation of the concepts described in the previous section. In contrast to ADOxx, Camunda does not offer the possibility to customise the metamodel or to create one itself, but the basic focus is strongly on process automation (Camunda, 2021a). This is especially noticeable in the configuration of service tasks in the Camunda Modeler, its own modelling tool. From this tool, models can be deployed and executed directly on a hosted Camunda platform. As an example, all manual tasks can be marked as done via the platform by user logins. Furthermore, service classes can call Java classes and execute the code.

As described in Camunda (2021b) and displayed in Figure 34, the extension provides an HTTP-connector to the Service Tasks and the attribute connectorID.

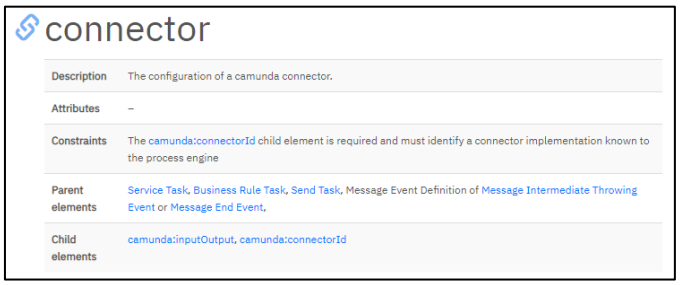



Figure 34 Excerpt of Camunda (2021b) displaying the added HTTP-Connector

The Camunda BPM engine is an open-source, Java-based framework that provides an intelligent workflow or business process management (BPM) system for any kind and size of the organisation. It offers pre-designed BPM systems that can be modelled and executed for workflow and business process automation, Case Management, and Business Decision Management (Camunda, 2021a).

Camunda developers are continuously striving to make design and implementation easier for Java developers who work with workflow processes. To enable non-Java developers, use the BPM system and the process engine technology, Camunda BPM offers a REST API that allows one to build applications with a remote process engine connection (REQ-205). The core consists of a lightweight execution engine that uses less than 3MB of disk space. The engine can run in any Java Virtual Machine (JVM) and comes with extended integration for different runtime containers (Lal, 2017)

## 6.5 Summary

This chapter presented firstly, that BPMN has been chosen as the language to extend and the mobility planning scenario was further adapted. Section 2 then explains the idea to model BPMN with four possible tools in order to further evaluate which requirement can be fulfilled with BPMN as it is implemented in these tools and what exactly needs to be modified.. In a second step, three concepts for a DSML are presented:

- adding attributes to the existing task elements;
- a language from scratch with its own elements;

- or a hybrid solution where a new element is created but integrated into the existing structure of BPMN later called BPMN4MoPla.

The last part of the chapter then deals with the execution in a CPS environment and the transformation of the model to be executed in an ADOxx or Camunda environment.

# 7 Proof of concept - BPMN4MoPla

This phase of the DSR approach is concerned with the development of the artefact. According to the concept and plans elaborated in chapter 5, the proof of concept creation commences by using different modelling tools to cross-examine the coverage of BPMN with the elicited requirements. Within the ADOxx Development environment the Bee-Up library is extended and the process of creating a new language (leaned on BPMN) with a new CarMovement element is designed. Then BPMN4MoPla is created with the integration of the CarMovement element into the Bee-Up library, embedded as a Sub-class of a task. In order to generate an impact in the physical world two approaches of transforming the model to a platform with the support of a workflow engine to move the mBot through the defined street layout and thus generate a feedback loop.

## 7.1 BPMN tools, models, and nuances

In this part of the proof of concept, a possible solution is shown how the trip to the airport scenario can be modelled in BPMN. To model this trip to the airport scenario, four different tools were considered and tested.

- Visual Paradigm
- Bee-Up, described in chapter 5.2
- ADOxx Modelling Toolkit (with the Bee-Up library)
- Camunda Modeler

Although each tool serves the purpose of modelling BPMN, all vendors have different interpretations of how in particular, the task element is described in its numerous facets.

Figure 35 presents the trip scenario modelled with the ADOxx Modelling Toolkit. For the sake of clarity, the car movements are shown as sub-processes.

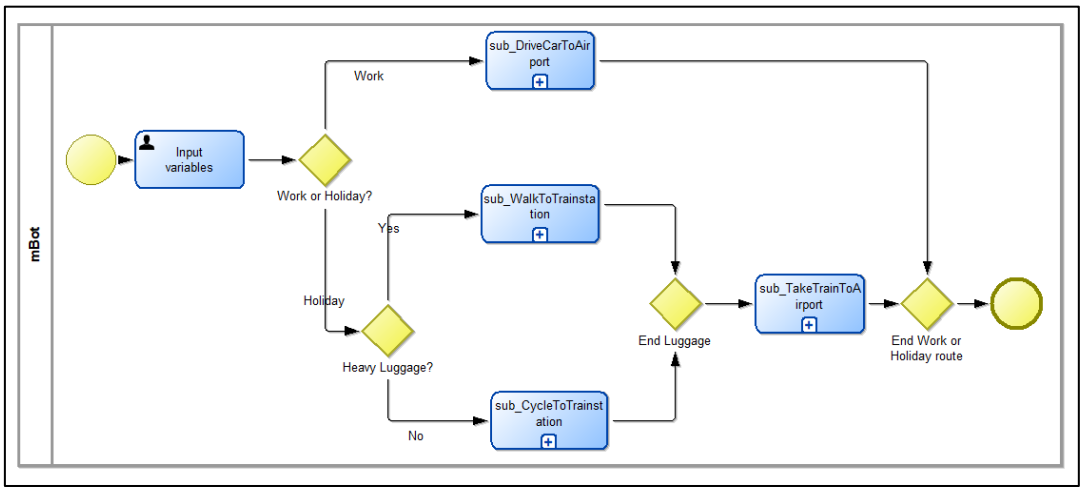



Figure 35 Trip modelled in BPMN.

Additionally, Figure 36 depicts an example of the movements performed to drive the car to the airport.

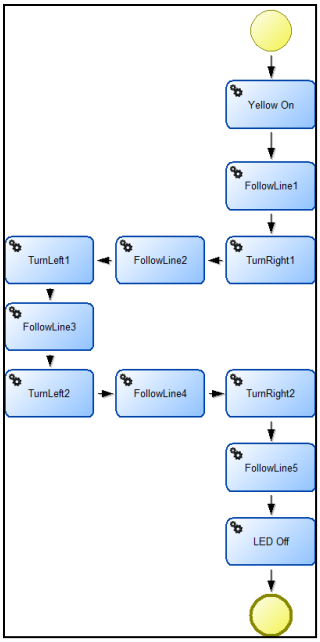



Figure 36 sub_DriveCarToAirport

The modelled process appears identical when modelling the same process with the Bee-Up tool. But Figure 37 displays dissimilarities between the two ADOxx tools when defining individual tasks. Within the Bee-Up tool the Task type is specified in a distinct tab and is displayed as a radio button in comparison to the dropdown of the Modelling Toolkit. Although the presentation is different the meta model indicates that in both implementations the Task type is defined as an Enumeration.

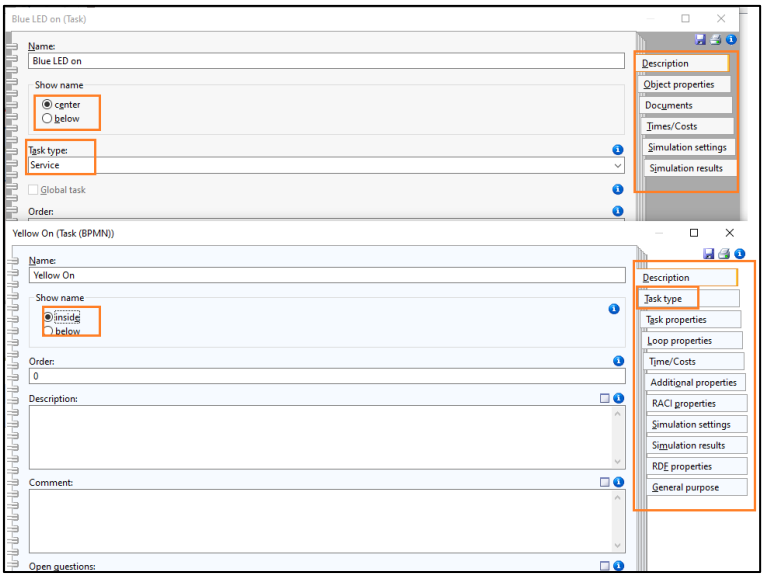



Figure 37 Differences ADOxx BPMN library (top) and Bee-Up (bottom)

Whilst investigating with a limited effort, the implementation of Visual Paradigm revealed a different style of representation. Figure 38 provides a visualisation of a task within Visual Paradigm (academic version provided by the University of Camerino).

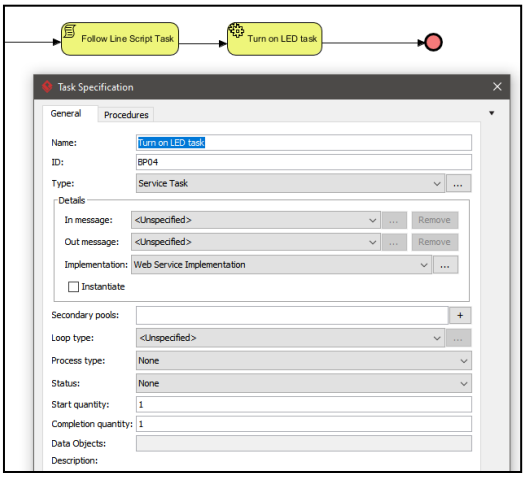



Figure 38 Example of a Service Task within Visual Paradigm

Additionally, no related workflow engine could be elaborated; therefore, the CPS environment requirements have a very low chance to be fulfilled and Visual Paradigm has no longer been considered as a tool suite to create or adapt a DSML.

The fourth tool that has been used is the Camunda Modeler, which can then be used to deploy syntactic correct models to the Camunda platform.

An implementation of the Trip planning scenario in the Camunda Modeler is presented in Figure 39. Identical to the ADOxx tools, Camunda provides the possibility to either model the routes via sub-processes or all individual movement tasks within one model, Figure 39 displays both approaches combined in one model.

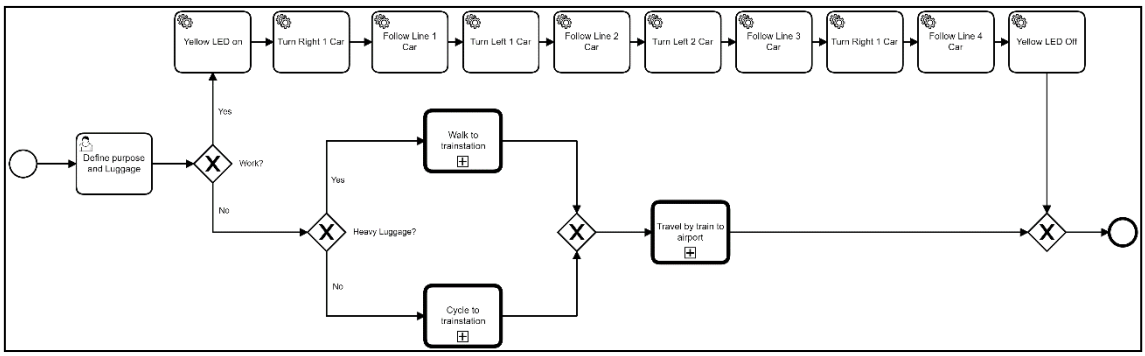



Figure 39 Camunda Modeler Trip planning overview

The configuration of the service tasks is described in chapter 7.4.2. This decision was made because the aspect of the Camunda workflow engine that is available free of charge and provides an already existing BPMN extended DSML is elaborated separately.

This section showed that from a modelling perspective, the integration of the CPS movements could be displayed in BPMN without any extension with different existing tools. Using these different tools also revealed the effort needed to switch from modelling to executing models in a business environment. This presented room for improvements regarding convenience and reducing the technical knowledge and skills that a Modeller requires to model this specific trip scenario.

## 7.2 Creating a DSML

This section describes the extensions of BPMN as well as the creation of a new modelling language. For the former purpose, several fields were added and described accordingly how their execution with a workflow engine could look like. The latter describes, based on the proposed elements from the previous chapter (6.3.2.1), a new modelling language, which includes a new element as well as basic elements mostly adapted from BPMN.

### 7.2.1 BPMN task extension

In order to extend BPMN in a first step, the tool or the underlying metamodel of Bee-Up was considered. The main reason for this decision was the public availability of this metamodel library.

Figure 40 displays a possible extension of the existing task element, where various attributes have been added and described below:

- Number 1 marks the separate tab on the right side. This separates CPS commands from previous task properties. In a further step, this tab could also be made visible only when the Modeler selects, for example, Script Task or Service Task, or "CPS Task" could be added to the existing task types of BPMN. Important to remind that if creating a new Task type it is a value in an Enumeration and not the implementation of a new sub-Task.
- Number 2 marks the idea of a dropdown where you can choose in which direction or which movement is performed (LED colours included). This leads to the consequence, that the individual scripts or API requests must be developed and embedded in an underlying execution engine.
- Number 3 displays the idea to implement custom ADOScripts. This variant can be observed within flowchart elements of the Bee-Up tool (chapters 7.4.1 and OMiLAB Bee-Up tool5.2). In comparison to the above-mentioned number 2 the

code has not to be embedded in a workflow engine, but it requires an engine that is able to execute ADOScript code.

- Number 4 would require the same engine described in number 3; Although this would have the vehement advantage that Modelers do not have to write scripts but can import them from an external source (or even the modelling library). This would also bring the benefit of a high reusability.

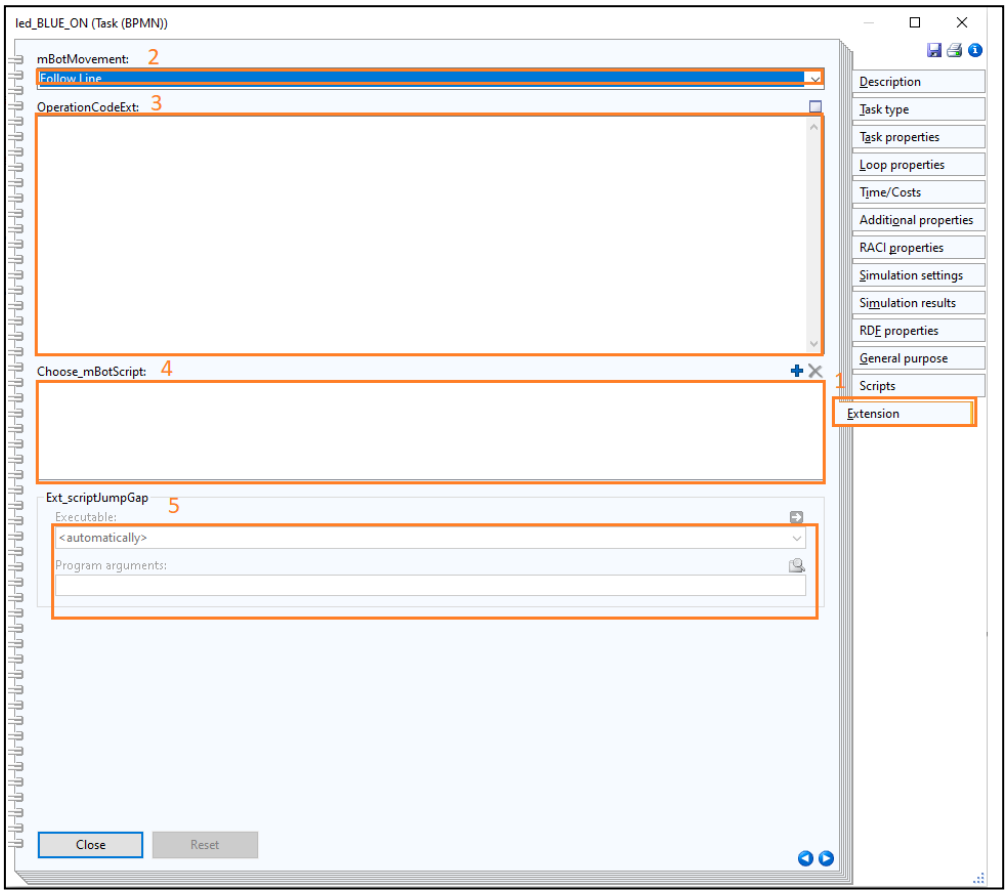



Figure 40 Extensions of the BPMN task of the Bee-Up library

After the attempts to extend the task, the idea quickly arose to separate the CPS actions with a completely new modelling element instead of trying to add all these commands and attributes into an existing and already attribute-clotted task. It has been decided to start from scratch and create a modelling language.

