



# TU Clausthal

Due to the rapid development of information and communication technology, a new communication network, which combines the "real world" and "virtual world" together and recognizes the whole combination as one closely interacting system, has emerged. This network is the cyber-physical system (CPS). The CPS with long lifetime or lifecycle is defined as long-living cyber-physical system (LL-CPS).

In many areas, especially in smart factories, digital manufacturing and smart logistics, the LL-CPS plays a very important role and should be always operated to meet continuous and fast-changing requirements, for example, the proliferation of business models, the rapid development of technology, the fast-changing market and customer requirements etc.

However, the development of a LL-CPS is accompanied by risks and high expenditures. Generally, a LL-CPS cannot be defined perfectly at the beginning. The multiple interactions between cyber and physical parts in a LL-CPS make engineering difficult. Changes of these parts can generate some problems during evolution of a LL-CPS. At present, approaches or procedures have not been defined. This reduces risks of development and optimizes expenditures of system implementation. Based on a uniform formal description of LL-CPSs, this thesis provides an approach to guarantee the consistency between system evolution requirements and system implementation during the managed evolution of LL-CPSs. Furthermore, this approach is also used to optimize expenditures of system implementation.

Daning Wang Managed Evolution of Long-Living CPSS

Vol. 22 2020



Daning Wang

## Managed Evolution of Long-Living Cyber-Physical Systems

ISSE-Dissertation 22



Institute for Software and Systems Engineering  
Lehrstuhl von Prof. Dr. Andreas Rausch

### **Bibliografische Information der Deutschen Nationalbibliothek**

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN 978-3-8439-4685-8

Dissertation Clausthal, SSE-Dissertation 22, 2020  
D104

Chairperson of the Board of Examiners  
Prof. Dr. Sven Hartmann

Chief Reviewer  
Prof. Dr. Andreas Rausch

2. Reviewer  
Prof. Dr. Jörg P. Müller

3. Reviewer  
Prof. Dr. Rijia Ding

Date of oral examination: June 10, 2020

© Coverbild: <https://stock.adobe.com/de/228460259>

© Verlag Dr. Hut, München 2021  
Sternstr. 18, 80538 München  
Tel.: 089/66060798  
[www.dr.hut-verlag.de](http://www.dr.hut-verlag.de)

Die Informationen in diesem Buch wurden mit großer Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und ggf. Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene fehlerhafte Angaben und deren Folgen.

Alle Rechte, auch die des auszugsweisen Nachdrucks, der Vervielfältigung und Verbreitung in besonderen Verfahren wie fotomechanischer Nachdruck, Fotokopie, Mikrokopie, elektronische Datenaufzeichnung einschließlich Speicherung und Übertragung auf weitere Datenträger sowie Übersetzung in andere Sprachen, behält sich der Autor vor.

1. Auflage 2021

# Managed Evolution of Long-Living Cyber-Physical Systems

Doctoral Thesis

(Dissertation)

to be awarded the degree of

Doctor rerum naturalium (Dr. rer. nat)

submitted by

**Daning Wang**

from Fuxin province Liaoning

approved by Faculty of Mathematics/Computer Science and Mechanical Engineering

Clausthal University of Technology

Chairperson of the Board of Examiners  
Prof. Dr. Sven Hartmann

Chief Reviewer  
Prof. Dr. Andreas Rausch

2. Reviewer  
Prof. Dr. Jörg P. Müller

3. Reviewer  
Prof. Dr. Rijia Ding

Date of oral examination: June 10, 2020

Für meine Tochter



## **Abstract**

The continued rapid information technological progress is causing major changes. From embedded systems to intelligent embedded systems and cyber-physical systems, the system evolution is always confronted with the challenge of the frequently changing requirements: the proliferation of business models, the rapid development of technology, the fast changing market and customer requirements, new varieties of development methods and models, etc. Cyber-physical systems (CPS) with a long-term life cycle (long-living) play a very important role in many areas, especially in smart factories, digital manufacturing, smart logistics, and energy efficiency. It combines the physical part with the cyber part in a holistic way, where the two parts have to flexibly and dependably adapt to each other to adapt to the changing system environment. On the other hand, a long-living CPS is complicated and multi-configurational, so an incompatible or non-combinable system development can lead to problems or high expenditures. Therefore, an approach is required to reduce the evolution risks and investment needs during the evolution of long-living CPSs.

This thesis provides an approach for the managed evolution of long-living CPSs, which is based on the formal descriptions and model transformations of managed evolution of the long-living CPSs. This approach guarantees the consistency between the system evolution requirements and system implementation. Furthermore, with this approach the influence factors for investment needs like the cost of implementation can be scaled to optimize the expenditures of implementation. This approach is evaluated with two practical cases to ascertain the suitability for the managed evolution of long-living CPSs.





# Danksagung

An dieser Stelle bietet sich die Gelegenheit mich bei Allen zu bedanken, die mich bei der Erstellung dieser Dissertation unterstützt haben. Diese Arbeit entstand als Doktorand am Lehrstuhl von Prof. Dr. Andreas Rausch an der Technischen Universität Clausthal.

Zunächst bedanke ich mich bei Prof. Dr. Andreas Rausch für die Übernahme der Doktorvaterschaft wodurch die Promotion überhaupt erst zustande gekommen ist. Seine tatkräftige Unterstützung bei der Meinungsbildung während der Konzeptionsphase hat wesentlich zum Gelingen der wissenschaftlichen Arbeit beigetragen. Prof. Dr. Andreas Rausch hat es zu jedem Zeitpunkt verstanden, mich zu den jeweils nächsten Schritten und schlussendlich zur Einreichung dieser Arbeit zu motivieren.

Bei Prof. Dr. Jörg P. Müller bedanke ich mich für die prompte Bereitschaft, die Erstellung des Zweitgutachtens zu übernehmen.

Auch bedanke ich mich herzlich bei Dr. rer. nat. habil. Christoph Knieke für seine Bereitschaft, sowie die hilfreichen Kommentare und Verbesserungsvorschläge.

Für die Hilfe und Unterstützung meiner Arbeit möchte ich mich auch bei meinen Kollegen am Institute for Software and Systems Engineering an der Technischen Universität Clausthal bedanken. Ohne ihre Hilfe und Unterstützung wäre diese Arbeit bestimmt nicht das geworden, was sie heute ist.

Des Weiteren gilt der Dank meinen Eltern, meinem Onkel und meiner Tante, die ein besonderes Verständnis und Rücksicht auf meine Doppelbelastung genommen haben und sich stets um den Stand der Arbeit besorgt gezeigt haben.

Im Besonderen danke ich meiner Frau, die besonders während der Erstellung der Promotionschrift so manche Stimmungsschwankung mit viel Entgegenkommen erwidert hat und durch viele fürsorgliche Worte einige für mich hoffnungslose Situationen zum Besseren gewendet hat.

Daning Wang

Clausthal-Zellerfeld, den 18.12.2019



# Contents

1	Introduction .....	1
1.1	Long-Living Cyber-Physical Systems .....	2
1.2	Motivation .....	5
1.3	Goals of this Work .....	6
1.4	Structure and Content of the Thesis .....	6
2	Basics and Fundamentals.....	9
2.1	Graph theory.....	10
2.1.1	Graph.....	10
2.1.2	Graph operation .....	12
2.1.3	Walks and paths .....	13
2.1.4	Graph search .....	17
2.2	Optimization problem .....	19
2.3	Systems development models.....	20
2.4	Model-driven development.....	22
2.4.1	System modeling languages.....	22
2.4.2	Metamodeling .....	25
2.4.3	Mathematical description of models .....	26
2.4.4	Model transformation.....	27
2.4.5	Model mapping .....	29
2.4.6	Operation of models .....	29
3	Problem Statement and Analysis with Example.....	31
3.1	A sample of LL-CPS: A conveyor system with ASRS.....	32
3.1.1	Process-oriented description .....	34
3.1.2	Component-oriented description .....	36
3.2	Managed evolution scenario of LL-CPS .....	50
3.2.1	Problems of the ongoing LL-CPS .....	50
3.2.2	The targeted status of this LL-CPS.....	50
3.3	State of the art and existing approaches for managed evolution of LL-CPSs .....	52
3.3.1	Cyber physical system modeling .....	52

3.3.2	Formal modeling of system evolution .....	57
3.3.3	Modeling and optimizing the costs of reconstruction .....	59
3.4	Research questions of this thesis .....	59
4	Formal Descriptions and Transformations of Managed Evolution of LL-CPSs .....	61
4.1	Formal description for VSM.....	68
4.1.1	Formal semantical foundation .....	68
4.1.2	Model-based description .....	71
4.1.3	Semantical mapping .....	76
4.1.4	Concrete modeling .....	80
4.2	Formal description for IBD .....	84
4.2.1	Formal semantical foundation .....	84
4.2.2	Model-based description .....	87
4.2.3	Semantical mapping .....	93
4.2.4	Concrete modeling .....	95
4.3	Formal mapping relation from VSM to IBD .....	97
4.3.1	Formal semantical foundation .....	98
4.3.2	Model-based description .....	101
4.4	Formal managed evolution of LL-CPSs .....	103
4.4.1	Formal semantical foundation .....	103
4.4.2	Model-based description .....	106
5	Solution Approach Overview .....	109
5.1	Problems formalization .....	111
5.2	Semantical mapping .....	113
5.3	Generating graph solutions .....	114
5.3.1	Reforming the mapping domain .....	115
5.3.2	Creating the mapping codomain.....	116
5.3.3	Path morphism .....	118
5.3.4	Graphs combination .....	119
5.4	Concrete modeling .....	119
5.5	Optimizing the model solutions .....	121
6	Implementation .....	123

6.1	System requirements .....	123
6.1.1	Use case diagram .....	123
6.1.2	Input models restructuring .....	124
6.2	System architecture.....	125
6.2.1	System structure .....	125
6.2.2	System behavior .....	127
6.3	Functions realization .....	128
7	Evaluation .....	131
7.1	Case study 1: Conveyor System with ASRS.....	131
7.1.1	Managed evolution of LL-CPS by using the approach.....	132
7.1.2	Comparison of solutions .....	136
7.1.3	Development risk evaluation .....	136
7.1.4	Economic evaluation .....	137
7.2	Case study 2: Project “Synus” .....	138
8	Conclusion and Outlook.....	141
8.1	Conclusion .....	141
8.2	Outlook.....	142
9	Publication Bibliography .....	143
10	List of Figures .....	149
11	List of Tables .....	153
12	List of Definitions .....	155
13	Appendixes.....	157
13.1	Pseudocodes for oppositeSearch .....	157
13.2	Comparing code changes .....	157



# 1 Introduction

## Content

---

1.1 Long-Living Cyber-physical system

1.2 Motivation

1.3 Goals of this Work

1.4 Structure and Content of the thesis

---

Today's world is entering in the networking era. Due to the rapid development of information and communication technology, a new communication network has emerged, which combines the "real world" and "virtual world" together and recognizes the whole combination as one closely interacting system. This network is named the cyber-physical system (CPS).

In many areas, especially in smart factories, digital manufacturing, smart logistics, and energy efficiency, the CPS plays a very important role. For instance, modern mechanical engineering products are increasingly being supplemented by programmable controllers. In diverse areas of application like automobile or production automation, the trend is to network the embedded systems with each other, but also to integrate them into a higher digital level [1].

A long-living cyber-physical system (LL-CPS) is defined as a huge CPS with a complex system architecture, long time life cycle and dynamic changing system boundary. In this chapter, the definitions and characteristics of CPS, long-living systems and LL-CPSs will first be introduced. The ongoing LL-CPS requires continuous operation to meet the fast changing requirements. Section 1.2 introduces the motivation of the evolution of LL-CPSs.

An approach is regarded as the goal of this work and it is used for the managed evolution of LL-CPSs with the local minimal costs and controlled risks in the operations, which is introduced in section 1.3. The structure and content of this thesis are outlined in section 1.4.

## 1.1 Long-Living Cyber-Physical Systems

### What is a Cyber-Physical System?

There are varied understandings about CPSs. The first proposition of a CPS was characterized by Helen Gill in the USA to describe a connection between physical processes with a calculation share. The National Science Foundation (NSF) declared it a core research point of national research work at the end of 2006. Since then, many different definitions and descriptions of CPSs have been developed. The general definitions or descriptions are based on the conceptual definition of Helen Gill [2].

Because there is no standard and prevailing definition and description of CPSs, some popular definitions and descriptions are selected from two different perspectives and introduced in this section.

- From the perspective of integration and networking:

R. Rajkumar defined a CPS as “a physical and engineered system whose operations are monitored, coordinated, controlled and integrated by a computing and communication core” [3].

Edward A. Lee defined a CPS as “the integration of computation with physical processes.” The computation part can control and monitor the physical processes, and the physical processes affect the computations with feedback. These interaction effects will be implied with the integration and networking between the computation and physical part [4].

A.Sangiovanni-Vincentelli described a combination system with “a cyber side (computing and networking) with a physical side (mechanical, electrical and chemical processes)” as a definition of a CPS [5].

- From a software-intensive perspective:

In the description of H. Giese et al., “Cyber-physical Systems (CPSs) present a unified view of computing systems that interact strongly with their physical environment” [6].

Compared to traditional embedded systems, CPSs are more modular, dynamic, networked and large-scale and can provide more computing. I. Gerostathopoulos described that CPSs increasingly depend on software, and the software part has become their most intricate and extensive constituent [7].

E. Geisberger and M.Broy defined a CPS as “an open and networked system.” It uses sensors to identify situations and information in the physical world and makes them available to the network-based services, as well as acting directly on



the processes in the physical world through actuators to control the behaviour of devices, things, and services [8]. In addition, in the publication of M. Broy software-intensive systems are regarded as innovation drivers that enable new functionalities to be developed in different application areas [1].

### **Characteristics of Cyber-Physical System**

Although there are so many different definitions and descriptions of CPSs, the core descriptions are essentially the same, e.g. the CPS connects the cyber and physical worlds. The following characteristics position a CPS with five aspects [8]:

- (1) Fusing of the physical and virtual world
- (2) System of systems with dynamic changing of system boundaries
- (3) Context-adaptive and fully or partially autonomous system and active control in real time
- (4) Cooperative systems with distributed and alternative controls
- (5) Comprehensive human-system cooperation

### **What is a Long-Living System?**

A long-living system is a system that has a long lifetime or lifecycle. Such systems are usually used as industrial software systems.

“Software systems in the industrial automation domain are typically long-living systems, i.e., some of them may operate for more than 20 years, because of the investment in the underlying machines and devices. Examples for such systems are distributed control systems to manage industrial processes, such as power generation, manufacturing, chemical production, or robotics systems to automate manual tasks, such as welding, pick & place, or sealing” [9].

Not only software systems but also the connected hardware and mechanical systems in the industrial domain are typically long-living systems. U. Goltz et al. expressed that “nowadays, software relies on several independent or loosely coupled components using complex technology stacks comprising hardware, middleware and reusable software components and other (software) systems” [10]. Figure 1 shows the lifecycle of hardware/software systems, which comprise various areas: technical system, platform and software ([10] quoted in [11]).

In the work of U. Goltz et al., the evolutionary life cycle of a technical system in automation engineering includes two phases: the design and construction phase and operation phase (the vertical axis in Figure 1).

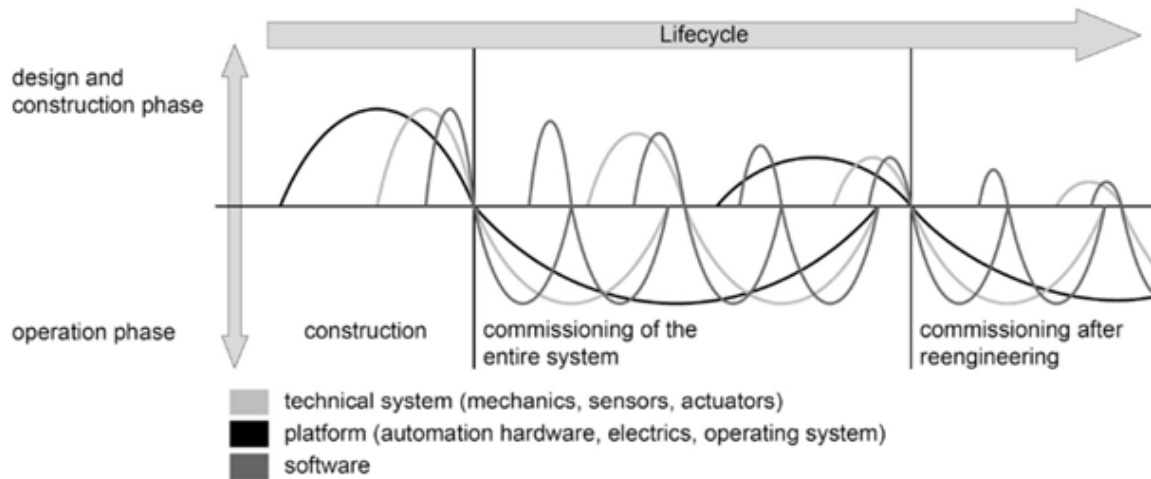


Figure 1: Integration of development and operation of hardware/software systems

The requirements of new products or products changes are defined and specified in a design and construction phase, which is understood as the driving force of system evolution in the lifecycle (the lifetime is the horizontal axis from left to right in Figure 1). These requirements in the design and construction phase will be implemented in the operation phase. In Figure 1, the three different grey scales express the three areas: technical system, platform and software and the height of each curve indicates the amount of effort required in these areas.

Before the adaptations and updates of changes starts, the system may have to be shut down to commission the entire system. During the commissioning time, the system needs to be re-engineered to meet new design requirements. After the re-engineering, the system proceeds to the next commissioning and operation.

In Figure 1, the technical system and platform have a smaller change frequency than in software engineering, because they are running for more years than software. However the amount of effort of the technical system or platform is much more compared with software engineering.

### What is a Long-Living Cyber-Physical System?

A long-living cyber-physical system (LL-CPS) is understood as a CPS with a long lifetime or lifecycle. The closely-dependent relationship between different areas: technical system, platform and software in a long-living hardware/software system also exists in a LL-CPS. In addition, a LL-CPS has the following additional characteristics:

#### Characteristics of LL-CPSs

- A LL-CPS is a large-scale and complex system.
- The components in a LL-CPS always have different lifecycles and come from different suppliers.
- A LL-CPS has a multi-level hierarchy system architecture with multi-domains.

- Usually a LL-CPS has monolithic structure and cannot be defined perfectly at the beginning.

## 1.2 Motivation

Although a LL-CPS is a large-scale and long-living system, it need to be in constant evolution and should be always operated to meet the continuous and fast-changing requirements [6].

- System repeatedly extended: in general, a complex system is not defined perfectly at the beginning. Sometime, the LL-CPS has to monitor itself for its health. Besides being driven by the availability of new technology, the LL-CPS is repeatedly enhanced and extended in its prolonged life time.
- Different lifetime of parts: the different parts or components of LL-CPS have rather different life times. Some lower-cost elements like sensors and computer platforms tend to have short life time, while the costly and individual parts like production equipment generally have longer life time.
- Changing requirements of market: the LL-CPS needs to be continuously developed and objective to the dynamic changes of requirements of the market or needs of customs. Nowadays, these changes tend to be much faster.
- Changing environments: a LL-CPS is usually used in an open and unrestricted environment. Therefore it must be developed to adapt to the changes in the environment.

However, the development of LL-CPS comes with risks and high expenditures. Not only the combination of cyber and physical parts makes the engineering much more difficult but also many other characteristics of LL-CPS can generate some problems during the evolution of LL-CPSs [6].

- The changes can occur at different levels in a LL-CPS: the elements in a LL-CPS allows at the same time belong to different subsystems. So the changes of these elements can affect the structures in different subsystems. The multiple interactions and side effects that can arise in the system can become very complex.
- A large-scale system: like most large-scale systems, a LL-CPS is not easily partitioned, analysed and operated. It presents a monolithic structure. "That makes the interaction among the decision variables very hard to separate".[5]

- In general, it cannot be shut down: in general, a LL-CPS cannot be shut down for system development or evolution. Therefore, it should support developing or evolving on a running system.

At present, there are no defined approaches or certification procedures ensuring the safe, secure and economical evolution of LL-CPSs. In order to meet the needs for safety and economy, an approach or procedure should be developed and established.

### **1.3 Goals of this Work**

The goal of this work is to develop an approach for the managed evolution of a LL-CPS with the local optimal costs and controlled risks in ongoing operations. This goal can be structured in three sub-goals:

- 1. Define a description method to uniformly represent the requirements of system evolution and the plan of implementation of a LL-CPS:**

A LL-CPS can be described from different aspects. Each description aspect focuses on certain features of a LL-CPS to make design and analysis. In this thesis, a uniform formal description method of LL-CPSs is necessary to define the requirements of system evolution and the plan of system implementation concurrently.

- 2. Derivate the requirements of a LL-CPS evolution to plan of implementation:**

With the uniform formal description of LL-CPSs, the requirements of system evolution have to be derived to the plan of implementation. This can be used to control the risks of the managed evolution of a LL-CPS.

- 3. Optimize the costs of reconstruction:**

The costs of system evolution can be understood as the costs of reconstruction during the evolution of a LL-CPS in this thesis. A local optimal solution in a set of solutions is defined as the local optimal costs during the managed evolution of a LL-CPS.

### **1.4 Structure and Content of the Thesis**

An overview of the structure of this thesis is provided below.

In chapter 2, the basics and fundamentals in this thesis are introduced, including the relevant definitions and knowledge for the following chapters.

Chapter 3 analyses the problems during the managed evolution of a LL-CPS based on an example. Accordingly, the LL-CPS is modeled with two different modeling methods. One is

## Chapter 1 - Introduction

used to model the evolution requirements of this LL-CPS, while the other one models this LL-CPS from the implementation aspect. In addition, the existing research and approaches for the problems during the managed evolution of LL-CPSs will be also introduced in this chapter. At the end of this chapter, the evolution problems are summarized with four research questions.

In chapter 4, a uniform formal description of LL-CPSs is introduced to achieve the sub-goal 1 of this thesis. The models formed with two different modeling methods can be transformed to the uniform formal descriptions to derivate the evolution requirements to the plan of implementation.

An approach is introduced in Chapter 5 to solve the problems during the managed evolution of a LL-CPS. This approach is considered as the achievement of the sub-goals 2 and 3 of this work.

Chapter 6 presents the implementation of this approach. The design and architecture of the implementation are also introduced in this chapter.

In chapter 7, the evaluation of this approach is introduced with a concrete project.

The final chapter discusses the contributions of this work and highlights possible avenues for further works.