In comparison to the concept, it has been realised that the possibility to specify the Luggage and the Purpose attribute need to be provided. In order to do that the Task properties tab has been extended with the two attributes required, but they are only modifiable of is the User task is chosen. This change is displayed in the bottom of Figure 41.

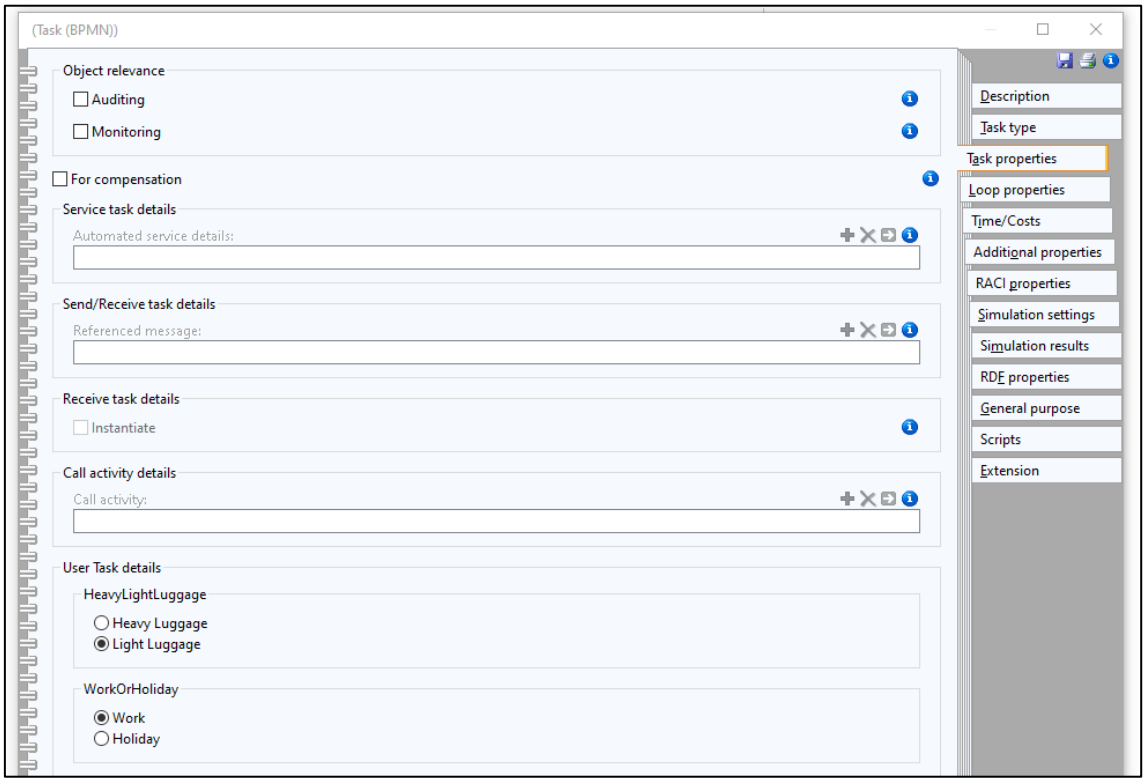



Figure 41 Added Luggage and Purpose Attribute to the Task element

## 7.2.2 Designing a basic language

The current step is to develop an element, that takes control of the movements of the CPS. This is to be done with a few basic elements such as the variable declaration, decision gateway, a start and end point and a sequence flow to determine the order of execution. The basic idea regarding the appearance of the elements is found in chapter 6.3.2.

The result of this section is a modelling tool in which one can declare variables, make one or more IF, Else decisions and control the robotic car, either with predefined commands like "Follow Line", "Jump Gap", "Turn Right" to name three examples or with custom

operation code adopted from flowcharts. This is implemented according to the design ideas from the previous chapter. The CarMovement element created here then served as the basis for the next section (7.3). Figure 42 shows an overview of the new modelling tool. On the left side, the file explorer and a small orientation guide can be spotted. In the middle is the palette, displaying the accessible elements and the white area is the modelling meadow.

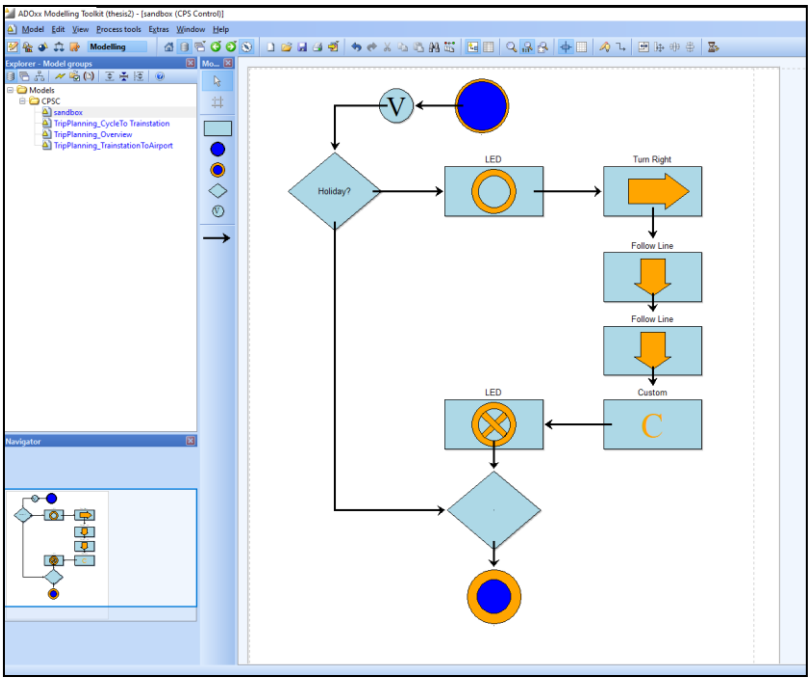



Figure 42 Overview implemented from scratch DSML.

When examining the main element, the CarMovement task, as soon as you select LED, the LED control field mentioned above becomes active and can be changed to five different values, including four colours and LED switch-off. The same principle applies to the Custom type, which enables the external scripts and the custom code. While Figure 43 provides an overview of the Movement Type dropdown Figure 44 and Figure 45 show the element in its entirety.


Figure 43 CarMovement element dropdown Movement Type


Figure 44 CPS Movement element description chapter

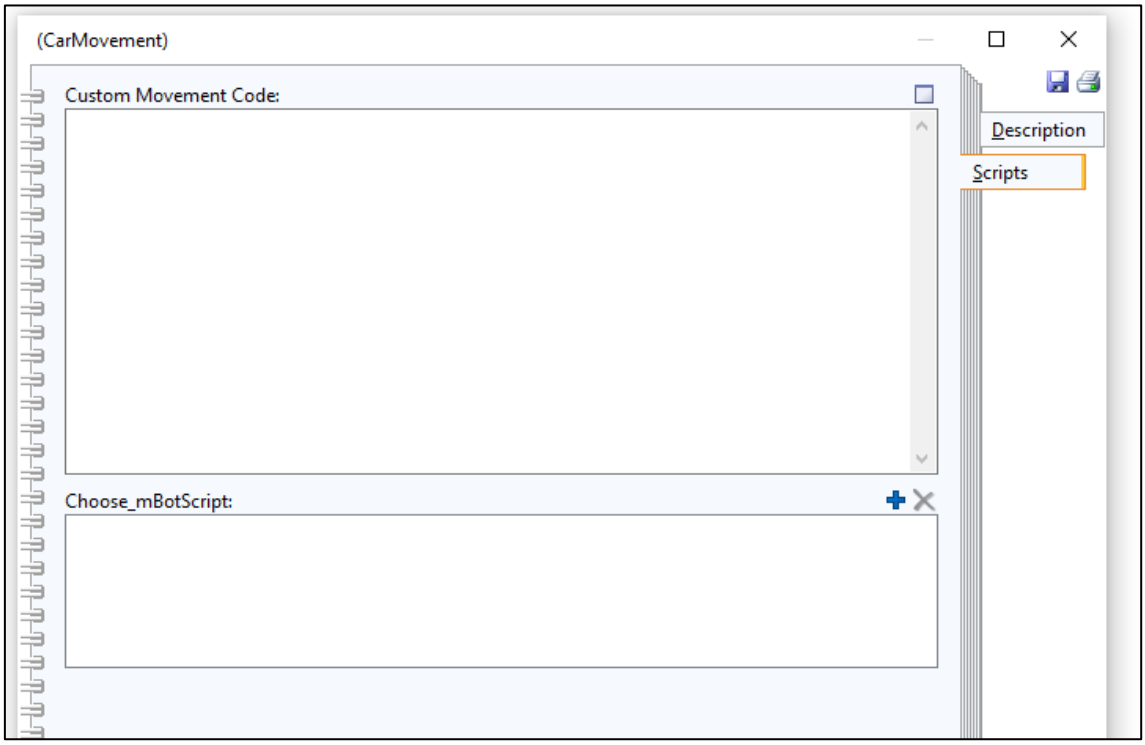



Figure 45 CPS Movement element scripts chapter

### *7.2.2.1 Implementation in ADOxx Development Kit*

Following the example suggested in chapter 6.3.2.1 the newly created library consists of the elements displayed in Figure 46.

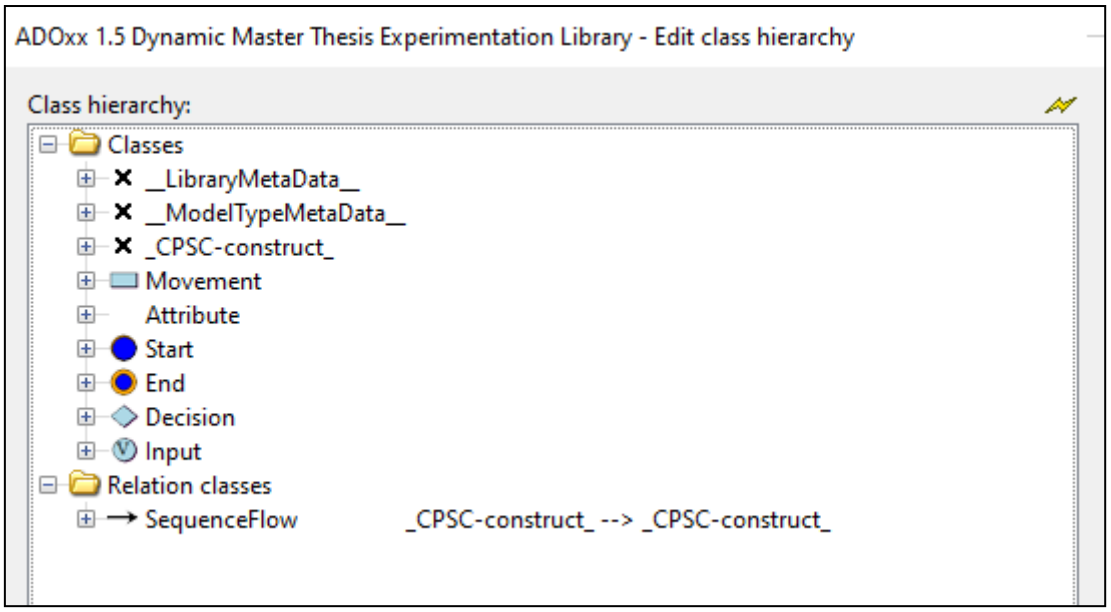



Figure 46 Overview from scratch DSML library

The language now consists, as planned, of a relation class, above called “SequenceFlow”, which has the task of showing which processes take place in which order. Furthermore, in an implementation of a process workflow engine, the function of entering conditions when this path is selected would be added, but this is a preconceived idea based on the experience with the workflow engine of Camunda in past modules at the FHNW and UNICAM.

The start and end nodes currently serve as graphical representatives, there is no function attached to that. In case of an engine the start node would be addressed with a start button or an external request, be it from a website sending a request or a link to process with a sub-process. Similarly, the end node needs to provide feedback in any form that the process has been completed.

For the input of the variables, an element was created that explicitly asks for value assignments of the variables Luggage and Purpose.

## 7.3 Creation of BPMN4MoPla

It is evident from the thesis statement that the focus of the language is to be applicable in a business world, i.e., not just controlling a CPS with a specific software but as a tangible consequence of a process or a business decision. Therefore, it is an immense advantage if the established and widespread BPMN structure, which allows and supports the integration of DMN in addition to the modelling of business processes and can be extended. At first glance, when inspecting Figure 47 there is almost no difference recognisable in comparison to Figure 35, which is not surprising but even aspired.

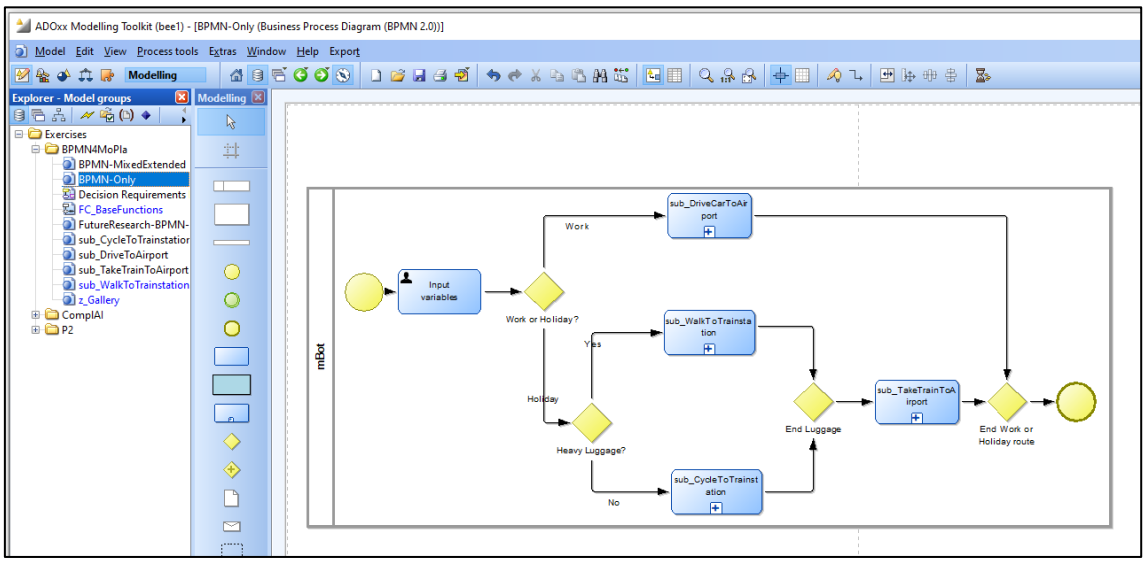



Figure 47 BPMN overview with as sub-processes

While taking a closer look to sub_DriveCarToAirport the difference is visually recognisable in Figure 48.