## 2 Basics and Fundamentals

### Content

---

- 2.1 Graph theory
    - 2.1.1 Graph
    - 2.1.2 Graph operation
    - 2.1.3 Walks and paths
    - 2.1.4 Graph search
  - 2.2 Optimization problem
  - 2.3 Systems development models
  - 2.4 Model-driven development
    - 2.4.1 System modeling languages
    - 2.4.2 Metamodeling
    - 2.4.3 Mathematical description of models
    - 2.4.4 Model transformation
      - 2.4.4.1 Bidirectional model transformation
      - 2.4.4.2 Unidirectional model transformation
    - 2.4.5 Model mapping
    - 2.4.6 Operation of models
- 

The relevant basics and fundamentals for this thesis are introduced in this chapter to reach a common understanding.

First, the important definitions and algorithms in graph theory are introduced as the mathematical foundations for the formal modeling of managed evolution of LL-CPS.

Subsequently, the local minimum cost problem is a typical application of optimization problem. Section 2.2 introduces the basic understanding of local minimum cost by using the standard and popular example.

In section 2.3, three classical system development models are introduced, namely the waterfall model, phased implementation model and prototyping model. They are used to specify how the activities are organized in the total system evolution.

Then, the model-driven development (MDD) is introduced, which focuses on creating and exploiting domain models. A common way for system evolution description is the usage of the term process-oriented models. In this chapter, a process-oriented modeling method named value steam mapping is introduced. On the other hand, for planning the implementation and costs associated with reconstruction component-oriented models are necessary. Here, two

component-oriented modeling methods named block definition diagram and internal block diagram will be introduced. In this thesis, the value stream mapping and internal block diagram are used to model the managed evolution of LL-CPS. Subsequently, the definition of metamodeling is introduced, which is used to analyse, construction and development of models. Two definitions of model transformation the bidirectional model transformation and the unidirectional model transformation are introduced in this chapter, which will be used for model transformation in the following chapters. The mapping relationship between models is introduced in the final section.

## 2.1 Graph theory

Feature models or variable models can be described in graph structures. The behaviors of the variants can be also represented in graph-based structures [11]. Therefore, the graph-based structure and graph theory play important roles in system engineering. In this section, some important basics and fundamentals about graph theory will be introduced.

### 2.1.1 Graph

A directed graph  $g$  is formalized with a four-tuple structure [12], [13]:

Definition 1

$$g := (V, E, src, tgt)$$

$$src := E \rightarrow V$$

$$tgt := E \rightarrow V$$

$$E := V \times V$$

$V$  is a finite set of vertices.  $E$  is a finite set of edges. The function  $src$  represents the relationship for any edge to its source vertex, like the function  $tgt$  represents the relationship for any edge to its target vertex. For every edge  $e_i \in E$ , there are one source vertex  $v_i \in V$  and one target vertex  $v_j \in V$ , so an edge from  $v_i$  to  $v_j$  is represented as  $(v_i, v_j)$ .

A set of directed graphs is defined with a set  $G$  [14]:

Definition 2

$$G := \{g\}$$

In a labelled directed graph  $g_w$ , all of the vertices and edges can be labelled. Though a labelling of the vertices in a graph that represents a mapping function  $ct := V_w \rightarrow A$ , where  $A$  is a set of labels. A labelling of the edges is also in the same way  $wt := E_w \rightarrow B$ , where  $B$  is a set of



labels. Often, these labels are numbers or colors, which can be called “weights” of vertices or edges [14], [15].

Definition 3

$$g_w := (V_w, E_w, src_w, tgt_w, ct, wt, A, B)$$

$$src_w := src |_{E_w \rightarrow V_w}$$

$$tgt_w := tgt |_{E_w \rightarrow V_w}$$

$$E_w = V_w \times V_w$$

$$ct := V_w \rightarrow A$$

$$wt := E_w \rightarrow B$$

$A := a \text{ set of labels}$

$B := a \text{ set of labels}$

The labels can be understood as the description information of vertices and edges in graphs. A **directed connected graph structure** is defined by the labellings of the vertices and edges, which are used to save the description information as the attributes in the vertices and edges.

Definition 4

$$g_{in} := (V_{in}, E_{in}, src_{in}, tgt_{in}, attributes_{in}, Key_{in}, Value_{in})$$

$$src_{in} := src |_{E_{in} \rightarrow V_{in}}$$

$$tgt_{in} := tgt |_{E_{in} \rightarrow V_{in}}$$

$$E_{in} = V_{in} \times V_{in}$$

$$attributes_{in} := (V_{in} \cup E_{in}) \rightarrow (Key_{in} \rightarrow Value_{in})$$

$Key_{in} := a \text{ set of strings}$

$Value_{in} := a \text{ set of strings}$

The function  $attributes_{in}$  maps every vertex and edge to the owned attributes, which are represented with a key-value structure comprising a set of strings  $Key_{VSM}$  and a set of strings  $Value_{VSM}$ . The relationship from the keys to values are represented with a hash function. A set of graphs  $\{g_{in}\}$  can be represented with  $G_{in}$ .

Definition 5

$$G_{in} := \{g_{in}\}$$

Another representation for this directed connected graph structure is using a set of vertices.

Definition 6

$$g_{in} := \{V_{in}\}$$

### 2.1.2 Graph operation

Graphs  $g_1$  and  $g_2$  are two directed connected graph graphs in  $G_{in}$ . The operator  $\subseteq$  represents that a graph  $g_2$  is a subgraph of a directed Graph  $g_1$  [12], [14]:

Definition 7

	$g_1 \in G_{in}$ and $g_2 \in G_{in}$
if	$V_2 \subseteq V_1$ $V_2 \neq \emptyset$
	$E_2 \subseteq E_1$
	$src_2 := src_1  _{E_2 \rightarrow V_2}$
	$tgt_2 := tgt_1  _{E_2 \rightarrow V_2}$
then	$g_2 \subseteq g_1$

If the graph  $g_2$  is a subgraph of  $g_1$ , the graph  $g_1$  must include all vertices and edges of the graph  $g_2$ . The function  $src_2$  is the function  $src_1$  for the sets  $E_2$  and  $V_2$ , as same as the function  $tgt_2$  is the function  $tgt_1$  for the sets  $E_2$  and  $V_2$ . Accordingly, the graphs  $g_1$  and  $g_2$  must have the same connection structure for the sets  $E_2$  and  $V_2$ .

The binary operation is used to create a new graph from two initial graphs. The graph union for two graphs  $g_1$  and  $g_2$  is represented with the operator "+".

Definition 8

$g_1 \in G_{in}$ and $g_2 \in G_{in}$
$g_3 = g_1 + g_2$
$V_3 = V_1 \cup V_2$
$E_3 = E_1 \cup E_2$
$src_3 := src  _{E_3 \rightarrow V_3}$
$tgt_3 := tgt  _{E_3 \rightarrow V_3}$
$g_1 \subseteq g_3$

$$g_2 \subseteq g_3$$

The graph difference for one graph compared with another is represented with the operator “-”. The new created graph is a subgraph in the initial graph.

Definition 9

$$g_1 \in G_{in} \text{ and } g_2 \in G_{in}$$

$$g_4 = g_1 - g_2$$

$$V_4 = V_1 \setminus V_2$$

$$E_4 = E_1 \setminus E_2$$

$$src_4 := src |_{E_4 \rightarrow V_4}$$

$$tgt_4 := tgt |_{E_4 \rightarrow V_4}$$

$$g_4 \subseteq g_1$$

A graph homomorphism of two graphs is defined as a mapping function  $hg$  for the vertices and edges from one graph to another [16], [17]. Simply write for graph homomorphism is  $hg(g_1) = g_2$ . There are different kinds of graph homomorphism, for example, the injective homomorphism, the subjective homomorphism, the bijective homomorphism (isomorphism) and the covering maps. The mapping from a vertex to a subgraph (representing with a set of vertices) or the mapping from edge to a path (representing with a set of edges) are also allowed in graph homomorphism.

Definition 10

$$g_1 \in G_{in} \text{ and } g_2 \in G_{in}$$

$$\forall v_i \in V_1 \exists (hg(v_i) := v_{i'} \wedge (v_{i'} \in V_2))$$

$$\forall (v_i, v_{i+1}) \in E_1 \rightarrow (hg(v_i), hg(v_{i+1})) \in E_2$$

### 2.1.3 Walks and paths

The walks and paths can be understood as the special forms of graphs. A walk from  $v_1$  to  $v_n$  in a directed graph  $g_1 \in G_{in}$  is defined with a finite sequence of vertices [12], [14], where  $v_1$  is the initial vertex and  $v_n$  is the terminal vertex. Every vertex in a walk is allowed to appear more than once.

Definition 11

$$w(v_1, v_n) := (v_1, \dots, v_i \dots, v_n)$$

$$g_1 \in G_{in}$$

$$\{v_1, \dots, v_i \dots, v_n\} \in V_1$$

$$\{(v_1, v_2) \dots (v_{n-1}, v_n)\} \in E_1$$

$$w(v_1, v_n) \subseteq g_1$$

$$\forall i = 1, \dots, n \quad n \in \mathbb{N}$$

A path is defined similarly as a walk and it is a special walk [18], [13], where  $v_1$  is the initial vertex and  $v_n$  is the terminal vertex. Every vertex in a path is allowed to appear only once. If a path  $p(v_1, v_n)$  is in the graph  $g_1$ , it can be written as  $p(v_1, v_n) \subseteq g_1$ .

Definition 12

$$p(v_1, v_n) := (v_1, \dots, v_i \dots, v_n)$$

$$g_1 \in G_{in}$$

$$\{v_1, \dots, v_i \dots, v_n\} \in V_1$$

$$\{(v_1, v_2) \dots (v_{n-1}, v_n)\} \in E_1$$

$$p(v_1, v_n) \subseteq g_1$$

$$\forall i = 1, \dots, n \quad n \in \mathbb{N}$$

A path is a special graph, so a path morphism is defined as a multivalued mapping function from a path in a graph to a set of paths in another graph (see Definition 10). An example of multivalued function is a square roots function  $\sqrt{x} = y$ , if  $x=4$ , dann  $y=2$  and  $-2$ .

There are two graphs  $g_1 \in G_{in}$  and  $g_2 \in G_{in}$ , and there is a path  $p(v_1, v_n)$  in  $g_1$ . The mapping function  $pv$  maps the initial vertex  $v_1$  and the terminal vertex  $v_n$  of the path  $p(v_1, v_n)$  to a vertex  $v_{1'}$  and a vertex  $v_{n'}$  in graph  $g_2$ . All the paths  $p_j(v_{1'}, v_{n'})$  in the graph  $g_2$ , which are begins from  $v_{1'}$  and ends to  $v_{n'}$ , are mapped with the multivalued function  $pm$ .

Definition 13

$$pm := p(v_1, v_n) \rightarrow \{p_j(v_{1'}, v_{n'})\}$$

$$g_1 \in G_{in} \quad \text{and} \quad g_2 \in G_{in}$$

$$\begin{aligned}
 p(v_1, v_n) &\subseteq g_1 \\
 pv &:= V_1 \rightarrow \mathcal{P}(V_2) \\
 pv(v_1) &:= v_{1'} \quad v_{1'} \in V_2 \\
 pv(v_n) &:= v_{n'} \quad v_{n'} \in V_2 \\
 pm(p(v_1, v_n)) &:= \{p_j(v_{1'}, v_{n'}) \mid p_j(v_{1'}, v_{n'}) \subseteq g_2\}
 \end{aligned}$$

In path morphism, the vertices and edges between the initial vertex  $v_1$  and the terminal vertex  $v_n$  in  $p(v_1, v_n)$  in  $g_1$  do not need to be mapped. One path in  $g_1$  can have any number of mapped paths in  $g_2$ .