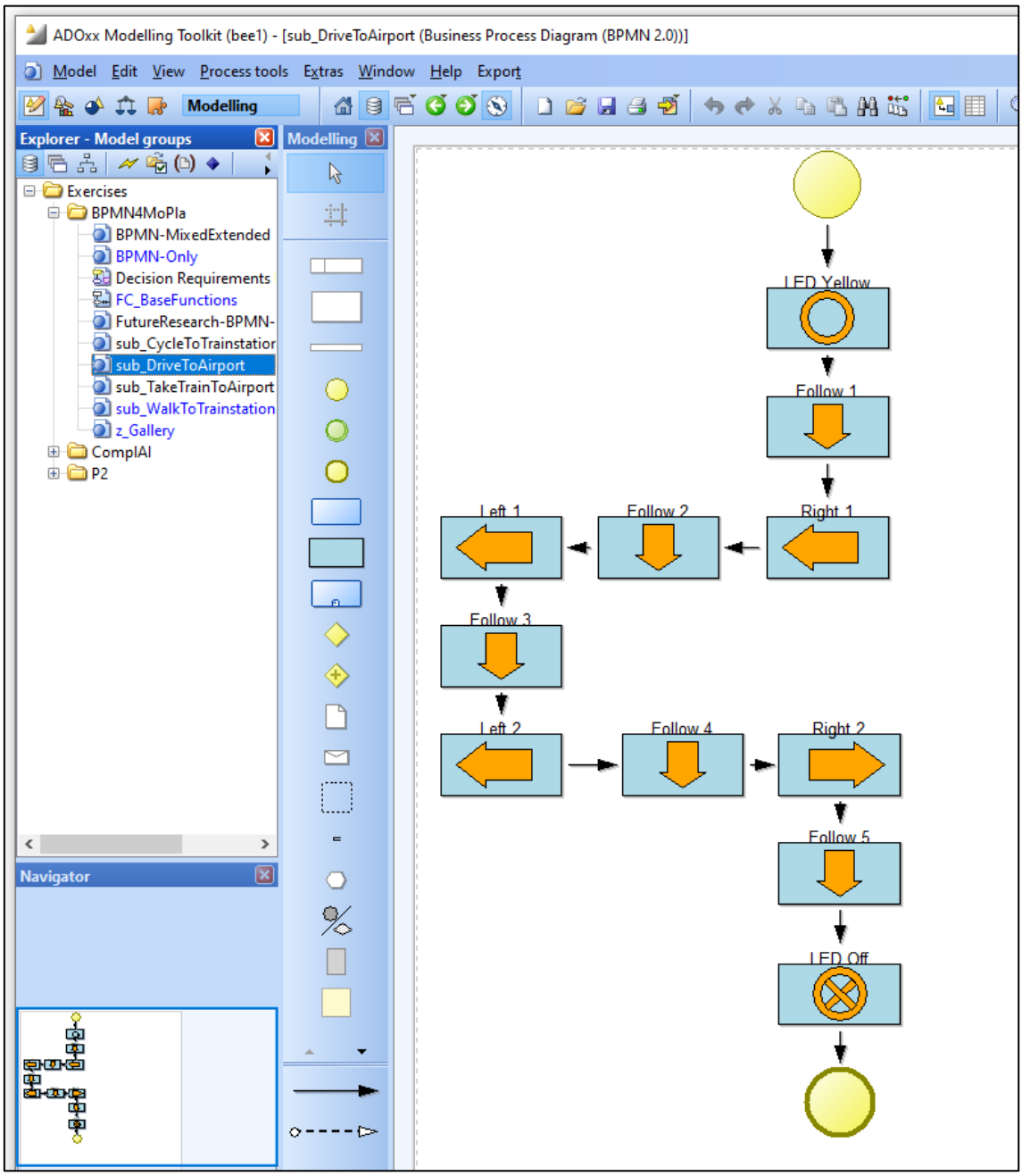



Figure 48 Drive to the airport by car sub-process

Between the file explorer on the left and the model, the palette with BPMN elements can be seen, including the newly embedded CarMovement element. Of course, it is also possible to apply existing and new elements side by side. Figure 49 depict how this can be displayed.

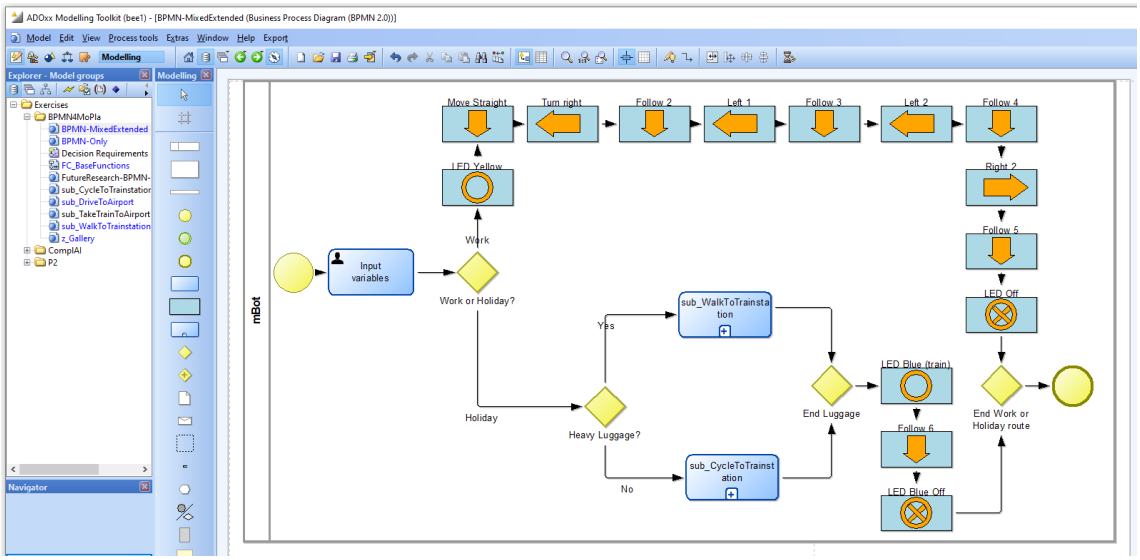



Figure 49 BPMN and CarMovement elements combined in one model

## 7.3.1 Library adaptation

The Bee-Up library, which is available on the OMiLAB site[7], was used as a starting point. A CarMovement element was then integrated as a sub class of a Task.

Figure 50 shows a section of the bee-Up library with a focus on the CarMovement element that has newly been embedded.

[7] https://austria.omilab.org/psm/content/bee-up/download

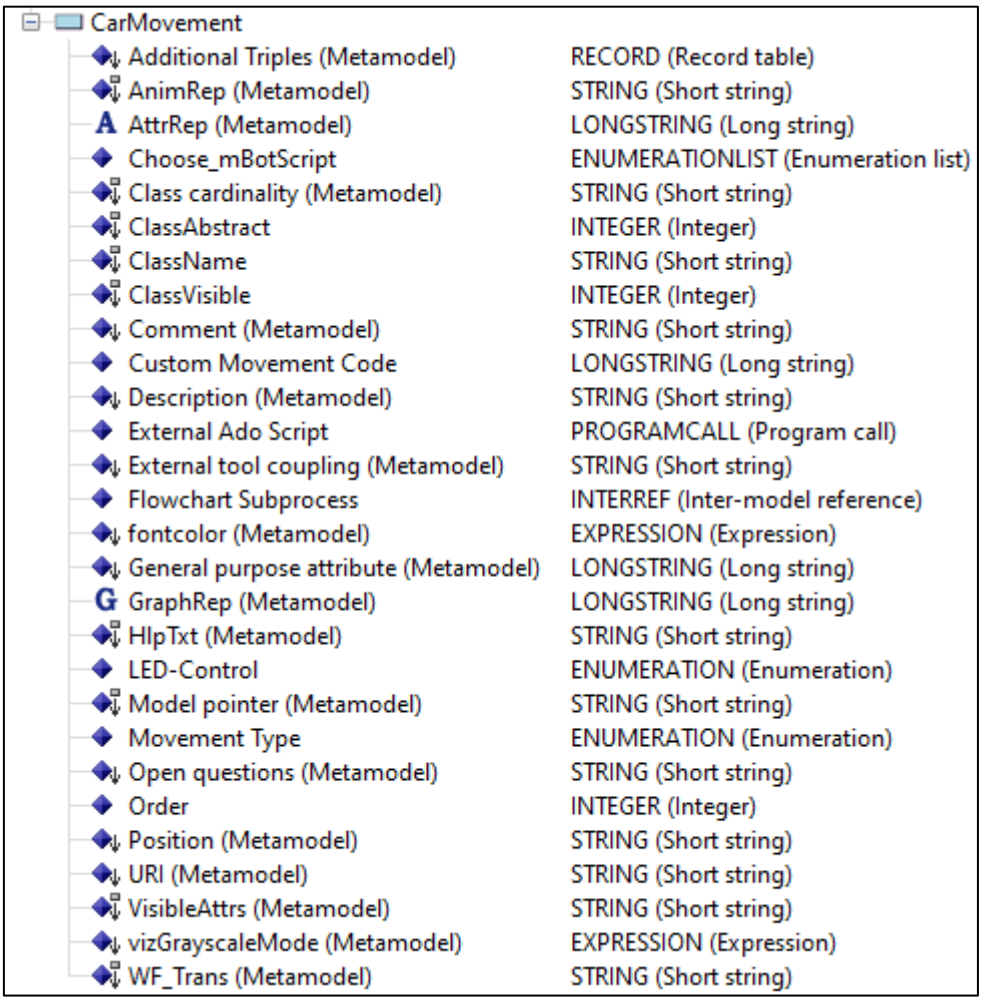



Figure 50 Extract of the adapted Bee-Up library

Very inconspicuous, but on closer inspection, the attribute "Flowchart Subprocess", an inter-model reference mentioned can be recognised. This allows a reference to a flowchart process, as shown in Figure 51.

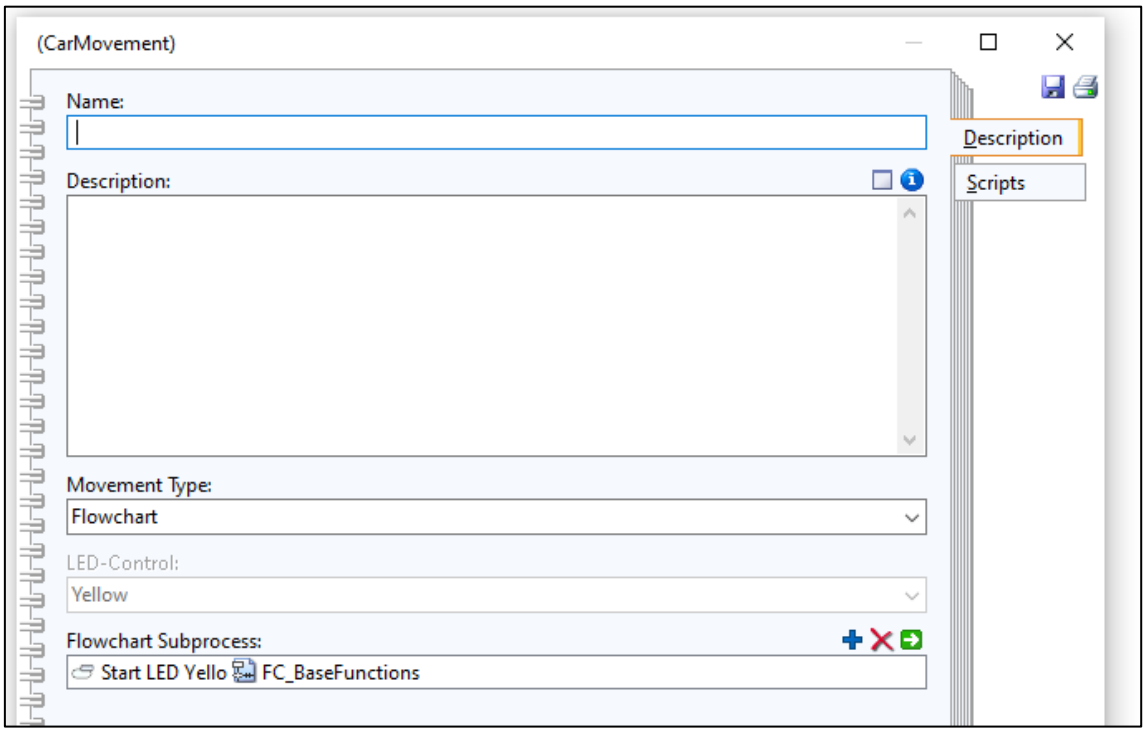



Figure 51 CarMovement element with selected flowchart item

To display the window as shown in Figure 51, the class attribute "AttrRep", which controls the availability and structure of the ADOxx-notebook, is configured as shown below. The class attribute "AttrRep" is of type long string. Hence the text entered as value is interpreted equally to a configuration script of the so-called notebook. Each notebook has chapters, which contains a list of attributes that may be grouped and relations that are allowed for this class can be automatically created as an own chapter. The appearance of attributes is defined by lines, dialogue, control types (ctrltype), width or format. The command AVAL is used to declare variables which, as in every programming and scripting language, serve as basic elements and can be compared with each other. A complete excerpt of the AttRep class can be found on the ADOxx page[8].

[8] https://www.adoxx.org/live/attrrep

```
1.  NOTEBOOK
2.  CHAPTER "Description"
3.  ATTR "Name"
4.  ATTR "Description"      lines:10
5.  AVAL mType:"Movement Type"
6.  ATTR "Movement Type"
7.  ATTR "LED Control"     enabled:(mType= "LED")
8.  ATTR "Flowchart Subprocess"   enabled:(mType= "Flowchart"   )
9.  CHAPTER "Scripts"
10. ATTR "Custom Movement Code"  lines:15     enabled:(mType= "Custom" )
11. ATTR "External Ado Script"     enabled:(mType= "Custom" )
```

Similar to the AttrRep the class attribute GraphRep is of type long string, hence the attribute value is a text that is interpreted as a script by the GraphRep interpreter. Figure 52 shows the 8 variations of appearance the CarMovement element can take on.