Figure 2 shows an example of path morphism from path  $p(1,2) = (1,3,2)$  in graph  $g_1$  to a set of paths in graph  $g_2$ .

There are  $pv(1) = 11$  and  $pv(2) = 14$ ,  
 then the mapping results are:

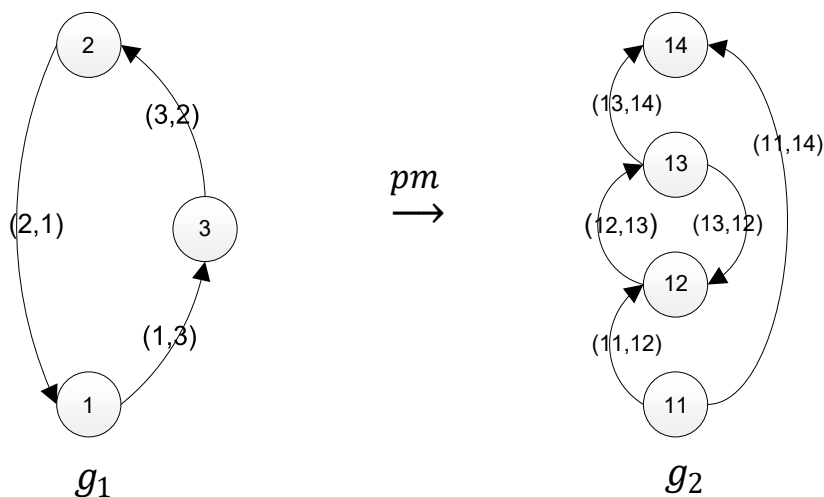
$$pm(p(1,2)) = \begin{cases} p(11,14) = (11,14) \\ p(11,14) = (11,12,13,14) \end{cases}$$


Figure 2: A path morphism example

The definition of walk morphism is formalized with a multivalued mapping function  $wm$  from a walk in a graph to a set of walks in another graph. There are two graphs  $g_1$  and  $g_2$ , and there is a walk  $w(v_1, v_n)$  in  $g_1$ . The mapping function  $wv$  maps the initial vertex  $v_1$  and the terminal vertex  $v_n$  of  $w(v_1, v_n)$  to a vertex  $v_{1'}$  and a vertex  $v_{n'}$  in graph  $g_2$ . There  $v_{1'}$  and  $v_{n'}$  are the initial vertex and terminal vertex in walks  $w_j(v_{1'}, v_{n'})$ .

Definition 14

$$wm := w(v_1, v_n) \rightarrow \{w_j(v_{1'}, v_{n'})\}$$

$$g_1 \in G_{in} \text{ and } g_2 \in G_{in}$$

$$w(v_1, v_n) \subseteq g_1$$

$$wv := V_1 \rightarrow \mathcal{P}(V_2)$$

$$wv(v_1) := v_{1'} \quad v_{1'} \in V_2$$

$$wv(v_n) := v_{n'} \quad v_{n'} \in V_2$$

$$wm(w(v_1, v_n)) := \{w_j(v_{1'}, v_{n'}) \mid w_j(v_{1'}, v_{n'}) \subseteq g_2\}$$

An example of walk morphism from a walk  $w_1(1,2) = (1,3,2)$  in graph  $g_1$  to a set of walks in graph  $g_2$  is shown in Figure 3. In this example, graph  $g_2$  has a cycle, so the mapped walks in  $g_2$  have to be defined as the walks that include the same vertex maximal twice.

There are  $wv(1) = 11$  and  $wv(2) = 14$ ,

then the mapping results are:

$$wm(w_1(1,2)) = \begin{cases} w_2(11,14) = (11,14) \\ w_2(11,14) = (11,12,13,14) \\ w_2(11,14) = (11,12,13,12,13,14) \end{cases}$$

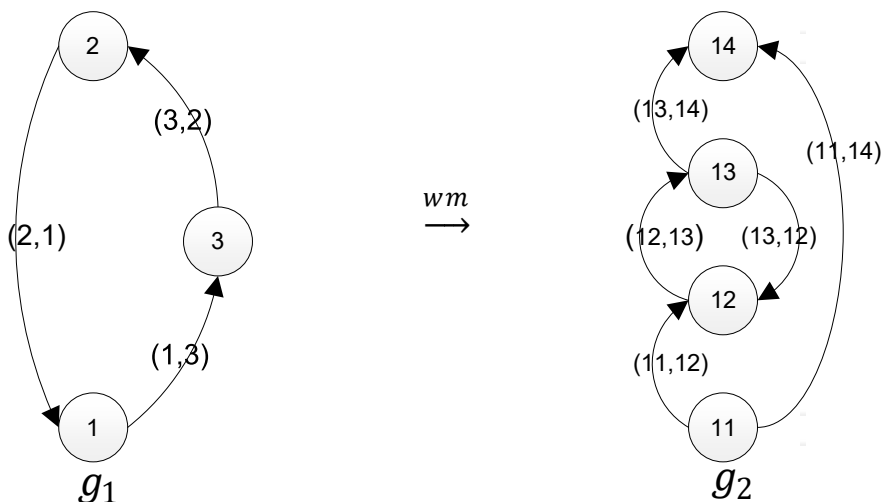


Figure 3: A walk morphism example

The weighting of a path is defined with the sum of the weights of the traversed vertices or edges. The following definition represents the weight of a path by using the vertex weights [14]. The function  $ct$  is a mapping function from vertices in a path to a set of labels  $A$  (see Definition 3).

Definition 15

$$ct(p(v_1, v_n)) := \sum_{i=1}^n ct(v_i)$$

$$p(v_1, v_n) = (v_1, \dots, v_i \dots, v_n)$$

$$ct := \{v_1, \dots, v_i \dots, v_n\} \rightarrow A$$

The weighting of a walk is defined the same as the weighting of a path, where the weights of every traversed vertices or edges are summed. Definition 16 shows the weight of a walk by using the vertex weights.

Definition 16

$$ct(w(v_1, v_n)) := \sum_{i=1}^n ct(v_i)$$

$$w(v_1, v_n) = (v_1, \dots, v_i \dots, v_n)$$

$$ct := \{v_1, \dots, v_i \dots, v_n\} \rightarrow A$$

### 2.1.4 Graph search

The graph search or traversal can be used to compute various properties of graphs, like reachability. This imitates a walk or path in the graph by following vertices. Some graph search algorithms can explore multiple paths in parallel at the same time, whereby some are sequential and search only one path at one time. The depth-first search (DFS) and breadth-first search (BFS) are two graph search algorithms that can be applied to solve a variety of graph search problems, such as finding all of the paths between two given vertices, or finding the all paths from one given vertex to all other vertices in a given graph, finding all of the shortest path from a given vertex to another, etc. DFS and BFS can be applied to directed and undirected graphs.

The DFS in a directed graph is described as following [19]:

Algorithm 1:

Depth-first search from a given vertex to another given vertex in a directed graph

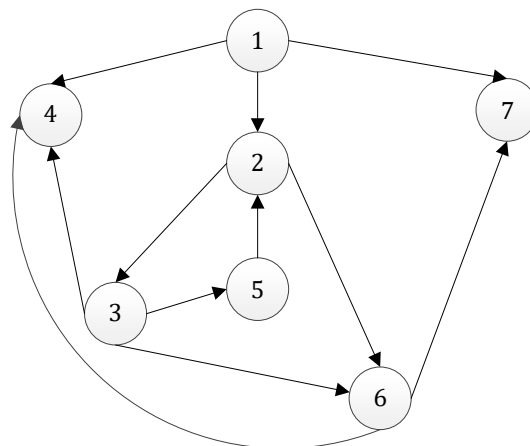
1. Determine the vertices where the search begins and finishes. Mark beginning vertex as visited.
2. Expand all of the following vertices of the start vertex and save them in a stack.
3. Dequeue the top vertex in the stack and examine it.
  - a. If the target vertex is found, cancel the search and deliver the result.
  - b. Otherwise enqueue the following vertices that have not been discovered in the stack, and mark this vertex as visited.
4. If the stack is empty, return "no result".
5. If the stack is not empty, repeat from step 2.

The BFS in a directed graph is described below [19]:

Algorithm 2:

Breadth-first search from a given vertex to another given vertex in a directed graph

1. Determine the vertices where the search begins and finishes. Mark the beginning vertex as visited and save it in a queue.
2. Dequeue a vertex from start of the queue and examine it.
  - a. If the target vertex is found, cancel the search and deliver the result.
  - b. Otherwise enqueue the following vertices that have not been discovered to the end of the queue, and mark this vertex as visited.
3. If the stack is empty, return "no result".
4. If the stack is not empty, repeat from step 2.



DFS: (1,2,3,4,5,6,7)

BFS: (1,4,2,7,3,5,6)

Figure 4: An example of DFS and BFS in a directed graph



Figure 4 shows an example of DFS and BFS traversing in a directed graph. The given initial vertex is vertex 1. The traversing result using DFS is a scheduling of vertices in (1,2,3,4,5,6,7), and using BFS is a scheduling of vertices in (1,4,2,7,3,5,6).

## 2.2 Optimization problem

The optimization problem in mathematics or computer science means problems finding the best or sub-best solutions from all feasible solutions.

“Optimization is a mathematical discipline that concerns the finding of the extreme (minima and maxima) of numbers, functions, or systems. The great ancient philosophers and mathematicians created its foundations by defining the optimum (as an extreme, maximum, or minimum) over several fundamental domains such as numbers, geometrical shapes optics, physics, astronomy, the quality of human life and state government, and several others.” [20]

The origin of the definition of the optimization problem is undoubtedly found in Greece from 569 BC to 475 BC. Since then to now, many understandings, methods, and theories have been developed, like linear programming, non-linear programming, dynamic programming, stochastic approximation, direct search methods, evolutionary programming, differential evolution, etc. [20]. The local optimal problem is defined as the optimization problem, where the solutions are finite [21]. A general constrained optimization problem is defined in the work of Tasgetiren and Suganthan [22] as follows.

$P$  is a constrained optimization problem. The function  $f(x)$  is the objective function. There are two constraints: the inequality constraints are represented with function  $r_i(x) \leq 0$ , and the equality constraints are represented with function  $h_j(x) = 0$ .

Definition 17

$$\begin{aligned}
 &P := \min f(x) \\
 \text{subject to} & \\
 &r_i(x) \leq 0, \quad \text{for } i = 1, \dots, q \\
 &h_j(x) = 0, \quad \text{for } j = q + 1, \dots, m \\
 &f(x) := R^n \rightarrow R \\
 &r_i(x) \leq 0 \\
 &h_j(x) = 0
 \end{aligned}$$

### The shortest path problem

The shortest path problem in graph theory is defined as the problem of finding a path between two vertices such that the sum of the “labels” of its traversed vertices is minimized. This is a special case of the optimization problem and it can be defined with the optimization problem  $P_{sp}$  [23], [24].

The paths  $Paths_{1-j}$  is a set of paths including all paths from  $v_1$  to  $v_j$  in a graph. The function  $ct$  maps this set of paths to a set of labels  $B$ . The shortest path is defined as the path with the minimized label.

Definition 18

$$P_{sp} := \min\{ct(p_i)\}$$

$$ct := \{p_i\} \rightarrow B$$

$$\forall p_i \in Paths_{1-j}$$

## 2.3 Systems development models

System development models provide structure portions for different development phases to organize the development processes. Thereby, these set of phases are assigned to corresponding methods and techniques of the organization, and chained with workflow or dataflow together.