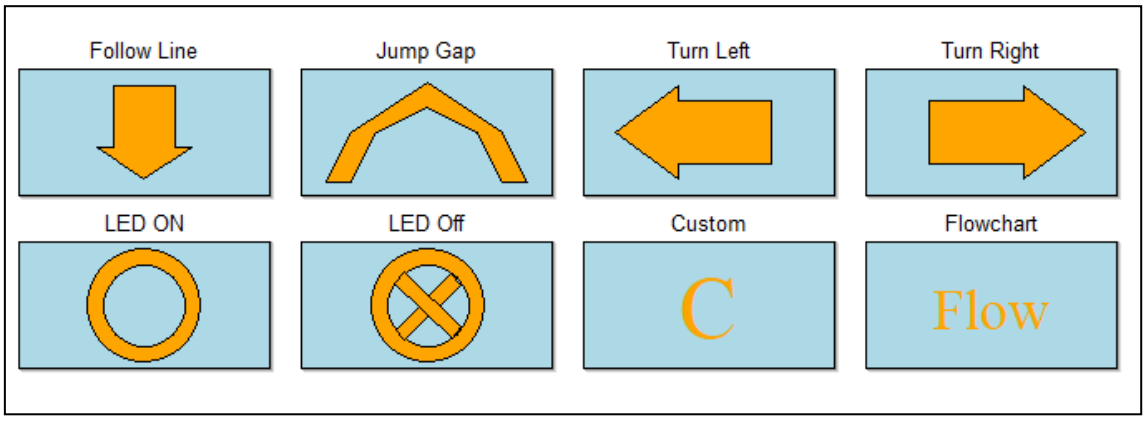



Figure 52 Different symbols for the CarMovement element

To achieve that, the following code block has been developed and stored in the GraphRep attribute. The complete excerpt of the GraphRep class can also be found on the ADOxx page[9].

[9] https://www.adoxx.org/live/graphrep

```
GRAPHREPlayer: - 1 sizing:asymmetrical

# select type for color
AVAL mType:"Movement Type"
AVAL ledControl: "LED- Control"

SET f: "lightblue"
FILL color:(f)
# drawing main box
RECTANGLE x: - 2cm y: - 1cm w:4cm h:2cm
ATTR "Name" x:0cm y: - 1.5cm w:c h:t

IF(mType = "Turn Right" )
FILL color:orange
POLYGON 7
x1:1.5cm y1:0cm x2:0.5cm y2: - 0.75cm x3:0.5cm y3: - 0.5cm x4: - 1cm y4: - 0.5cm x5: -
1cm y5:0.5cm x6:0.5cm y6:0.5cm x7:0.5cm y7:0.75cm
ELSIF(mType = "Turn Left" )
FILL color:orange
POLYGON 7
x1: - 1.5cm y1:0cm x2: - 0.5cm y2:0.75cm x3: -
0.5cm y3:0.5cm x4:1cm y4:0.5cm x5:1cm y5: - 0.5cm x6: - 0.5cm y6: - 0.5cm x7: -
0.5cm y7: - 0.75cm
ELSIF(mType ="Jump Gap")
FILL color:orange
POLYGON 10
x1: - 1.6cm y1:0.8cm x2: - 1.2cm y2:0cm x3:0cm y3: -
0.8cm x4:1.2cm y4:0cm x5:1.6cm y5:0.8cm x6:1.2cm y6:0.8cm x7:0.8cm y7:0cm x8:0
cm y8: - 0.4cm x9: - 0.8cm y9:0cm x10: - 1.2cm y10:0.8cm
ELSIF(mType ="Follow Line" )
FILL color:orange
POLYGON 7
x1: - 0.5cm y1: - 0.75cm x2:0.5cm y2: -
0.75cm x3:0.5cm y3:0.25cm x4:0.75cm y4: 0.25c m x5:0cm y5:0.75cm x6: -
0.75cm y6:0.25cm x7: - 0.5cm y7:0.25cm

ELSIF(mType= "LED")
FILL color:orange
ELLIPSE x:0.00cm y:0cm rx:0.9cm ry:0.9cm
FILL color:lightblue
ELLIPSE x:0.00cm y:0cm rx:0.65cm ry:0.65cm
IF(ledControl= "Off" )
FILL color:orange
POLYGON 4
x1: - 0.5cm y1:0.39cm x2: - 0.37cm y2:0.55cm x3:0.55cm y3: - 0.31cm x4:0.39cm y4: -
0.5cm
FILL color:orange
POLYGON 4
x1: - 0.52cm y1: - 0.35cm x2: - 0.35cm y2: -
0.52cm x3:0.54cm y3:0.35cm x4:0.40cm y4:0.52cm
ENDIF
ELSIF(mType="Custom" )
FONT "Times" h:40pt color:orange style:bold
TEXT "C" y:0.13cm w:c h:c
ELSIF(mType="Flowchart" )
FONT "Times" h:26pt color:orange style:bold
TEXT "Flow" y:0.13cm w:c h:c
ENDIF
```

## 7.4 Execution with a workflow engine

The first task is the definition of basic car movement tasks that later can be used as building blocks to create routes. In a second task, the defined car movement tasks need to communicate with the REST interface to physically move the mBot. In a first attempt, no line following function was used during the route creation. This is because road maps were not yet available at that time. Therefore, these line following functions have been adapted an iteration later.

### 7.4.1 The ADOScript flowchart execution engine

#### *7.4.1.1 Technical abstractions*

For the abstraction, a flowchart model was chosen. This decision is based on several principles:

Firstly, the sequential and linear flow is perfectly suited to represent vehicle movements such as 2 seconds at a speed of 100 followed by a turn left for 1 second at the speed of 120. From the API could not be derived what the speed of 100 or 120 exactly meant. By trial and error, it has been found out that with values under 60, the motor does not provide enough power to move the car at all. In addition, the time is measured in seconds, and the input for both is an integer, which prevents the usage of decimal numbers.

Secondly, the software Bee-Up, a used modelling tool in chapter 7.1, developed by OMiLAB advertises the use of this CPS. It was presented at the OMiLAB opening on September 1st in Olten. The "IMKER" study (Karagiannis et al., 2017) provides an overview of the fundamental structure of the software and the provided functions.

Figure 53 visualizes the basic functions of the mBot car. At this moment, the functions are moving forward by either just drive or following a line, turn left and right with or without a line, as well as changing the LED colours to represent the transportation mode according to the defined scenario that has been introduced at the beginning of this chapter.

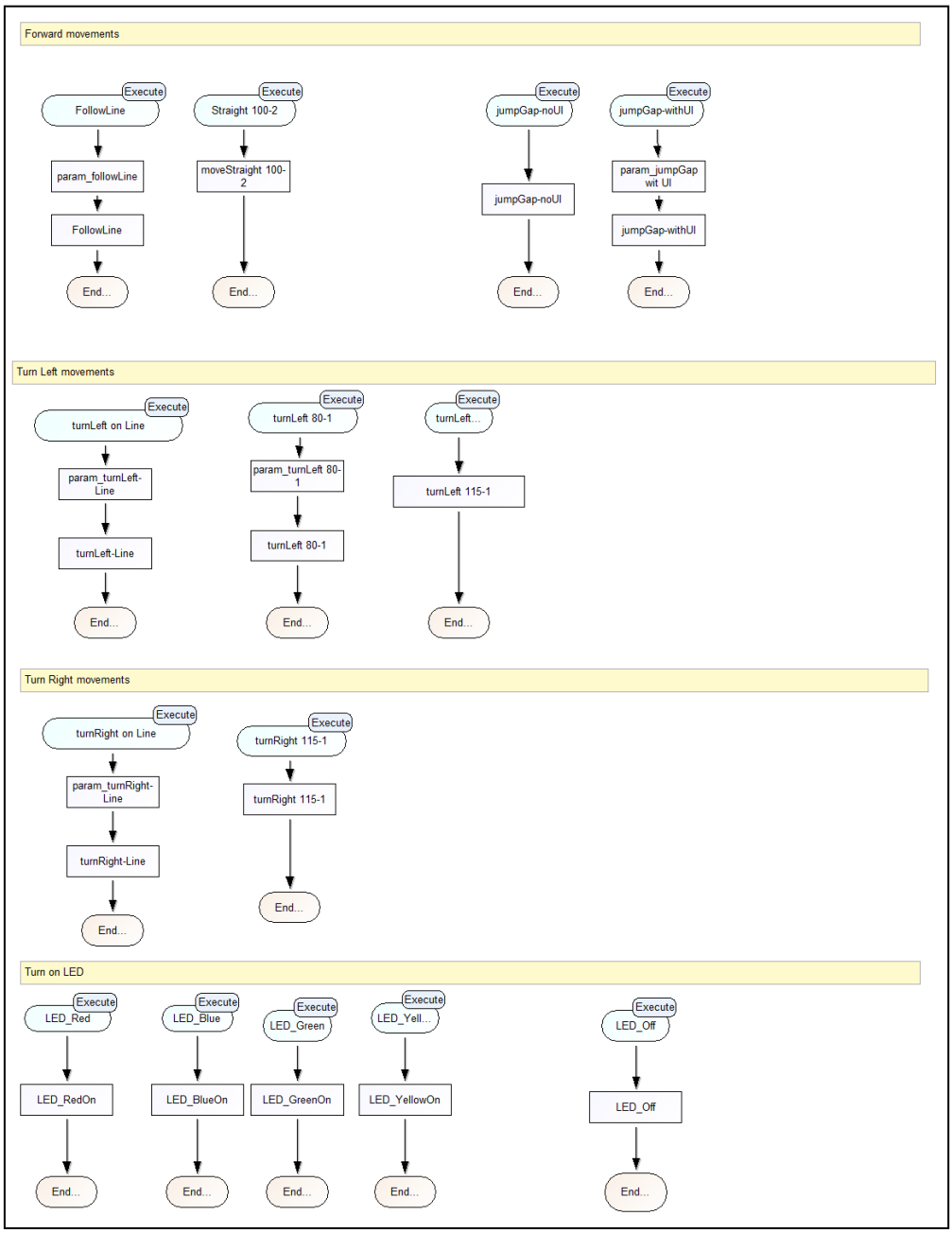



Figure 53 Fundamental car movements, designed with the Bee-Up tool.

In a second step, four different routes have been created, one for each transportation possibility. This step has mostly been conducted to separate all the movements from the main model where decisions are made, which would lead to cleaner models, especially if more routes or alternate routes are implemented in the future. Figure 54 represents the routes compounded by the mBot movements.

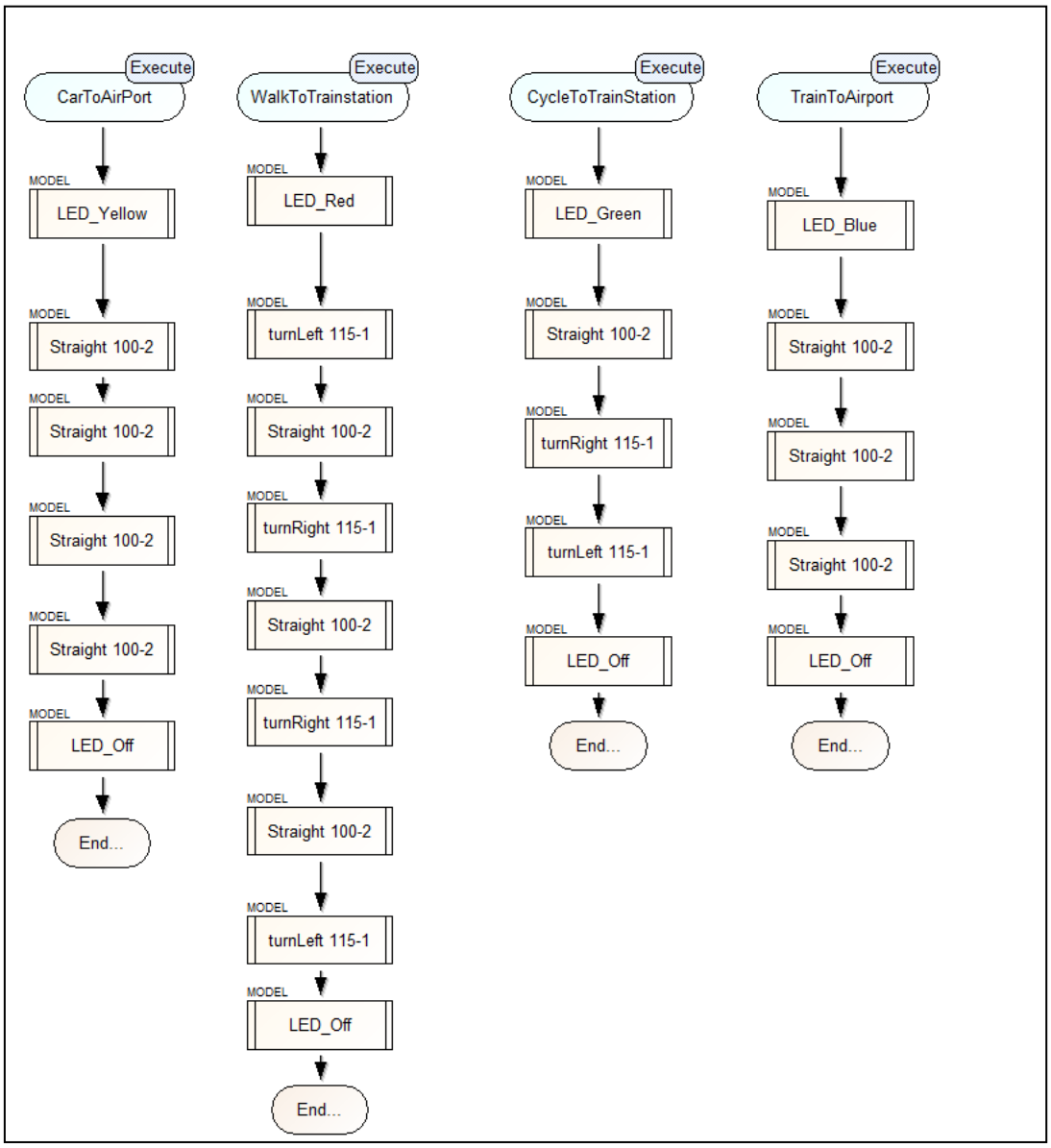



Figure 54 Routes consolidated of vehicle movements.

### *7.4.1.2 Administer the car via REST*

To physical move the mBot, the provided REST interface is called with an AdoScript. An overview of all the REST components can be found in the Appendix. Example code snippets used in the first iteration are presented in Table 7.

| Function | AdoScript |
|---|---|
| Switch LED Colour | SETL map_headers:({"Content-Type":"application/json"})<br>HTTP_SEND_REQUEST<br>**("http://10.0.6.59:8080/mBot/api/internalled/turnonled?red=100")**<br>str_method:("GET")<br>map_reqheaders: (map_headers)<br>str_reqbody:("do")<br>val_respcode:val_httpcode<br>map_respheaders:map_respheaders<br>str_respbody:str_respbody |
| Car straight | SETL map_headers: ({"Content-Type":"application/json"})<br>HTTP_SEND_REQUEST<br>("http://10.0.6.59:8080/mBot/api/movement/moveForward?speed=100&secs=2")<br>str_method:("GET")<br>map_reqheaders: (map_headers<br>str_reqbody:("do")<br>val_respcode:val_httpcode<br>map_respheaders:map_respheaders<br>str_respbody:str_respbody |

Table 7 JSON code snippets to call REST functions

#### *7.4.1.3 Model design to define the trip*

The model, which combines the user input with the technical abstractions, is showed in Figure 55. Additionally, Figure 56 displays the User inputs in order to run through the whole process. In its current state, the user is asked if the trip is regarding work or holiday. For work, the model is taking the car. If the value holiday is chosen, the user must distinguish if only a backpack is taken to the trip or is there some luggage as well. When there is luggage, the model is walking to the train station instead of taking the bicycle.

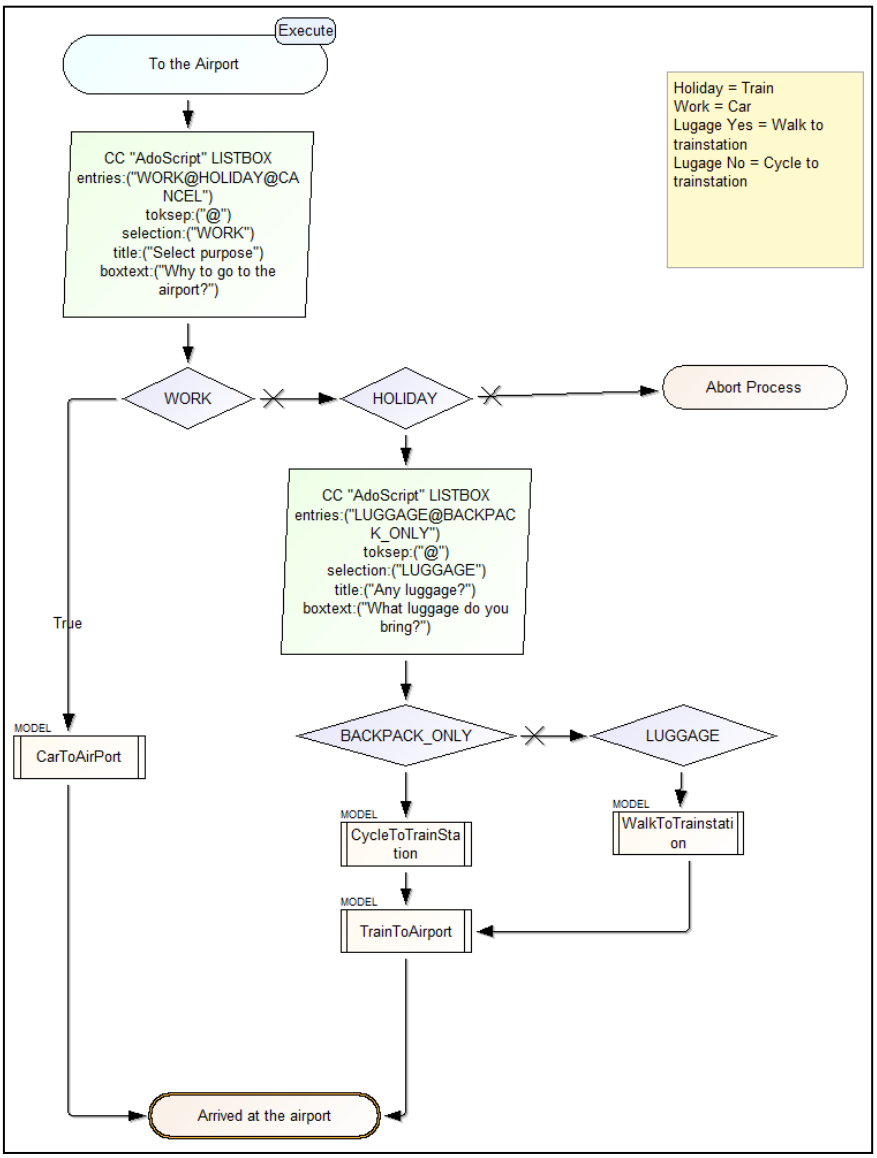



Figure 55 First iteration main model

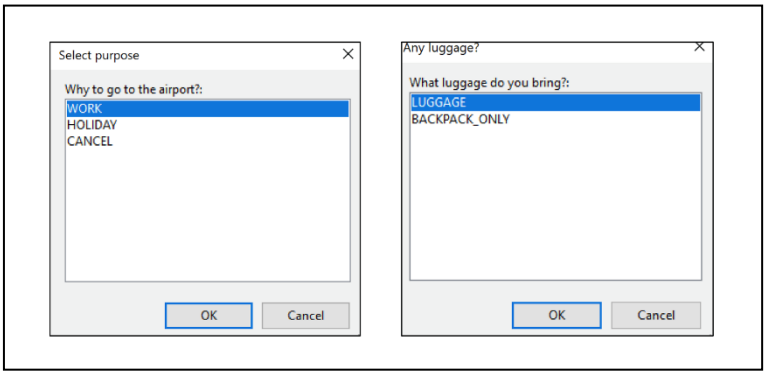



Figure 56 User input example

#### *7.4.1.4 Showcase of the first iteration*

As one of OMiLAB its great strengths (Bork et al., 2019), the first iteration was specifically leaned on "fast prototyping".

Figure 57 displays the mBot movements where the different colours mirror the colours from the LEDs the mBot uses to represent the various transportation methods.

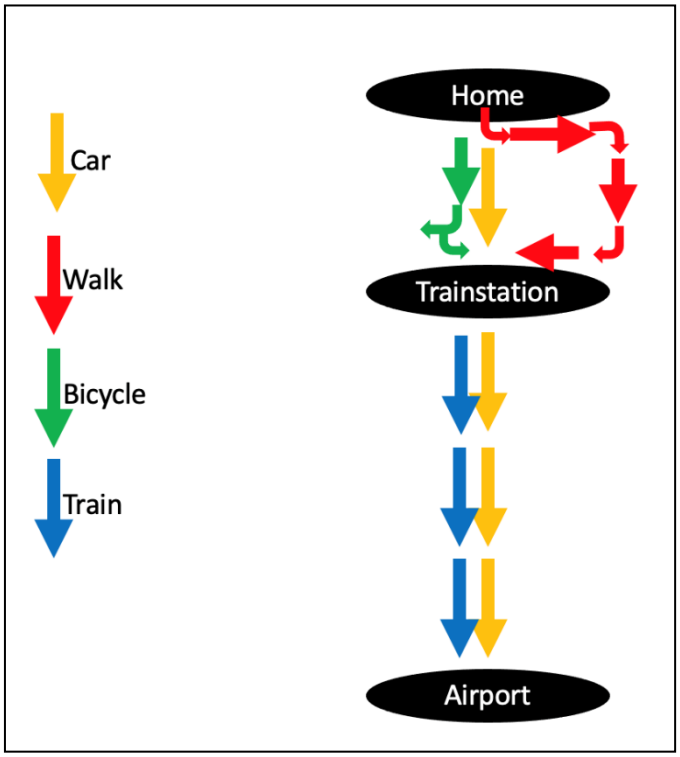



Figure 57 Graphical visualisation of the car movements

#### *7.4.1.5 Conclusion*

The first iteration set the goal to introduce the topic to the author and proof in a first step, the movement of a CPS with a model-based approach is possible. The next step is to create the whole thing in BPMN.

#### *7.4.1.6 Revised model*

In a second iteration, the street layouts, mirroring the one used in other project and the demonstration at the FHNW in Olten, have been created. After this, the street layouts

were conform with the one displayed in Figure 29. In terms of the base functions displayed during the chapter, the task moveForward have been replaced by the FollowLine task.

#### *7.4.1.7 Summary*

The implementation in this section has focused on functionality and an executable prototype. Importantly, it can be shown that a small number of different HTTP requests, executed in the form of a script, can lead to actual movements of a vehicle. An immense advantage is, that Bee-Up provides this built-in function to execute and the possibility to send HTTPS requests with the in-house scripting language ADOScript. Even though the tool Bee-Up was used, it is possible to execute ADOScript within the ADOxx Modelling Toolkit. To achieve that a DLL file must been added to the modelling tool installation path and initiated within the library. More information can be found on the ADOxx webpage[10].

### 7.4.2 Transformation to use the Camunda Workflow Engine

A possible implementation of the Trip planning scenario in the Camunda Modeler is presented in Figure 58 (mirroring Figure 39). It demonstrates the possibility to either model the routes via subprocesses or modelling the movement tasks one by one.

[10] https://www.adoxx.org/live/extended-http-requests-use

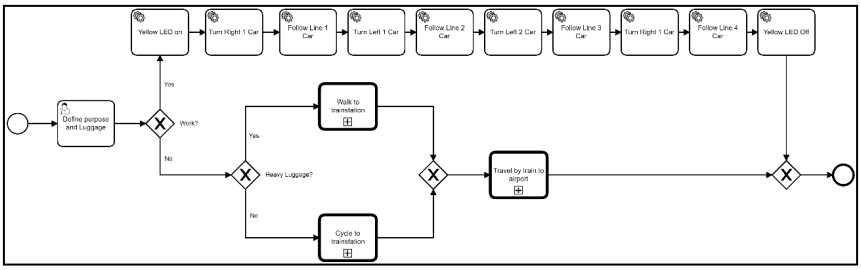



Figure 58 Camunda Modeler Trip planning overview

The model starts with a User Task to provide the inputs Luggage and Work for the decision making. At the gateways, more specifically the sequence flow, the string variable "form_WorkHoliday" resulting from the user task "Define purpose and Luggage" displayed in Figure 58 is compared to a string with the value "Work". The service tasks have been configured to send HTTP requests to the REST API of the mBot. Presented in Figure 59, the HTTP request is disseminated into multiple input parameters, namely the method, URL, header, script declaration and the final script that converts the beforementioned information to the JSON format.

The deployment can either be done to the localhost:8080, which requires a local running Tomcat instance, or to an online hosted platform of a provider, for example Heroku.

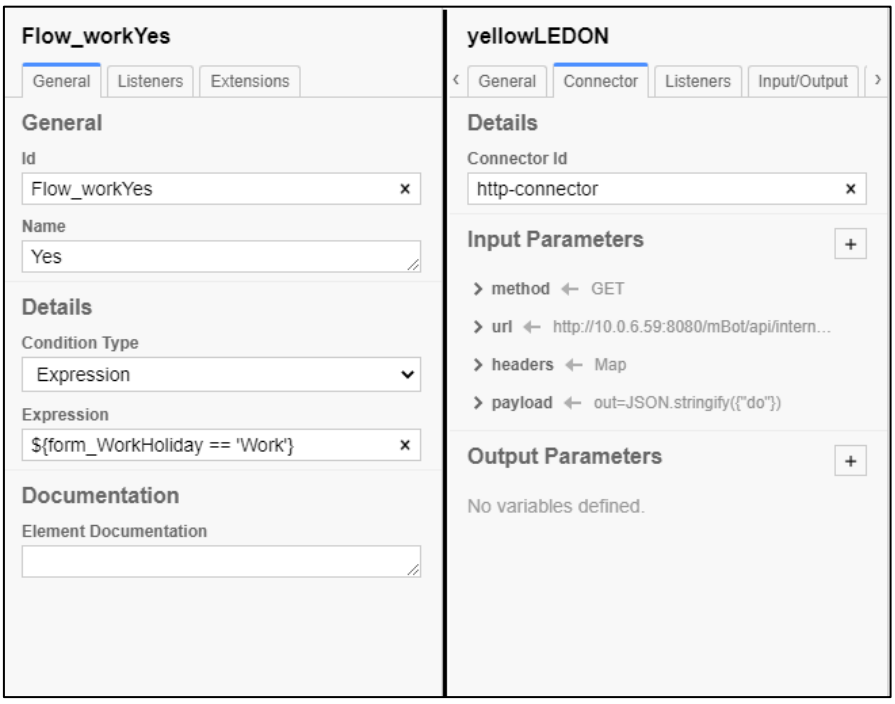



Figure 59 Example of a sequence flow and a Service Task with an HTTP-connector

Further clarifications regarding the service task configurations are provided with the following list:

- The method used is GET.
- The URL and command of the API to address the chosen motor or sensor realised.
- Headers to specify the content-type to application/json.
- The JSON payload, containing “do”.

After configuring a syntax-error free model it can directly be deployed via the Camunda Modeler. The deployment can then be accessed on the Camunda platform and appears as presented in Figure 60 with additional deployment information displayed left-hand side. On the top right-hand side, a process instance could now be started.

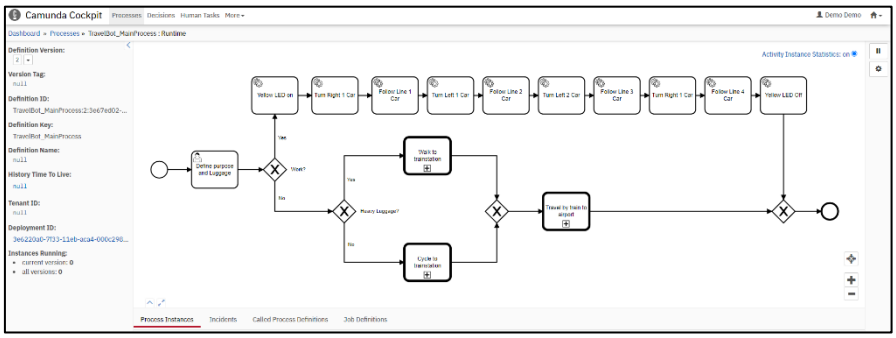



Figure 60 Deployed model on the Camunda workflow engine.

This chapter could now illustrate how to draw a BPMN model in the Camunda environment, adapt the configurations to the extended technical BPMN capabilities, and deploy it on a hosted java-based workflow engine platform.

## 7.5 Impressions

To provide some impression of how the street layout look in Olten, in Figure 61 a small collage is presented. On the top right recognizable with the glowing red LED the pedestrian simulating mBot make its way to the airport. The red arrow indicates the correct lane. On the bottom left and with a slightly offput angle the shining green coloured LED symbolises the bicycle mBot driving straight down from the starting point, with the green arrow indicating its correct position. The mBot with the yellow and the one with the blue LED colours are put directly to their lane, hence the missing arrows. The mBot with the yellow colour symbolising the car is part of the initial photo whereas the blue LED coloured train is added in.