### Waterfall model

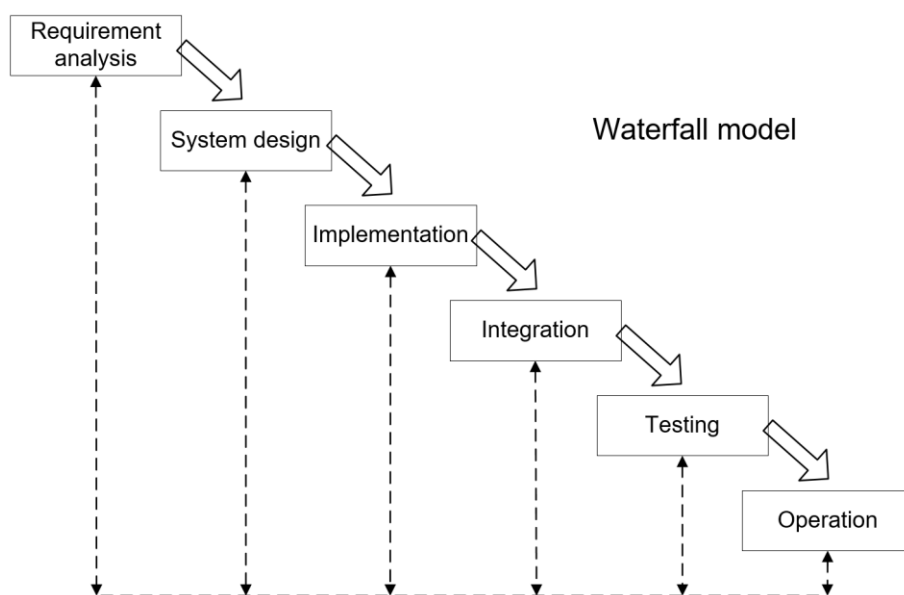


Figure 5: Waterfall model

The waterfall model is a development model in which the phases are consecutively traversed in line during system development. A new phase begins when the previous one has been completed. The chained phases are represented in steps [25]. Figure 5 shows a waterfall model with an iteration, in which the blocks represents the development phases and the thick arrows show the execution sequence of the phases; for instance, after the execution of the requirement analysis phase the phase of system design begins. The slim arrows show the feedback paths from every phase to its preceding phases. If errors are detected at any phase, these feedback paths allow this phase to be reworked, in which errors are committed to correct the errors.

Phases in waterfall model:

- Requirement analysis: This phase is also called requirements engineering or requirements specification. The customer's stakeholders and needs are identified and documented as requirements. The documentation of the requirements is generally made in a standard form that is easy to analyse and can be coordinated with the customer's stakeholders and needs.
- System design: This phase is also referred to as design or architectural design. The large systems are divided into subsystems. The requirements are specified according to the system architecture. The system basics and their relationships are identified and documented.
- Implementation: The systems are implemented with hardware and software according to the system design, which is documented in the previous phase.
- Integration: The implementation units or subsystems will be integrated together if the system has multiple subsystems.
- Testing: The interaction of the subsystems and the integrated system will be tested. It checks whether the implementation has correct system specification.
- Operation (or Installation): The system will be installed and taken into operation. This operation often involves maintaining the system and correcting errors in the running system.

In practice, the phases are often performed in parallel and iterative, but these are very costly. The phases are chained together and the impacts from one phase can be delivered to its following phases. For example, the changes of requirements have an impact on the architecture or structure of the system design, due to high costs of system implementation.

### **Phased implementation model**

In order to reduce the risks during the replacement of an old system, the new system can be implemented in a phased manner. The old system does not need to stop completely; rather the new system is gradually replacing parts of the old system. In comparison with a parallel running model, the phased implementation does not take a lot of extra time and high costs

for the parallel running of a new and old system at the same time. In comparison with a direct changeover model, it keeps a part of the old system to act as a back-up if anything goes wrong during the phased implementation. However, sometimes a direct changeover is the only way to implement a new system.

### **Prototyping model**

A new system is first trailed in one part of a new system on an intermediary trial system. Once the intermediary trial system is running successfully, the new one can be introduced to all of the system. The advantages are apparent: in the intermediary trial system the users can get used to the new system and the new features can also be fully trailed. The existing interfaces and modules on the intermediary trial system provide an easier configuration and integration for system evolution. If something goes wrong with the now features, this only affects part of system, and prompts the need to go back to reengineer the features. Another disadvantage is the high costs and extra time [26].

## **2.4 Model-driven development**

Today, model-driven development is generally accepted as an important method enabler to cope with complex system development. It leverages graphical models and components so that users can visually construct complex system architecture.

### **2.4.1 System modeling languages**

The system modeling language is a modeling language that can be used to express systems in a structure that is defined by a consistent set of rules. An example of system modeling language is unified modeling language (UML) [27] [28].

#### **2.4.1.1 Process-oriented modeling**

The process-oriented modeling is defined to help to analyse and design business or production processes. The main goal is to identify the strategic processes according to the system development requirements. Usually process-oriented modeling is described with formal graphical and textual representations. A process-oriented model comprises process model elements and flows and it understood to mean a distinct arrangement of model elements that are related with flows to one another.

- **Process model element:** A process model element is defined as an agent of a set of objects and it comprises several activities. Every process model element has the defined input and output, which can be machinery materials, product, manpower, energy, signal or information.
- **Flow:** The process model elements are chained together with the flows, which express the connections between processes and can be categorized into different types [29].

Value stream mapping (VSM) is a process-oriented modeling method for analysing the current status and designing a targeted status and it is described with notation modeling language. VSM is a central instrument of lean management, which deals with the visualization of value streams in particular. The main field of application of VSM is the flow production with the original system and low variations in the automotive industry. The central modeling element of VSM is the graphic notation for mapping process and flows of material and information. The notation is defined to model the central production control and various Kanban species [30] [31].

The integrated modeling of material and information flow is an important advantage of VSM. At the same time, the status changes can be very well represented. In this thesis, VSM is used to model the existing status and targeted status of a LL-CPS and it determines the evolution requirements during its managed evolution. More details about this modeling method will be introduced in section 4.1.2. Figure 6 shows an example of a VSM model [32].

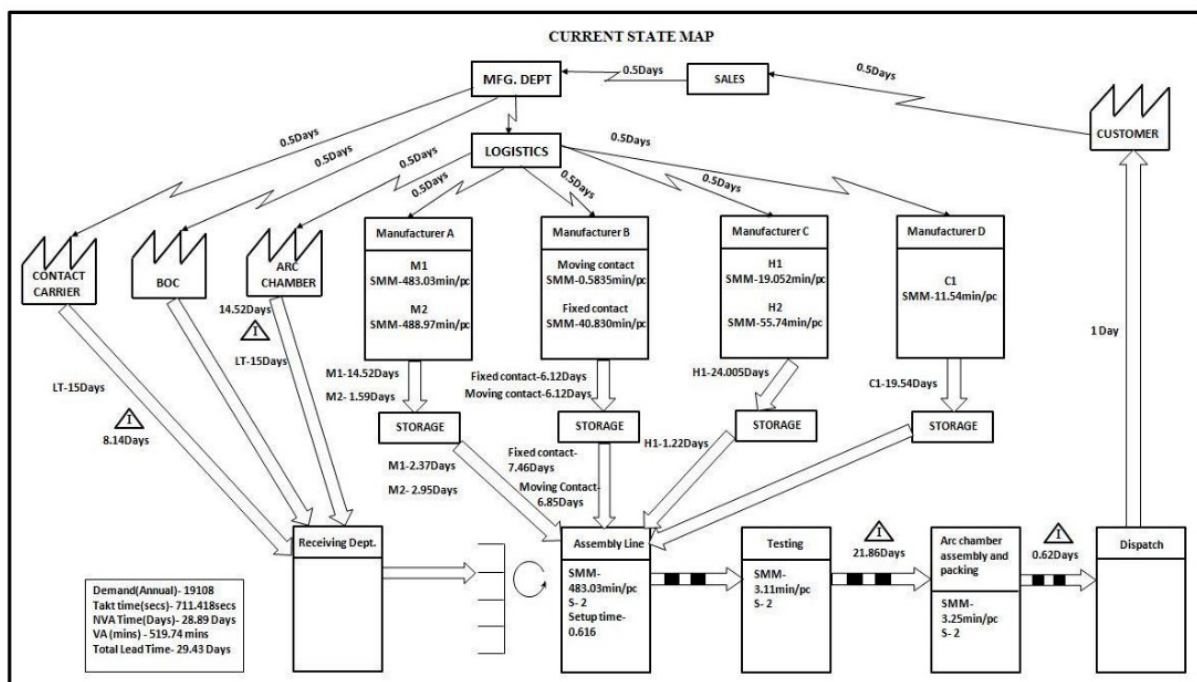


Figure 6: An example of a VSM model

### 2.4.1.2 Component-oriented modeling

The architecture modeling used for the system development, is resorted to the architectural principles and design alternatives. Taking into account the complexity of system decomposition in vertical, component-oriented modeling can reduce complexity by simplifying and strict interface specifications by assembling strongly encapsulated modeling entities, which are called components. Reuse of components can be maximized by finding the

guarantees on implementation of the given components. These component models are also called models [33] [34]. The Object Management Group (OMG) announced the systems modeling language (OMG SysML) to support the specification, analysis, design, verification and validation of complex systems. Two important component-oriented modeling languages in OMG SysML are introduced in the following.

- **Block definition diagram (BDD):** It provides a black box representation of a system. The hierarchies between components in a system are represented by main block and its composite blocks. A BDD is a graphical modeling language of OMG SysML, in which the blocks represent the functions or components and are connected by lines describing the relationships between the blocks. This modeling language is heavily used in engineering for hardware design, software design, electronic design, etc. Figure 7 shows an example of a block definition diagram [35]. In this example, the main block represents a RobotDomain and is comprises the composite blocks: Driver, Robot, PowerSource, Platform and Load.

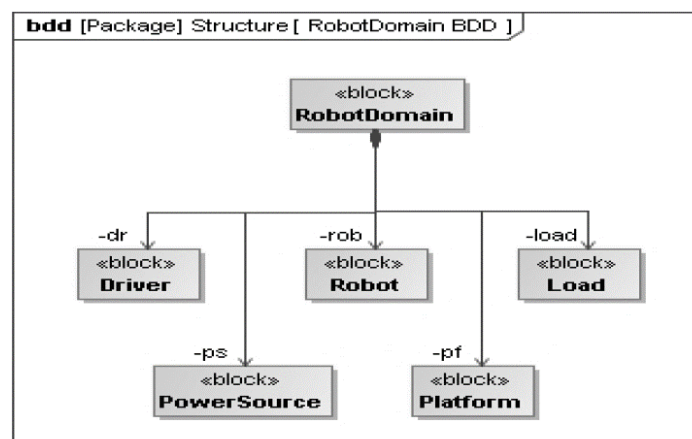


Figure 7: An example of a block definition diagram for a robot domain

- **Internal block diagram (IBD):** The IBD provides the internal view of a system block, and it is usually instantiated from the block definition diagram to represent the assembly of all blocks within the main block. These composite blocks are assembled through ports/interfaces and connectors. The ports/interfaces of the main block are all associated with ports/interfaces of the internal blocks via connectors. Figure 8 illustrates an example of the RobotDomain block with an IBD [35].

In this thesis, VSM and IBD are used to model the managed evolution of LL-CPSs. More details and examples about these two modeling methods will be introduced in chapters 3 and 4.

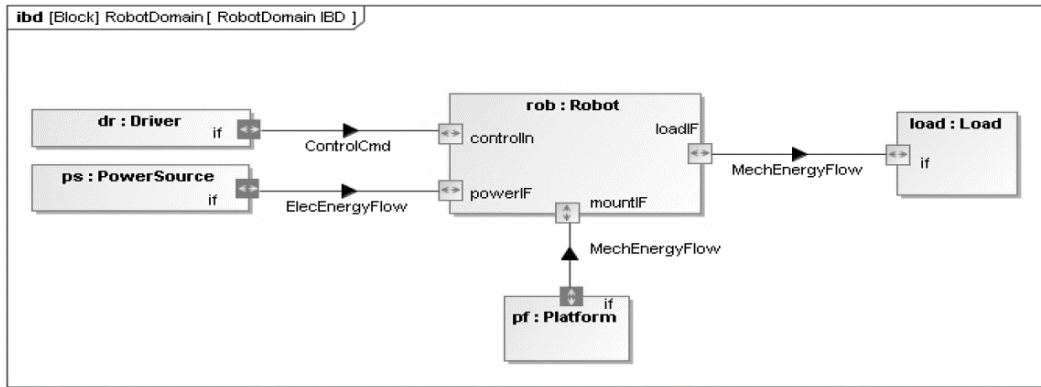


Figure 8: An example of an internal block diagram for the RobotDomain block

### 2.4.2 Metamodeling

The object management group has defined a standard four-layer metamodeling architecture, which is called the meta object facility (MOF). It provides a standard description for the hierarchy of modeling in four layers.

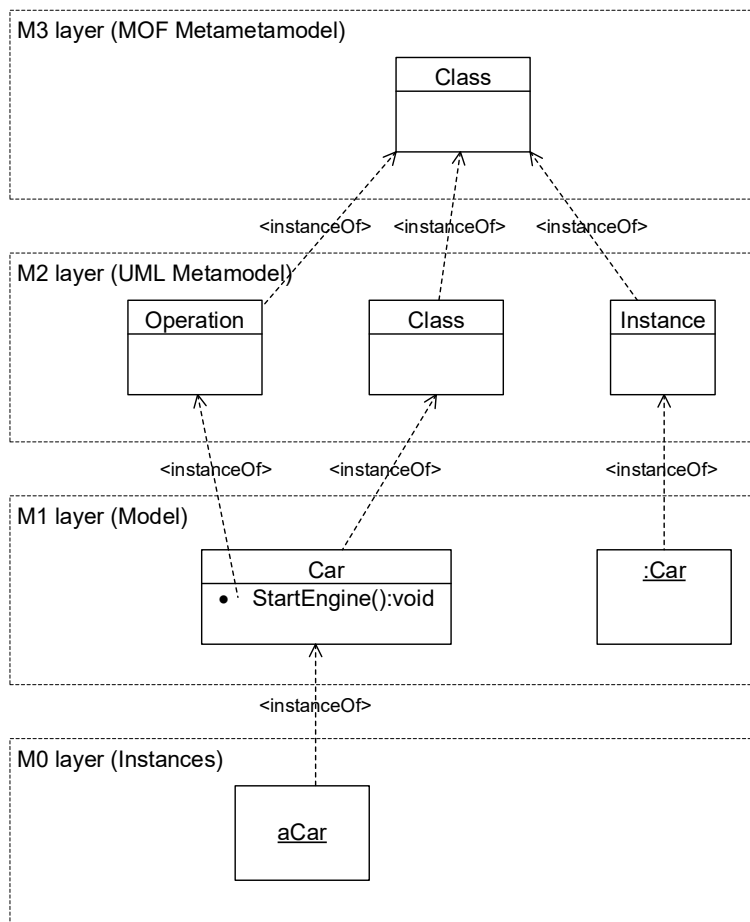


Figure 9: An example of OMG's four-layer metamodel architecture

The first layer provides an instances layer named the M0 layer. At the M0 layer, there is the running system in which the real instances exist.

The M1 layer is model layer and it contains models, for example, a UML model of a real car. Accordingly, the real car is an instance of car model. At the M1 layer, all categorizations or classifications of instances at the M0 layer are represented and designed.

The M2 layer contains the model of the model of an instance, which is also called a metamodel. A metamodel is described as an abstractive syntax to determine the structure and meaning of a model at the M1 layer.

The model at the M3 layer is the model of model of model of an instance. Indeed, this layer is the top layer of the four-layer modeling architecture [36], [37].

Figure 9 shows an example of the OMG defined four-layer metamodel architecture from UML [38]. In this example, a real instance of car exists at the M0 layer. At the M1 layer, this real instance is modelled with a car model using UML. The object “: Car” at the M1 layer represents a car object and is abstracted to a instance at the M2 layer. The car model is abstracted to Operation and Class at the UML metamodel layer. They are the instances of the Metametamodel Class at the M3 layer.

### 2.4.3 Mathematical description of models

For a formal schematic representation, any network structure can be mathematically modeled as a graph [8]. A model  $m$ , which represents **an integrated system in a network structure**, is formed with a directed connected graph structure (see Definition 1).

Definition 19

$$m := (ME, A, beg, end)$$

$$beg := A \rightarrow ME$$

$$end := A \rightarrow ME$$

$$A := ME \times ME$$

$$m \in G$$

$$\{m_i\} = M \subset G$$

$$\forall i = 1, \dots, n \quad n \in \mathbb{N}$$

A model comprises a set of model elements  $ME$  and a set of relation elements  $A$ . The functions  $beg$  and  $end$  combine all model and relation elements together to an integrated system. Every model has an identifier. The ID of a relation element is represented with its



beginning model element and target model element. The model elements and relation elements can be continually specified by different types, processes or functions. These characters of elements are understood as the description information, which can be formed with a key-value structure. A set of models  $\{m_i\}$  is represented by  $M$ .

Another representation for a model  $m$  is using a set of model elements.

Definition 20

$$m := \{ME\}$$

## 2.4.4 Model transformation

The model transformation is a basic method of model-driven architecture (MDA) [39]. There are two important model transformations: bidirectional model transformation and unidirectional model transformation.

### 2.4.4.1 Bidirectional model transformation

A bidirectional model transformation is described with a one-to-one mapping relationship between one model on a modeling domain and a model on another modeling domain. This model transformation can be used to keep the model consistency and it allows the transformation between a concrete model and an abstract model. Stevens [40] represents the bijection as a special case of bidirectional transformation, where the source elements contain the same information as the target elements. In Stevens' example, a bidirectional model transformation takes place between UML activity diagrams and Petri net models.

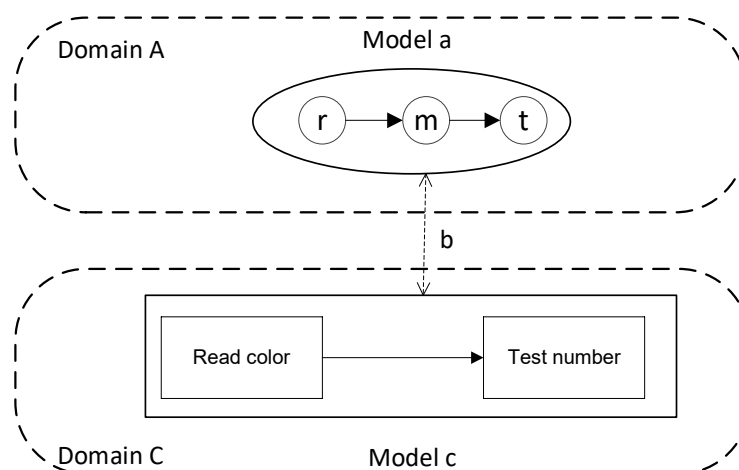


Figure 10: An example of bidirectional model transformation

A bidirectional model transformation is defined with a mapping relationship  $b$ , where  $A$  is a domain including a set of models, and  $C$  is another domain including a set of models.

Definition 21

$$b := A \leftrightarrow C$$

In Figure 10, model  $a$  in domain  $A$  has a bidirectional transformed model  $c$  in domain  $C$ . This transformation can be understood as the projection of one model in a domain into another domain.

#### 2.4.4.2 Unidirectional model transformation

The unidirectional model transformation is a refinement mapping and is defined as a mapping relationship of models from one modeling level to another. Czarnecki and Helsén [41] proposed a possible taxonomy for the classification of several existing and proposed model transformation approaches, whereby most of these approaches are used for unidirectional model transformation. A unidirectional model transformation is formalized as a mapping relationship  $u$ , where  $A$  is its domain and  $B$  is its codomain.

Definition 22

$$u := A \rightarrow B$$

Figure 11 shows an example of unidirectional model transformation. Model  $a$  in domain  $A$  is transformed to model  $b$  in domain  $B$ . In addition, a bidirectional model transformation can be achieved using two opposite unidirectional one-to-one model transformations [41].

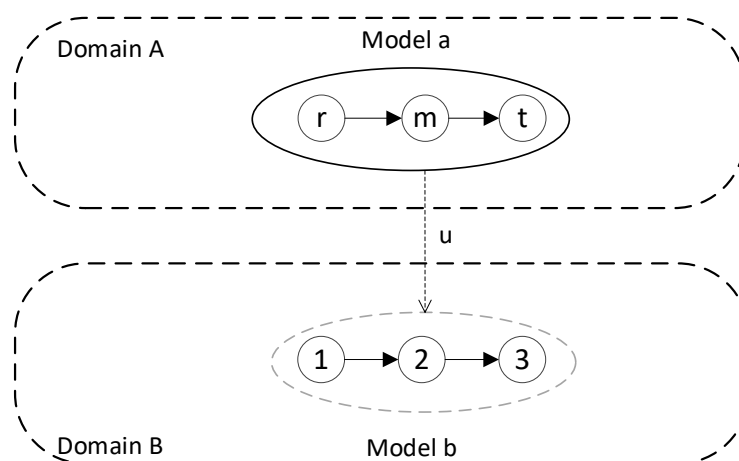


Figure 11: Examples of unidirectional model transformation

The bidirectional and unidirectional model transformations are in essence the transformations for model elements and relation elements from one model to another model, which will be introduced in chapter 4.

### 2.4.5 Model mapping

A unidirectional mapping function  $q$  represents a mapping relationship from a model to another model, where  $A$  is its domain comprising a set of models and  $D$  is its codomain comprising a set of models.

Definition 23

$$q := A \rightarrow D$$

Figure 12 shows an example of model mapping. The mapping function  $q$  maps the model  $a'$  in domain  $A$  to model  $d$  in domain  $D$ .

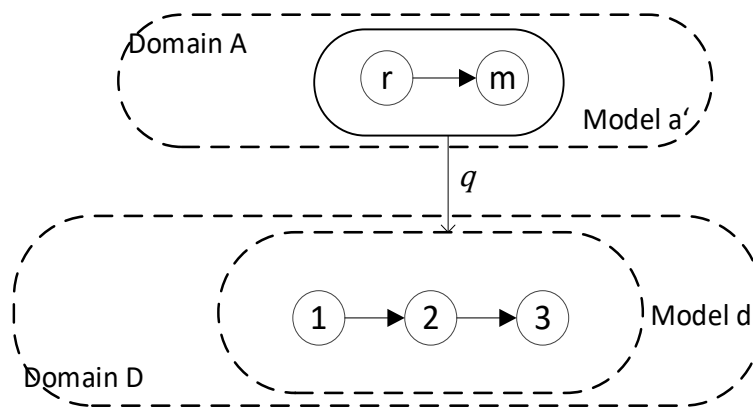


Figure 12: An example of model mapping

The model mapping is essentially the mappings for model elements and relation elements from one model to another, which will be introduced in chapter 4.