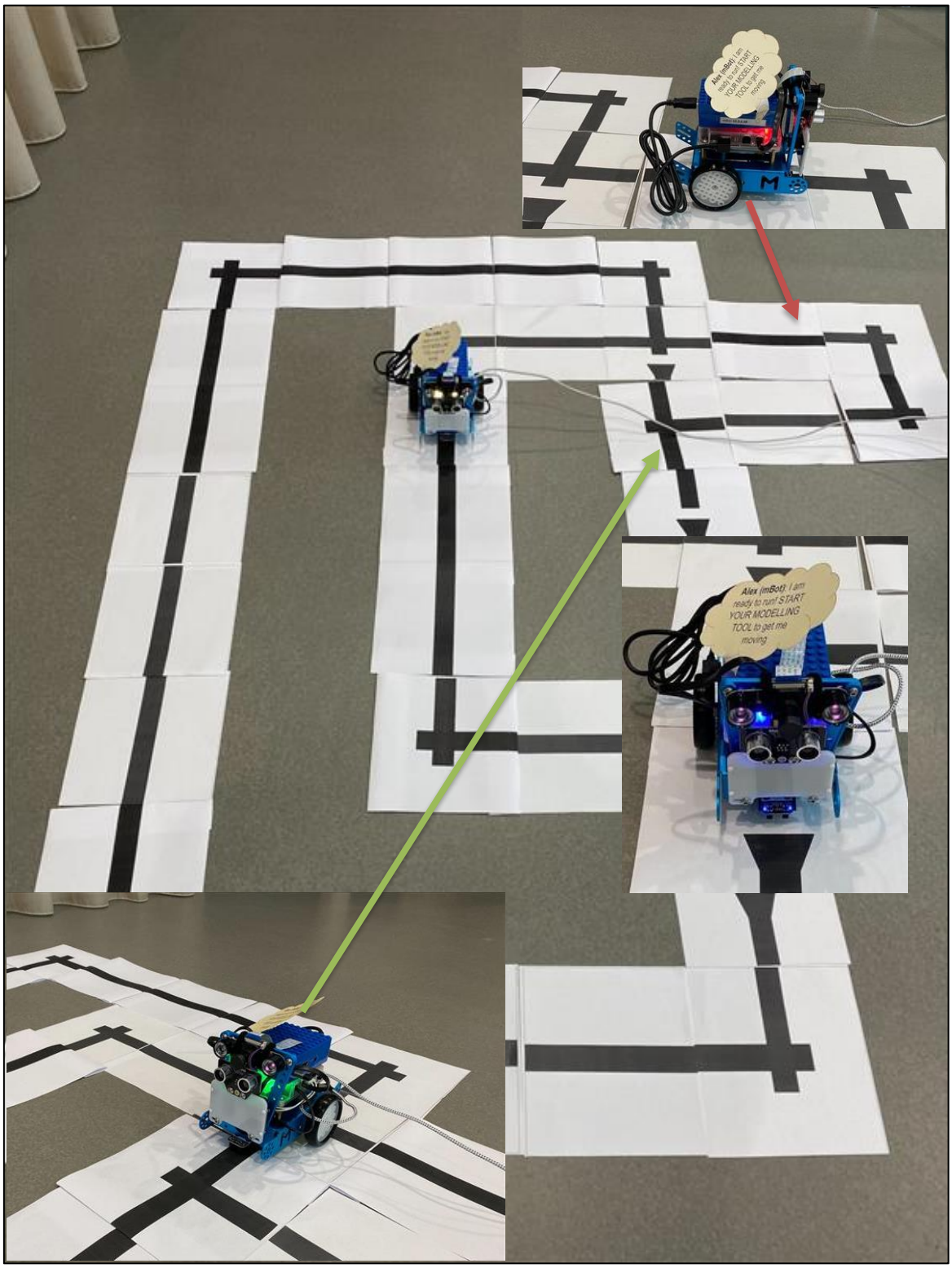



Figure 61 Live picture collage from the OMiLAB in Olten displaying the different colours in action

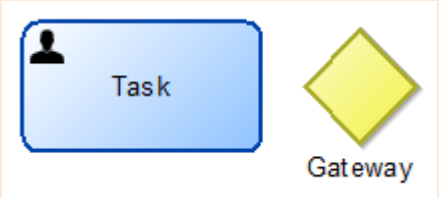
# 8 Evaluation

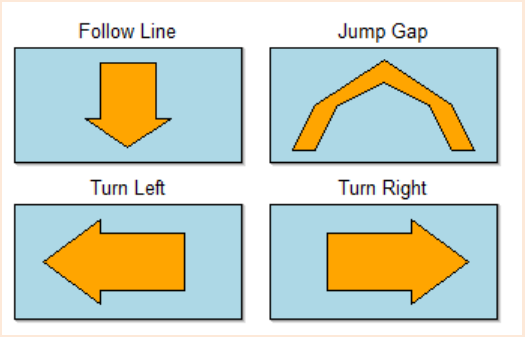
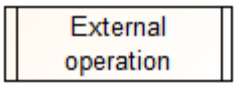
The DSR evaluation phase concerns the developed artefact. Vaishnavi, Kuechler, & Petter (2004) define the purpose of this phase that an artefact is evaluated according to criteria that are always implicit and frequently made explicit in the Awareness of Problem phase.

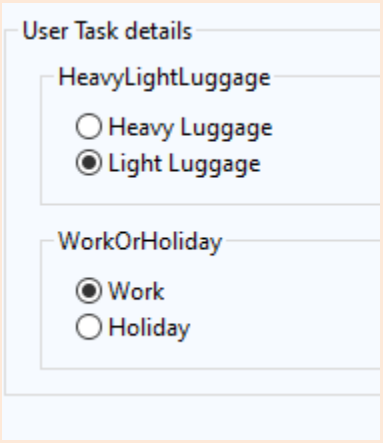
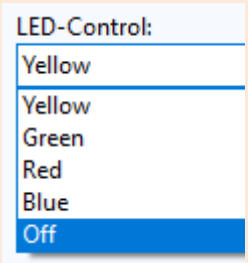
In the following, each requirement from Chapter 4 - Conceptualization is tested for fulfilment. While in chapter 6 the fulfilment of requirements has been partly demonstrated, this chapter individually addresses every requirement.

## 8.1 Comparison to requirements

In this section, the coverage of the elicited requirements with the functions of the proof of concepts BPMN4MoPla, including the execution, is assessed. Table 8 presents the requirements coverage for the DSML, Table 9 for the modelling tool, and Table 10 for the CPS infrastructure.

| Name | Description | Rationale |
|---|---|---|
| **REQ-010** | The DSML should accommodate business-like activities and decisions. | Next to the task element, a gateway element is shown below.   |

| | | |
|---|---|---|
| **REQ-011** | The DSML should accommodate model elements that are triggered by a process flow. | The sequence flow is indicating the process flow. Both execution approaches follow along the sequence flow in order determine the execution order. |
| **REQ-012** | The DSML should accommodate the ability to represent different transportation movements | Visually, the DSML provides a clear distinction.<br>  <br>In case of both execution approaches the difference is only recognisable if the name is chosen as a clear identifier. |
| **REQ-013** | The DSML should accommodate mechanisms such that a group of model elements can be incorporated into a model element having a higher abstraction. | The DSML as well as both execution approaches support the creation of sub-processes. While BPMN is using the "+" sign, flowcharts use an "External operation".<br>  |
| **REQ-020** | The DSML should accommodate constructs for | Within the DSML the User task has been modified that when the user task is |

| | | |
|---|---|---|
| | specifying the travelling purpose. | selected the Luggage and Purpose attributes are enabled to modify.<br>  <br>For the Bee-Up execution an AdoScript is used. Camunda handles attribute entries via manual task. |
| **REQ-021** | The DSML should accommodate attributes to define the luggage that is brought to the trip. | See REQ-020. |
| **REQ-022** | The DSML should accommodate the ability to differentiate between the transportation modes. | The DSML allows the Movement Type dropdown to choose the LED function and the LED-Control dropdown to choose the colour.<br>  |

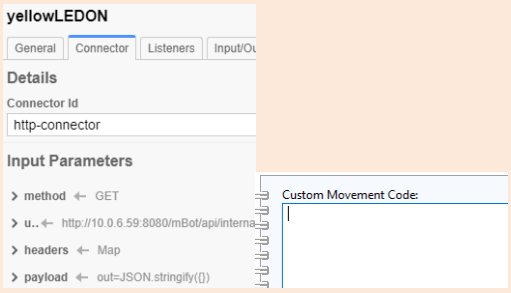
| | | |
|---|---|---|
| | | Similar to REQ-012 for the execution the transportation mode only differs if a describing name is chosen. |
| **REQ-030** | The DSML should accommodate scripts for the control of the car movement. For example, a script that execute the robotic car movement task. | This is displayed with a dropdown within the DSML but would require a workflow engine to have the movement scripts embedded. |
| **REQ-031** | The DSML should accommodate custom scripts for the control of the car movement. | The DSML foresees this with a CustomCode field that is then interpreted as Script by a workflow engine. While this is the case for the Bee-Up tool, Camunda provides the HTTP-connector.   |

Table 8 DSML requirements fulfilment

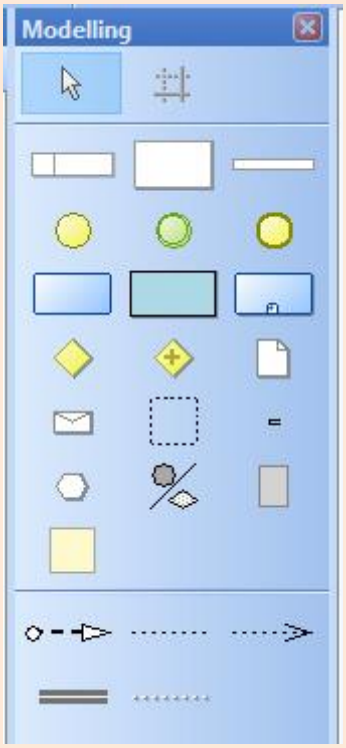
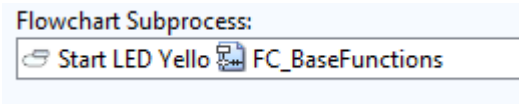
| Name | Description | Fulfilment |
|---|---|---|
| **REQ-101** | The modelling tool should display (through the palette) all the set of modelling constructs used for the mobility-planning scenario, especially those for the transportation movements. | The palette allows quick access to the modelling elements.<br>  |
| **REQ-102** | The modelling tool should provide interoperability with other modelling languages. | While possible in the DSML and the Bee-Up tool, Camunda is limited to DMN and CMN as additional modelling languages.<br>  |

Table 9 Modelling tool requirements fulfilment

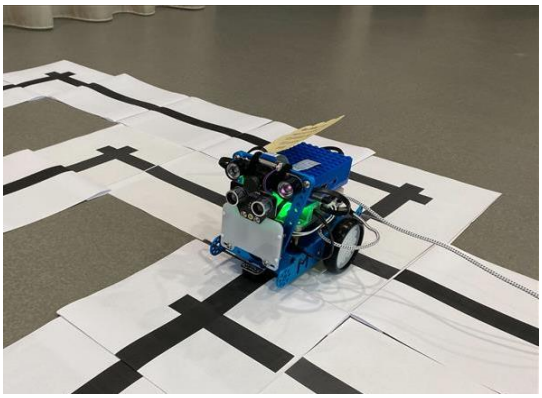
| Name | Description | Fulfilment |
|---|---|---|
| **REQ-201** | The CPS environment should provide connectivity to the cyber-physical space and be able to execute physical car movements. | The mBot features a Swagger UI REST API. When executing with Bee-Up and Camunda the tasks are executed in the modelled order. |
| **REQ-202** | The CPS environment should foresee a robotic car with sensors capable of detecting the physical space and follow predefine routes | The mBot is able to detect lines on the ground and has a colour programmable (R/G/B values) LED.  |
| **REQ-203** | The CPS environments should provide indicators to differentiate among the different transportation modes. | See REQ-202 |

Table 10 CPS environment requirements fulfilment

## 8.2 Conclusion

The only partial fulfilment has been noted from REQ-030 and is further elaborated below Table 11. This is, next to the inconvenience of using a workaround for the execution, a result of a missing open-source workflow engine for BPMN, which can be adapted in order to execute DSML elements, within the ADOxx environment.

| | | |
|---|---|---|
| REQ-030 | The DSML should accommodate scripts for the control of the car movement. For example, a script that execute the robotic car movement task. | This is displayed with a dropdown within the DSML but would require a workflow engine to have the movement scripts embedded. |

Table 11 Observed and elaborated deviations

A possible workaround to address this issue is, as discussed in chapter 7.4.1.1 Technical abstractions, to create one process flow for each desired function as displayed in Figure 62. Where in Bee-Up all flows can be stored in one process model, Camunda might require one process model for each function.

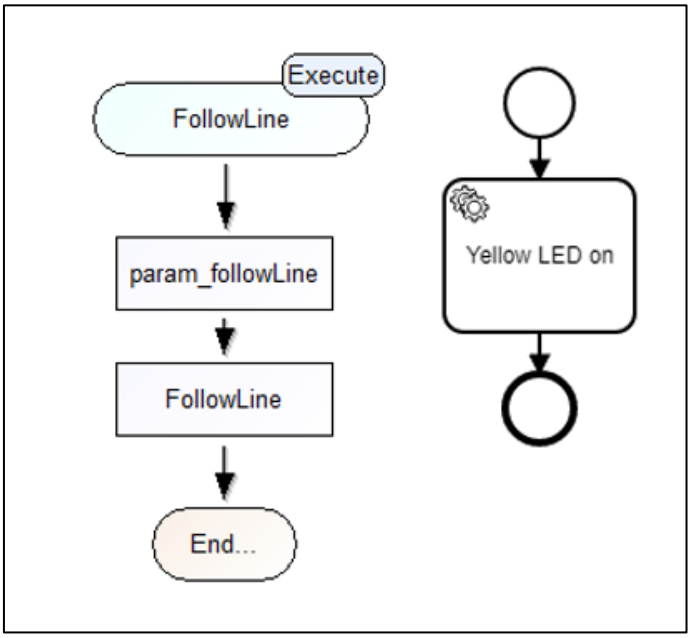



Figure 62 Technical implementation of a script

Modelling and executing the scenarios has proven that the proof-of-concept BPMN4MoPla mostly fulfils the presented requirements. The Proof of concept elaborated the need of a transformation in order to execute the modelled (extended) CPS. This outlines the opportunity for a future project to implement a workflow engine to execute BPMN within the ADOxx environment or similar to a recent project: Model-based framework for cyber-physical systems management in OMiLAB[11]. On the positive side, the Proof of concept has achieved the following improvements:

- More domain-specific usefulness with the addition of a separate element responsible for movements.
- Intuitive User guidance through conditionally available attributes.
- Separation of Business- and CPS elements.
- Advantages of BPMN remain.
- Usage of the Bee-Up library allows the interconnection of BPMN with Flowcharts.

On a last note it can be said that with the suggested proof of concept and the workaround to address gaps, the scenario of one person that intends to go to the airport, with a suggested transportation method based on the purpose and the luggage, that is then executed on a CPS with physical movement, can be completed.

[11] https://austria.omilab.org/psm/content/modelBasedFramework

# 9 Conclusion and Future Research

In the beginning, the following research question is stated:

*"How can modelling languages be designed for Mobility Planning, which allows to combine modelling of movements with the process- and decision logic of a business domain?"*

With the completion of this thesis, a proof of concept – BPMN4MoPla – was developed to answer the research question.

After the conducted literature review, it was apparent that mobility planning scenarios are only rarely tackled with business process modelling whilst connecting it with a CPS to generate a feedback loop. Also, BPMN extensions for IoT often focus on the modelling part and only rarely execute the models on an underlying CPS layer. The literature review also describes the wide range of problems and domains that can be tackled with domain-specific modelling and CPS. Some main advantages are simpler models, an easier modelling process, the integration of several languages and standards, better process quality and less modelling errors.

This thesis was conducted using design science research, and thus the main part consisted of awareness, suggestion, development, evaluation and ends with this chapter, the conclusion.

Especially during the modelling of the DSML, multiple transitions have been recognized. From meta modelling to a modelling language, designed in a tool, then from the tool into another environment for a practical execution in a working CPS. In practice multiple roles like domain experts, language experts, model experts and system architects shall collude during this kind of DSML approach. The author and the supervisors switched hats in order to cover these roles. As the time to develop the new modelling language and find

possibilities to execute the model with impact on the real world was limited to only a few months, there is potential for further development of the BPMN4MoPla library.

## 9.1 Future work and research

In order to directly the work commence this thesis, further approaches can be derived. On the one hand, the scenario can be specified with the help of domain experts in the form of a real project, and BPMN4MoPla can be adapted to its needs. This can be captured via further variables or the subdivision of the CarMovement element attributes for the most diverse real vehicles (train, bus, streetcar, car, airplane). In another approach, the different vehicles could be represented by multiple CPS. An idea as presented in Figure 63. Similarly, time, location, and multiple users influences the complexity of mobility planning enormous, could be further elaborated. On the other hand, a workflow engine that can run BPMN in the ADOxx environment (similar to Camunda), could be implemented. In the sense of the community, extended BPMN functions or whole elements (of course only with appropriate programming skills) could then be added to this engine iteratively. While the suggested additions of functions are more project-oriented than research. A design for an adaptable execution system, for configuring DSML elements might be interesting for research in Software System Design.

On another note, the value of a physical feedback loop as a result of a business model can be evaluated.

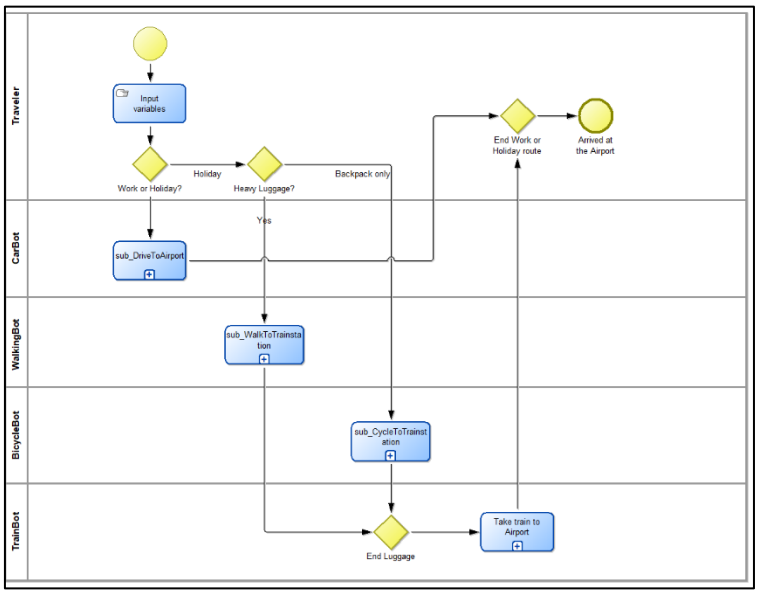



Figure 63 BPMN example of focussing on CPS identity

For the supervisors of FHNW, a learning could be achieved of how to approach future OMILAB theses (Bachelor and Master). As listed in the literature review, CPS is a hot topic in almost every domain imaginable. The already popular presentation with models that become even more tangible with a sample CPS to create a feedback loop is even more attractive.

# Bibliography

Acatech. (2011). *Cyber-Physical Systems - Driving force for innovations in mobility, health, energy and production*. Springer Berlin Heidelberg. Retrieved from https://www.springer.com/gp/book/9783642290909

Acumen. (2016). About Acumen. Retrieved from http://www.acumen-language.org/2016/04/about.html

Becker, J., Vom Brocke, J., Heddier, M., & Seidel, S. (2015). In Search of Information Systems (Grand) Challenges: A Community of Inquirers Perspective. *Business and Information Systems Engineering*, *57*(6), 377–390. https://doi.org/10.1007/s12599-015-0394-0

Bock, C. (2006). SysML and UML 2 support for activity modeling. *Systems Engineering*, *9*(2), 160–185. https://doi.org/doi.org/10.1002/sys.20046

Bork, D. (2018). Metamodel-based Analysis of Domain-specific Conceptual Modeling Methods, 17. https://doi.org/https://doi.org/10.1007/978-3-030-02302-7_11

Bork, D., Buchmann, R. A., Karagiannis, D., Lee, M., & Miron, E. T. (2019). An open platform for modeling method conceptualization: The OMiLAB digital ecosystem. *Communications of the Association for Information Systems*, *44*(1), 673–697. https://doi.org/10.17705/1CAIS.04432

Bork, D., Karagiannis, D., & Pittl, B. (2018). How are metamodels specified in practice? Empirical insights and recommendations. *Americas Conference on Information Systems 2018: Digital Disruption, AMCIS 2018*, (Amcis), 1–10.

Bork, D., & Miron, E.-T. (2017). OMiLAB - An Open Innovation Community for Modeling Method Engineering. *International Conference on Management and*

*Industrial Engineering*, (8), 64–77. Retrieved from https://manchester.idm.oclc.org/login?url=https://search.proquest.com/docview/1990423386?accountid=12253%0Ahttp://man-fe.hosted.exlibrisgroup.com/openurl/44MAN/44MAN_services_page?genre=unknown&atitle=OMiLAB+-+An+Open+Innovation+Community+for+Modeling+Met

Boulila, N. (2019). Cyber-Physical Systems and Industry 4 .0 : Properties , Structure , Communication , and Behavior. *Technical Report, Siemens Corporate Technology*, (April). https://doi.org/10.13140/RG.2.2.27890.76485

Braun, R. (2015). Towards the state of the art of extending enterprise modeling languages. *MODELSWARD 2015 - 3rd International Conference on Model-Driven Engineering and Software Development, Proceedings*, 394–402. https://doi.org/10.5220/0005329703940402

Braun, R., Schlieter, H., Burwitz, M., & Esswein, W. (2015). Extending a Business Process Modeling Language for Domain-Specific Adaptation in Healthcare. *Wirtschaftsinformatik Proceedings 2015*, (2015), 468–481. https://doi.org/10.1353/esc.0.0163

Braun, R., Schlieter, H., Burwitz, M., & Esswein, W. (2016). BPMN4CP revised - Extending BPMN for multi-perspective modeling of clinical pathways. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2016-March*, 3249–3258. https://doi.org/10.1109/HICSS.2016.407

Burwitz, M., Schlieter, H., & Esswein, W. (2013). Modeling Clinical Pathways - Design and Application of a Domain-Specific Modeling Language. *Wi*, (March), 1325–1339. Retrieved from http://aisel.aisnet.org/wi2013/83/

Burzynski, P., & Karagiannis, D. (2020). Bee-up - A teaching tool for fundamental conceptual modelling. *CEUR Workshop Proceedings*, *2542*, 217–221.

Camunda. (2021a). Camunda. Retrieved March 3, 2021, from https://camunda.com/solutions/

Camunda. (2021b). Camunda BPMN Extension elements. Retrieved from https://docs.camunda.org/manual/7.15/reference/bpmn20/custom-extensions/extension-elements/

Caracaş, A., & Bernauer, A. (2011). Compiling business process models for sensor networks. *2011 International Conference on Distributed Computing in Sensor Systems and Workshops, DCOSS'11*, (March). https://doi.org/10.1109/DCOSS.2011.5982159

Chatterjee, S., & Hevner, A. (2010). *Design Research in Information Systems: Theory and Practice*. *Springer* (Vol. 22). https://doi.org/10.1007/978-1-4419-5653-8

CIVITAS. (2021). Integrated & inclusive planning. Retrieved from Integrated & inclusive planning

Colloud, F. (2016). Advanced modelling of human movements using numerical optimisation. *Movement and Sports Sciences - Science et Motricite*, *90*(4), 1–3. https://doi.org/10.1051/sm/2016001

Compagnucci, I., Corradini, F., Fornari, F., Polini, A., Re, B., & Tiezzi, F. (2020). Modelling Notations for IoT-Aware Business Processes : a Systematic Literature Review, 1–13.

De Silva, V., Roche, J., & Kondoz, A. (2018). Robust fusion of LiDAR and wide-angle camera data for autonomous mobile robots. *Sensors (Switzerland)*, *18*(8). https://doi.org/10.3390/s18082730

Deka, L., Khan, S. M., Chowdhury, M., & Ayres, N. (2018). Transportation Cyber-Physical System and its importance for future mobility. In L. Deka & M. Chowdhury (Eds.), *Transportation Cyber-Physical Systems* (pp. 1–20). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-12-814295-0.00001-0

Derler, P., Lee, E. A., & Sangiovanni Vincentelli, A. (2012). Modeling cyber-physical systems. *Proceedings of the IEEE*, *100*(1), 13–28.

https://doi.org/10.1109/JPROC.2011.2160929

DHL Customer Solutions & Innovation report. (2016). Robotics in Logistics. *DHL Customer Solutions & Innovation*, (March), 37. Retrieved from http://www.dhl.com/content/dam/downloads/g0/about_us/logistics_insights/dhl_trendreport_robotics.pdf

Dibaji, S. M., Pirani, M., Flamholz, D. B., Annaswamy, A. M., Johansson, K. H., & Chakrabortty, A. (2019). A systems and control perspective of CPS security. *Annual Reviews in Control*, *47*, 394–411. https://doi.org/10.1016/j.arcontrol.2019.04.011

Domingos, D., & Martins, F. (2017). Using BPMN to model internet of things behavior within business process. *International Journal of Information Systems and Project Management*, *5*(4), 39–51. https://doi.org/10.12821/ijispm050403

Efendioglu, N., Woitsch, R., Utz, W., & Falcioni, D. (2017). ADOxx modelling method conceptualization environment. *Advances in Science, Technology and Engineering Systems*, *2*(3), 125–136. https://doi.org/10.25046/aj020317

Eltis. (2021). The SUMP concept. Retrieved from https://www.eltis.org/mobility-plans/sump-concept

Elvarsson, A. B. (2017). Modelling urban driving and stopping behavior for automated vehicles. *Semester Project, IVT, ETH Zürich, Zürich*, (June), 1–43.

Erickson, J., & Siau, K. (2013). Unified modeling language: The teen years and growing pains. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8016 LNCS*(PART 1), 295–304. https://doi.org/10.1007/978-3-642-39209-2-34

European Commision. (2013). Study to support an impact assessment of the urban mobility package - Final Report, (DG Move), 392. Retrieved from http://ec.europa.eu/transport/themes/urban/studies/doc/2013-10-urban-mobility-package-activity-31.pdf

Fragapane, G., de Koster, R., Sgarbossa, F., & Strandhagen, J. O. (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, (xxxx). https://doi.org/10.1016/j.ejor.2021.01.019

Frank, U. (2010). Outline of a Method for Designing Domain-Specific Modelling Languages. *ICB Research Reports*, (42).

Frank, U. (2013). Domain-Specific Modeling Languages – Requirements Analysis and Design Guidelines. *Domain Engineering: Product Lines, Languages, and Conceptual Models*, (May 2013), 1–404. https://doi.org/10.1007/978-3-642-36654-3_6

Frank, U. (2014). Mehrebenen-ModellierungMultilevel Modeling. *Wirtschaftsinformatik*, *56*(6), 347–367. https://doi.org/10.1007/s11576-014-0438-y

Froschauer, R., & Lindorfer, R. (2019). Workflow-based programming of human-robot interaction for collaborative assembly stations. *ARW & OAGM Workshop 2019*. https://doi.org/10.3217/978-3-85125-663-5-14

Fuglestvedt, J., Berntsen, T., Myhre, G., Rypdal, K., & Skeie, R. B. (2008). Climate forcing from the transport sectors. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(2), 454–458. https://doi.org/10.1073/pnas.0702958104

Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, *8*, 108952–108971. https://doi.org/10.1109/ACCESS.2020.2998358

Geisler, R., Klar, M., & Pons, C. F. (1998). Dimensions and Dichotomy in Metamodeling, (June). https://doi.org/10.14236/ewic/nfm1998.10

Glaessgen, E. H., & Stargel, D. S. (2012). The digital twin paradigm for future NASA and U.S. Air force vehicles. *Collection of Technical Papers* -

*AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, (April). https://doi.org/10.2514/6.2012-1818

Gora, P., & Rüb, I. (2016). Traffic Models for Self-driving Connected Cars. *Transportation Research Procedia*, *14*, 2207–2216. https://doi.org/10.1016/j.trpro.2016.05.236

Graja, I., Kallel, S., Guermouche, N., & Kacem, A. H. (2016). BPMN4CPS: A BPMN extension for modeling cyber-physical systems. *Proceedings - 25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2016*, 152–157. https://doi.org/10.1109/WETICE.2016.41

Hevner, A., & Chatterje, S. (2004). Design Science Research in Information Systems Overview of Design Science Research. *Ais*, 45. https://doi.org/10.1007/978-1-4419-5653-8

Huang, J., Bastani, F., Yen, I. L., Dong, J., Zhang, W., Wang, F. J., & Hsu, H. J. (2009). Extending service model to build an effective service composition framework for cyber-physical systems. In *IEEE International Conference on Service-Oriented Computing and Applications, SOCA' 09* (pp. 130–137). IEEE. https://doi.org/10.1109/SOCA.2009.5410453

Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., … Zhang, L. (2017). The Internet-of-Things Meets Business Process Management: Mutual Benefits and Challenges, (September). Retrieved from http://arxiv.org/abs/1709.03628

Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., … Zhang, L. (2020). The Internet of Things Meets Business Process Management: A Manifesto. *IEEE Systems, Man, and Cybernetics Magazine*, *6*(4), 34–44. https://doi.org/10.1109/msmc.2020.3003135

Karagiannis, D. (2015). Agile modeling method engineering. *ACM International*

*Conference Proceeding Series*, *01-03-Octo*, 5–10. https://doi.org/10.1145/2801948.2802040

Karagiannis, D., Burzynski, P., & Miron, E. (2017). *Open | Models The " IMKER " Case Study*. https://doi.org/10.5281/zenodo.345846

Karagiannis, D., & Kühn, H. (2002a). Metamodelling platforms. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2455*, 182. https://doi.org/10.1007/3-540-45705-4_19

Karagiannis, D., & Kühn, H. (2002b). Metamodelling Platforms. In K. Bauknecht, A. Min Tjoa, & G. Quirchmayer (Eds.), *Proceedings of the Third International Conference EC-Web at DEXA 2002*. Berlin: Springer-Verlag.

Karagiannis, D., Mayr, H. C., & Mylopoulos, J. (2016). *Domain-Specific Conceptual Modeling*. *Conceptual Modeling for Discrete-Event Simulation*. https://doi.org/10.1201/9781439810385-p5

Karagiannis, D., & Muck, C. (2017). OMiLAB Physical Objects (OMiPOB). Retrieved from https://www.omilab.org/assets/docs/OMiROB_description_draft.pdf

Khaitan, S. K., & McCalley, J. D. (2015). Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems Journal*, *9*(2), 350–365. https://doi.org/10.1109/JSYST.2014.2322503

Kumar, R., C, C., A, A., & Anjali, A. (2021). Internet of Things (IOT). *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3832727

Lah, O. (2018). Sustainable Urban Mobility in Action. In *Sustainable Urban Mobility Pathways: Policies, Institutions, and Coalitions for Low Carbon Transportation in Emerging Countries* (pp. 133–282). Elsevier. https://doi.org/10.1016/B978-0-12-814897-6.00007-7

Lal, R. (2017). Business Process Management with the Camunda Workflow Engine. Retrieved February 4, 2021, from https://www.srijan.net/blog/business-process-management-camunda-workflow#:~:text=The Camunda BPM engine is,kind and size of organisation.

Lamballais, T., Roy, D., & De Koster, M. B. M. (2017). Estimating performance in a Robotic Mobile Fulfillment System. *European Journal of Operational Research*, *256*(3), 976–990. https://doi.org/10.1016/j.ejor.2016.06.063

Laurenzi, E. (2020). *An Agile and Ontology-Aided Approach for Domain-Specific Adaptations of Modelling Languages By*.

Laurenzi, E., Hinkelmann, K., Reimer, U., Van Der Merwe, A., Sibold, P., & Endl, R. (2017). DSML4PTM: A domain-specific modelling language for patient transferal management. *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems*, *3*, 520–531. https://doi.org/10.5220/0006388505200531

Lee, E. A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors (Switzerland)*, *15*(3), 4837–4869. https://doi.org/10.3390/s150304837

Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, *37*(4), 316–344. https://doi.org/10.1145/1118890.1118892

Negri, E., Fumagalli, L., & Macchi, M. (2017). A Review of the Roles of Digital Twin in CPS-based Production Systems. *Procedia Manufacturing*, *11*, 939–948. https://doi.org/10.1016/j.promfg.2017.07.198

Niemueller, T., Karpas, E., Vaquero, T., & Timmons, E. (2016). Planning Competition for Logistics Robots in Simulation. *Icaps*.