### 2.4.6 Operation of models

The operation of models is used to represent the relationship between models or to create new models from the initial ones. Models  $m_1$  and  $m_2$  are two models in  $M$  (see Definition 19). The operator " $\leq$ " represents that model  $m_2$  is a sub-model in model  $m_1$ .

Definition 24

$$\begin{array}{l}
 \text{if} \\
 m_1 \in M \text{ and } m_2 \in M \\
 ME_2 \subseteq ME_1 \quad ME_2 \neq \emptyset \\
 A_2 \subseteq A_1 \\
 beg_2 := beg_1 |_{A_2 \rightarrow ME_2} \\
 end_2 := end_1 |_{A_2 \rightarrow ME_2} \\
 \text{then} \\
 m_2 \leq m_1
 \end{array}$$

The models union of two models  $m_1$  and  $m_2$  is represents with an operator " $\oplus$ ".The initial models are two sub-models of the new model.

Definition 25

$$\begin{array}{l}
 m_3 = m_1 \oplus m_2 \\
 ME_3 = ME_1 \cup ME_2 \\
 A_3 = A_1 \cup A_2 \\
 beg_3 := beg |_{A_1 \cup A_2 \rightarrow ME_1 \cup ME_2} \\
 end_3 := end |_{A_1 \cup A_2 \rightarrow ME_1 \cup ME_2} \\
 m_1 \leq m_3 \\
 m_2 \leq m_3
 \end{array}$$

The models difference for one graph comparing with another is represented by an operator " $\ominus$ ". In this example, model  $m_4$  is a sub-model of model  $m_1$ .

Definition 26

$$\begin{array}{l}
 m_4 = m_1 \ominus m_2 \\
 ME_4 = ME_1 \setminus ME_2 \\
 A_4 = A_1 \setminus A_2 \\
 beg_4 := beg_1 |_{A_4 \rightarrow ME_4} \\
 end_4 := end_1 |_{A_4 \rightarrow ME_4} \\
 m_4 \leq m_1
 \end{array}$$

## 3 Problem Statement and Analysis with Example

### Content

---

- 3.1 A sample of LL-CPS: A conveyor system with ASRS
    - 3.1.1 Process-oriented description
      - 3.1.1.1 Process-oriented structure
      - 3.1.1.2 Processes description
    - 3.1.2 Component-oriented description
      - 3.1.2.1 Interfaces specification
      - 3.1.2.2 System decomposition
  - 3.2 Managed evolution scenario of LL-CPS
    - 3.2.1 Problems of the ongoing LL-CPS
    - 3.2.2 The targeted status of this LL-CPS
  - 3.3 State of the art and existing approaches for managed evolution of LL-CPSs
    - 3.3.1 Cyber physical system modeling
    - 3.3.2 Formal modeling of system evolution
    - 3.3.3 Modeling and optimizing the costs of reconstruction
  - 3.4 Research questions of this thesis
- 

The aim of this chapter is to ascertain the central problems during managed evolution of LL-CPS. First a conveyor system in an auto manufactory factory as the application example will be introduced in section 3.1. This system is a classical LL-CPS.

The process-oriented modeling method VSM introduced in the previous chapter models the system structure and working processes of this conveyor system from the perspective of process design. On the other hand, the component-oriented modeling method IBD specifies the system hierarchy and interface specification of this conveyor system from the perspective of implementation plan. The existing status of this conveyor system is modeled by using of both modeling methods in section 3.1.1 and 3.1.2.

In order to analyse the problems during the managed evolution of a LL-CPS, the targeted status of this conveyor system is clearly defined using a VSM model in section 3.2. Section 3.3 provides an overview of the state of the art regarding the existing approaches supporting the development of the LL-CPS from the existing status to targeted status. Observing the problem analysis and state of the art of managed evolution of LL-CPS, the final section of this chapter will provide a summary of all the main research questions that will be particularly discussed and solved in the following chapters.

### 3.1 A sample of LL-CPS: A conveyor system with ASRS

In an automobile factory, a conveyor system with an automated storage and retrieval system warehouse (ASRS) is understood as a LL-CPS that comprises mechanics, sensors, hardware and software. This system chains the different manufacturing processes together to an integrated production system, e.g. welding, injection modeling, sub-assembly/assembly and painting and dry system. For the intralogistics in the factory, this conveyor system is the key part and has a long life cycle. Figure 13 shows the conveyor system with ASRS as the core part in an automobile factory.

This conveyor system with ASRS is a typical sample of LL-CPS, whereby people, machines, and products exchange information with each other to accomplish the conveyor tasks together.

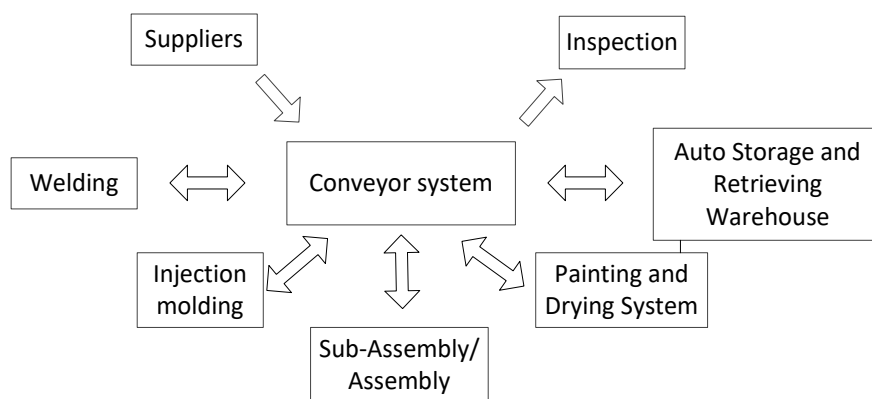


Figure 13: A conveyor system with ASRS in an automobile factory

A laboratory model of this conveyor system with ASRS is created as a substitution of the real factory to clearly define this LL-CPS and analyse the problems during its evolution. This laboratory model contains a conveyor system model and an auto storage and retrieving warehouse model. The conveyor system model is located between the warehouse model and the other CPSs, e.g. painting and dry system. The warehouse model is applied as a storage of the wares, before and after they go into the manufacturing processes (See Figure 14). The conveyor system model comprises four conveyor belts, a buffer belt and sensors, etc. The ARAS comprises a warehouse and a gripper robot. The automated parts of this laboratory model are controlled through two industrial programmable logic controllers (PLC) of Siemens. They connect to each other over the Ethernet to ensure the safety and reliability of the connection. The coordinated tasks between the sensors, actors, warehouse, four conveyor belts, and a buffer belt are controlled using these two PLCs. A computer is used as a human-

### Chapter 3 - Problem Statement and Analysis with Example

machine-interface (HMI). More detailed information on the system specification will be introduced in section 3.1.2.

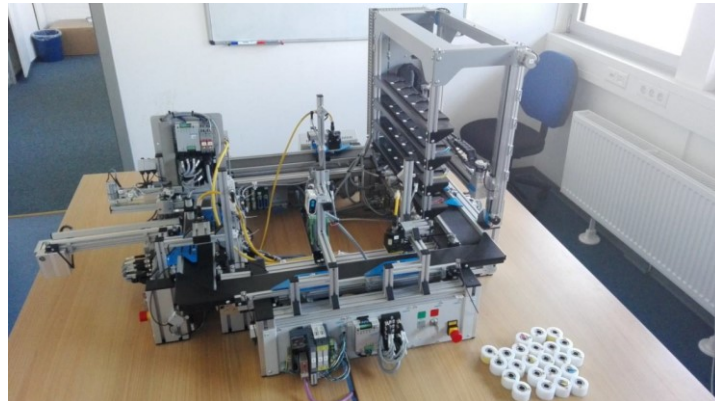


Figure 14: A laboratory model of a Conveyor system with ASRS as a LL-CPS sample

In practice, a LL-CPS is typically described and analysed by using multi-domain and multi-level models, where each model focuses on a fixed set of concerns on the system. This is conducive to system planners and engineers to understand a LL-CPS better and faster from different disciplines. Each modeling domain gives prominence to certain features and focuses on particular attributes.

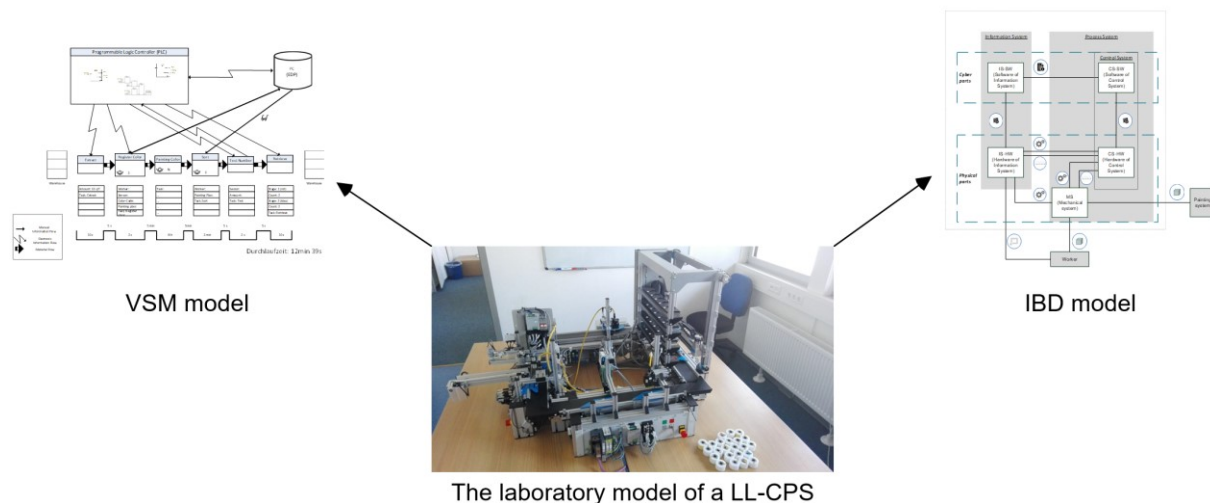


Figure 15: A LL-CPS modeled with VSM and IBD

In Figure 15, there are two modeling methods that describe this LL-CPS with two different modeling methods: the VSM and the IBD.

### 3.1.1 Process-oriented description

#### 3.1.1.1 Process-oriented structure

In this LL-CPS, the wares should be transported with the conveyor system from warehouse to the hall for the painting and dry processes in accordance with the production plan for painting and the dry hall. After the painting and dry processes, the wares will be transported back to the warehouse and wait for the following manufacturing processes. Subsequently, the wares will be stored in a warehouse in correct order. For example, the wares with different types, sizes, materials or paint colors can be divided into different groups and stored in the designated location or floor.

The whole process of this laboratory model is separated into six sub-processes, which are chained step by step from the first/leftmost process to the last/rightmost one. Every sub-process contains the functions and tasks. Figure 16 shows the processes structure of this LL-CPS sample.