Okraszewska, R., Romanowska, A., Wołek, M., Oskarbski, J., Birr, K., & Jamroz, K.

(2018). Integration of a Multilevel Transport System Model into Sustainable Urban Mobility Planning. *Sustainability*, *10*(2). https://doi.org/10.3390/su10020479

OMG. (2013). Business Process Model and Notation (BPMN). Retrieved January 20, 2021, from https://www.omg.org/spec/BPMN/2.0.2/PDF

OMG. (2016). ABOUT THE CASE MANAGEMENT MODEL AND NOTATION SPECIFICATION VERSION 1.1. Retrieved January 20, 2021, from https://www.omg.org/spec/CMMN

OMG. (2020). ABOUT THE DECISION MODEL AND NOTATION SPECIFICATION VERSION 1.3. Retrieved January 22, 2021, from https://www.omg.org/spec/DMN

OMiLAB. (2021). OMiLAB a Nonprofit Organization. Retrieved March 2, 2021, from https://www.omilab.org/

Sabegh, M. J., Lukyanenko, R., & Recker, J. (2017). Conceptual modeling research in information systems: What we now know and what we still do not know, (May), 19–20. Retrieved from https://eprints.qut.edu.au/107955/

Saunders, M., Lewis, P., & Thornhill, A. (2019). *Chapter 4: Understanding research philosophy and approaches to theory development*. *Research Methods for Business Students*.

Schönig, S., Ackermann, L., Jablonski, S., & Ermer, A. (2018). An integrated architecture for IoT-aware business process execution. *Lecture Notes in Business Information Processing*, *318*(May), 19–34. https://doi.org/10.1007/978-3-319-91704-7_2

Scratch. (2021). Create stories, games, and animations. Retrieved from https://scratch.mit.edu

Sprinkle, J., Rumpe, B., Vangheluwe, H., & Karsai, G. (2010). *Metamodelling - State of the Art and Research Challenges*.

SysML$^{TM}$, O. (2017). OMG Systems Modeling Language, 362. Retrieved from

https://www.omg.org/spec/SysML/1.5/PDF

Taha, W. M., Taha, A.-E. M., & Thunberg, J. (2021). *Cyber-Physical Systems : A Model-Based Approach*. Springer.

Takeda, H., Veerkamp, P., Tomiyama, T., & Yoshikawa, H. (1990). Modeling design processes. *AI Magazine*, *11*(4), 37–48.

Tao, F., Qi, Q., Wang, L., & Nee, A. Y. C. (2019). Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. *Engineering*, *5*(4), 653–661. https://doi.org/10.1016/j.eng.2019.01.014

The Open Group. (2019). ArchiMate® 3.0.1 Specification. Retrieved January 22, 2021, from https://pubs.opengroup.org/architecture/archimate3-doc/chap15.html

Torres, V., Serral, E., Valderas, P., Pelechano, V., & Grefen, P. (2020). Modeling of IoT devices in Business Processes: A Systematic Mapping Study. *Proceedings - 2020 IEEE 22nd Conference on Business Informatics, CBI 2020*, *1*, 221–230. https://doi.org/10.1109/CBI49978.2020.00031

Towards Datascience. (2019). Deep Pi Car. Retrieved March 20, 2021, from https://towardsdatascience.com/tagged/deep-pi-car

Tullis, T. (Thomas), & Albert, B. (William). (2013). *Measuring the user experience : collecting, analyzing, and presenting usability metrics*. Elsevier.

Vaishnavi, V., Kuechler, B., & Petter, S. (2004). DESIGN SCIENCE RESEARCH IN INFORMATION SYSTEMS, (1), 1–66. https://doi.org/1756-0500-5-79 [pii]\r10.1186/1756-0500-5-79

van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific languages: an annotated bibliography. *ACM SIGPLAN Notices*, *35*(6), 26–36. https://doi.org/10.1145/352029.352035

Völter, M., Stahl, T., Bettin, J., Haase, A., & Helsen, S. (2013). *Model-Driven Software*

*Development: Technology, Engineering, Management*. West-Sussex, England: John Wiley & Sons.

WEF. (2021). Shaping the Future of Mobility. Retrieved from https://www.weforum.org/platforms/shaping-the-future-of-mobility

Wiederkehr, S., Frommenwiler, T., & Walti, F. (2018). Literaturrecherche Handlungsempfehlungen "Mobility as a Service (MaaS)," (11), 39. Retrieved from https://www.bav.admin.ch/dam/bav/de/dokumente/themen/mmm/literaturrecherche-maas-awk.pdf.download.pdf/Literaturrecherche Handlungsempfehlungen «Mobility as a Service (MaaS)» - AWK.pdf

Yousfi, A., Bauer, C., Saidi, R., & Dey, A. K. (2016). UBPMN: A BPMN extension for modeling ubiquitous business processes. *Information and Software Technology*, *74*, 55–68. https://doi.org/10.1016/j.infsof.2016.02.002

Yousfi, A., Freitas, A. de, Dey, A. K., & Saidi, R. (2015). The Use of Ubiquitous Computing for Business Process Improvement. *IEEE Transactions on Services Computing*, *9*(4), 621–632. https://doi.org/10.1109/TSC.2015.2406694

Zarour, K., Benmerzoug, D., Guermouche, N., & Drira, K. (2019). A systematic literature review on BPMN extensions. *Business Process Management Journal*, (2013). https://doi.org/10.1108/BPMJ-01-2019-0040

Zeng, Y., Rose, C., Brauner, P., Taha, W., Masood, J., Philippsen, R., … Cartwright, R. (2014). Modeling basic aspects of cyber-physical systems, part II (Extended Abstract). *Proceedings - 16th IEEE International Conference on High Performance Computing and Communications, HPCC 2014, 11th IEEE International Conference on Embedded Software and Systems, ICESS 2014 and 6th International Symposium on Cyberspace Safety and Security*, (August), 550–557. https://doi.org/10.1109/HPCC.2014.119

Zou, B., Xu, X., Gong, Y. (Yale), & De Koster, R. (2018). Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. *European Journal of*

*Operational Research*, *267*(2), 733–753. https://doi.org/10.1016/j.ejor.2017.12.008

# Figures

# Tables

# Appendix / Appendices

All the artifacts can be found here:
https://drive.switch.ch/index.php/s/zE49YcH8D3Y3xbl

Within the BPMN4MoPla folder the following files can be found:

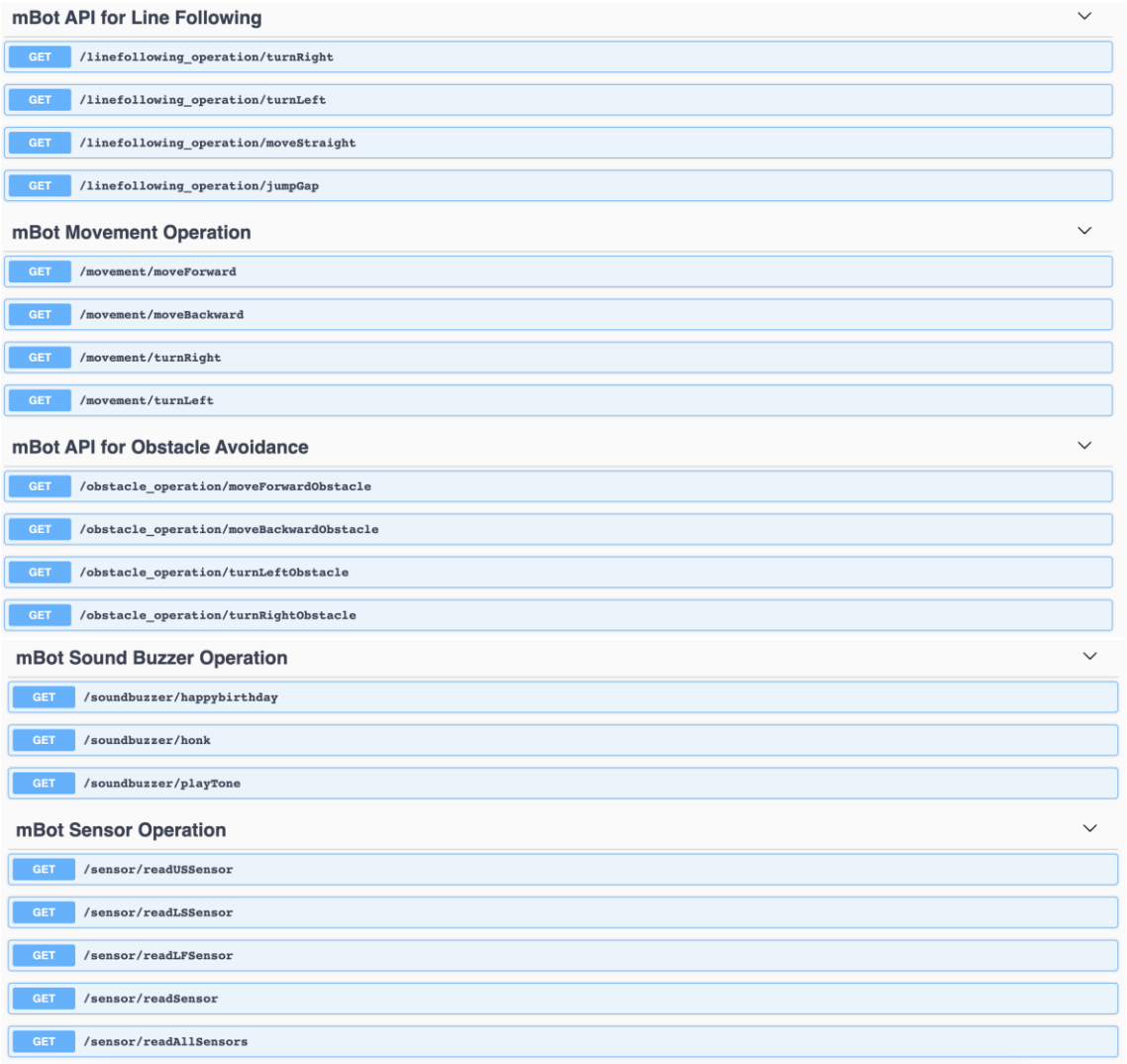
- Readme.txt; This file contains a brief guide how to import the diverse files.
- The camunda-bpmn-tomcat-7.12.0.rar; contains the used Camunda environment from the authors MacBook.
- ADOxxDevelopmentToolkit_BPMN4MoPla_Library.abl; contains the library that can be imported within the ADOxx Development toolkit.
- ADOxxModellingToolkit_BPMN4MoPla_ModelExport.adl; contains the models of the BPMN4MoPla DSML and can be imported into the ADOxx Modelling Toolkit.
- Bee-UpExecution_BPMN4MoPla.adl; contains the models of the Bee-Up execution approach that can be imported into the Bee-Up Tool.
- travelBot*.bpmn; the 4 .bpmn files contain the models that can be opened with the Camunda Modeler and then be deployed to the Camunda Workflow engine.

# MBot HTTP requests ADOScript

```
1.  _____Turn  Right   on  line_____
2.  SETL str_roboturl:(      "http://10.0.6.59:8080/mBot/api/linefollowing_operation/"                )
3.  SETL map_headers:({  "Content  - Type" : "application/json"       })
4.
5.  SETL str_roboturl:(      "http://10.0.6.5       9:8080/mBot/api/linefollowing_operation/"          )
6.  HTTP_SEND_REQUEST(str_roboturl+"turnRight?speed=100"    )
7.  str_method:(   "GET")
8.  map_reqheaders:   (map_headers)
9.  str_reqbody:(    "do" )
10. val_respcode:val_httpcode
11. map_respheaders:map_respheaders
12. str_respbody:str_re      spbody
13.
14. _____Straight    Forward_____
15.
16. SETL
17. str_roboturl:(      "http://10.0.6.59:8080/mBot/api/linefollowing_operation/"                )
18. SETL map_headers:({  "Content  - Type" :  "application/json"       })
19.
20. HTTP_SEND_REQUEST(str_roboturl+     "moveStraight?speed=100"    )
21. str_method:(   "GET")
22. map_reqheaders:   (map_headers)
23. str_reqbody:(    "do" )
24. val_respcode:val_httpcode
25. map_respheaders:map_respheaders
26. str_respbody:str_respbody
27.
28.
29. _____Jump Gap_____
30.
31. HTTP_SEND_REQUEST("http://10.0.6.59:8080/mBot/api/linefollowing_operation/jum
    pGap?speed=100")
32. str_method:(   "GET")
33. map_reqheaders:   ({ "Content  - Type" :  "application/json"       })
34. str_reqbody:(    "do" )
35. val_re  spcode:val_httpcode
36. map_respheaders:map_respheaders
37. str_respbody:str_respbody
38.
39.
40. _____T Cross   left_____
41.
42. SETL
43. str_roboturl:(      "http://10.0.6.59:8080/mBot/api/linefollowing_operation/"                )
44. SETL map_headers:({  "Content  - Type" :  "application/json"       })
```

```
45.
46.
47. SETL str_roboturl:        ( "http://10.0.6.59:8080/mBot/api/operation"                )
48. HTTP_SEND_REQUEST(str_roboturl+      "turnLeft?speed=60&secs=1"     )
49. str _method:( "GET")
50. map_reqheaders:   (map_headers)
51. str_reqbody:(    "do" )
52. val_respcode:val_httpcode
53. map_respheaders:map_respheaders
54. str_respbody:str_respbody
55.
56. ____________________________Jump Gap with   UI_____________________________________
57.
58. SETL
59. str_roboturl:(       "http://10.0.6.59:8080/mBot/api/linefollowing_operation/"                    )
60. SETL map_headers:({  "Content - Type" :  "application/json"        })
61.
62. CC "AdoScript"     INFOBOX( "Stop   finished?"    )
63. HTTP_SEND_REQUEST(str_roboturl+       "jumpGap?speed=100" )
64. str_method:(   "GET")
65. map_reqheaders:(map_headers)
66. str_reqbody:(    "do" )
67. val_respcode:val_httpcode
68. map_respheaders:map_respheaders
69. str_respbody:str_respbody
```

# MBot API commands


mBot API for Line Following
GET /linefollowing_operation/turnRight
GET /linefollowing_operation/turnLeft
GET /linefollowing_operation/moveStraight
GET /linefollowing_operation/jumpGap
mBot Movement Operation
GET /movement/moveForward
GET /movement/moveBackward
GET /movement/turnRight
GET /movement/turnLeft
mBot API for Obstacle Avoidance
GET /obstacle_operation/moveForwardObstacle
GET /obstacle_operation/moveBackwardObstacle
GET /obstacle_operation/turnLeftObstacle
GET /obstacle_operation/turnRightObstacle
mBot Sound Buzzer Operation
GET /soundbuzzer/happybirthday
GET /soundbuzzer/honk
GET /soundbuzzer/playTone
mBot Sensor Operation
GET /sensor/readUSSensor
GET /sensor/readLSSensor
GET /sensor/readLFSensor
GET /sensor/readSensor
GET /sensor/readAllSensors