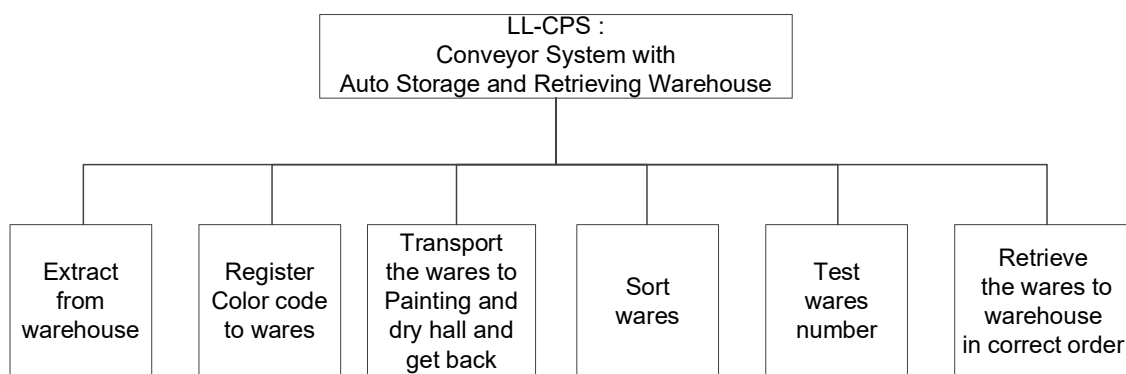


Figure 16: Processes structure of the LL-CPS sample

#### 3.1.1.2 Processes description

Figure 17 shows a VSM model for this LL-CPS, which is defined as the existing status of this LL-CPS. Every sub-process comprises different functions and tasks to accomplish the whole task together.

- Extract:  
The wares will be transported from the warehouse to the conveyor belt with a gripper robot. This is a completely automated process and it is controlled by one PLC. This PLC gives off the signal information, like the wares, amount and locations, to schedule the different functions and tasks in the extract process.



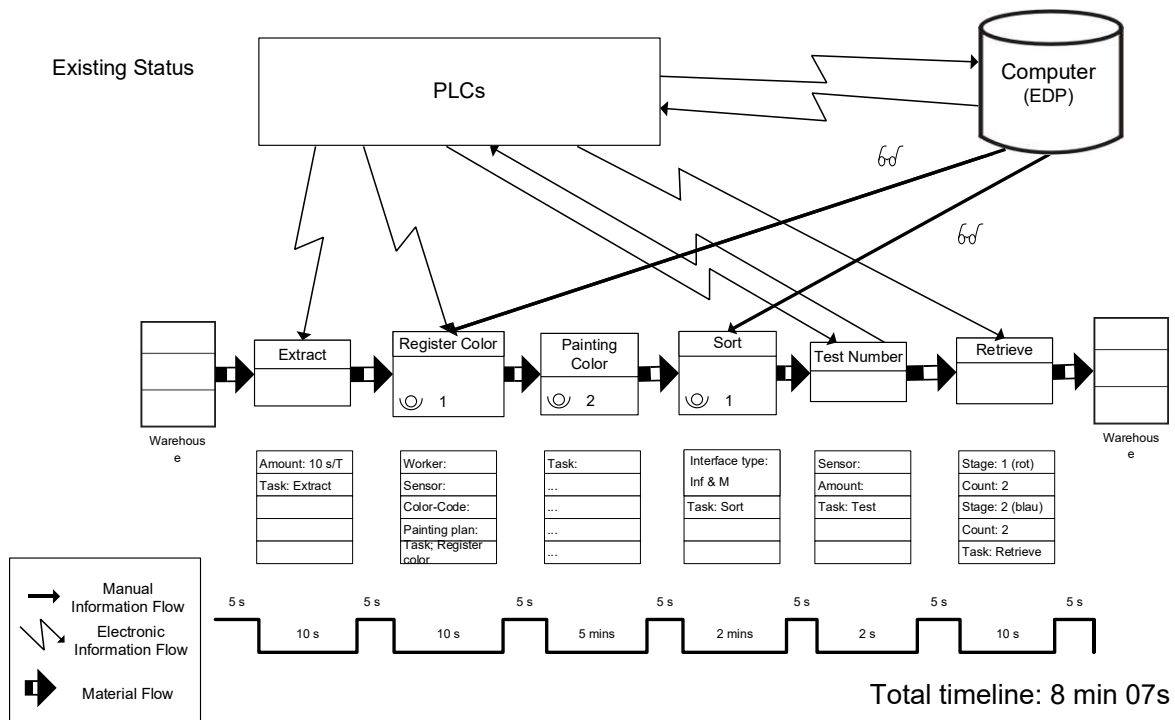


Figure 17: A VSM model for the existing status of the conveyor system with ASRS

- Register color:**

The wares run on the conveyor belt and pass through a RFID read/write sensor. More specifically, when a ware arrives at the RFID read/write sensor, the conveyor belt will be stopped. At the same time, the RFID chip in the ware waits for an input from the worker; for instance, the color information. The conveyor belt will restart once the RFID chip receives correct information and then will be stopped again until the second ware without any information arrives at this RFID read/write sensor. The worker gives information by using a HMI of a computer. This information will be stored in this computer and used for the other automated processes. The wares will continue to be transported to the next conveyor belt after they leave the current belt. The register color process is a semi-automatization process.
- Painting color:**

After the register color process, there is a buffer belt with a pusher, which is like a T-Stab. This buffer belt bridges the conveyor belt to the painting and dry system. Each ware which will be transported to the painting and dry hall must be pushed on the buffer belt by using the pusher. The painting and dry system is another LL-CPS and it is independent from this conveyor system with ASRS. The wares are taken sequentially to the reserved painting machine. The painting processes will not be introduced in this thesis. After the painting and dry process, all painted wares should be transported with the same buffer belt again, but from the painting and dry system side to the conveyor belt side.

- **Sort:**  
As a matter of course, the storage of painted wares must follow certain rules and standards. For instance, according to their types, sizes, materials or paint colors, the wares are able to be divided into different groups and then stored on the corresponding floors in the warehouse. For this reason, a worker must stand by the buffer belt and sort the wares by predefined sort order. For instance, the same color wares are divided into the same group and sorted on the buffer belt. This is a manual task.
- **Test number:**  
This process is an automated process. By using a photoelectric sensor, the printed wares will be counted according to the given number information from the worker in the register color process. It is important to ensure that the actually stored quantity is equal to the number of wares to be painted before. This photoelectric sensor can count the wares number, but not read the color information.
- **Retrieve:**  
Eventually, the wares will be retrieved through the gripper robot and then stored back in a predefined location or floor in the warehouse.

### **3.1.2 Component-oriented description**

The component-oriented modeling description focuses on a decomposition of the system into its system structure, which is a central document for the creation of products. The entire system is decomposed into segments and units and the relationships between these segments and units are identified and represented. Thereby, the component-oriented modeling method can support analysing the behavior in a large-scale system and guide the implementation of the system design.

The component-oriented modeling methods BDD and IBD introduced in section 2.4.1.2 are used to model the laboratory model of the conveyor system with ASRS.

#### **3.1.2.1 Interfaces specification**

The interfaces are first fine specified, which are declared within the system between the segments and units or on the outside of the system. These specified interfaces serve a better understanding of the system structure. Figure 18 shows the symbols of the specified interfaces in this LL-CPS.

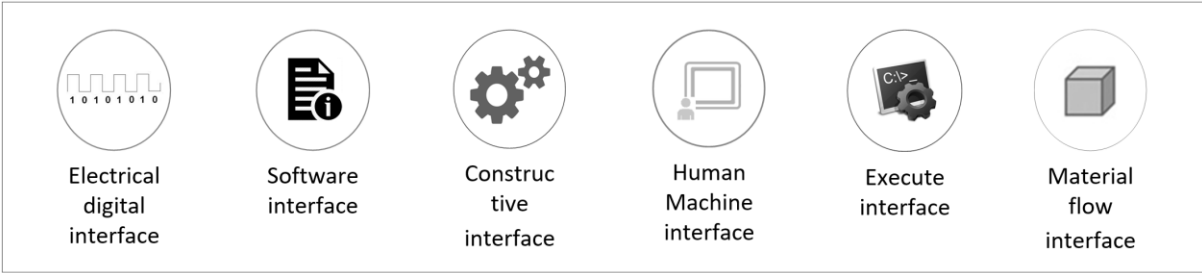


Figure 18: Interfaces specification

- **Electrical signal interface:**  
This interface is used to provide the connection for digital information from one electronic device to another. The digital information can take the instructions of functions or the results of the executed functions; for example a USB interface, PCI express interface or the IEEE 1394.
- **Software interface:**  
The software interfaces make the connection possible between software components to exchange information, functions or methods. In this sprcification, the software interfaces exist on software components or software components assembly.
- **Constructive interface:**  
A constructive interface can be understood as an integration of devices. In this sample, the four conveyor belts are constructed together to execute a corporate transportation.
- **Human machine interface:**  
A human machine interface is a special case of a user interface that connects human to mechanical or information systems. The mechanical or information systems can be the integrations of hardware, software, and mechanical components; for instance, a monitor or an operating panel of PLC or an operating platform of a conveyor belt. The HMI translates data or information from the machine side into human-readable visual representations. In our sample, the worker gives information into the information system by using the HMI of PC and reads the scheduling information from information system with the HMI again.
- **Execute interface:**  
The execute interface is a computer interface that provides connections between the required component and the provided component. In this sample, the computer hardware can provide the execution for the requirements of the connected software.

- Material flow interface:  
The material flow interfaces are defined as the transport channels for material. They are often found between transport equipment or working processes. In our sample, the worker can also be understood as a transport tool to chain the material flow together.

### 3.1.2.2 System decomposition

The decomposition of a LL-CPS is introduced with a hierarchical structure. At the first level, the LL-CPC is decomposed into two parts: a set of information systems (IS) and a set of process systems (PS). The information system is the cyber part in the LL-CPS and the process system is monitoring and controlling the physical processes by using sensors and actuators. These two parts are connected, strongly associate with each other and work independently of one another. They sense and influence the environment of the LL-CPS together (see Figure 19).

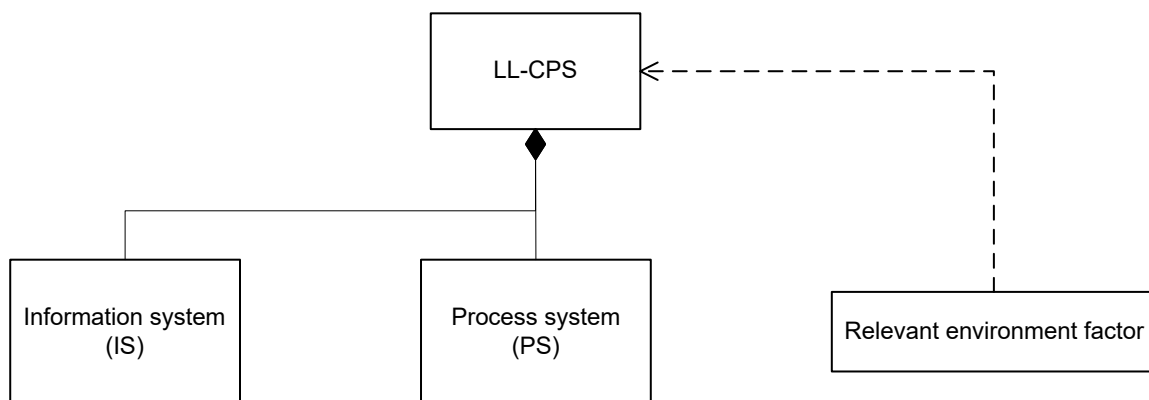


Figure 19: System decomposition of LL-CPS with BDD

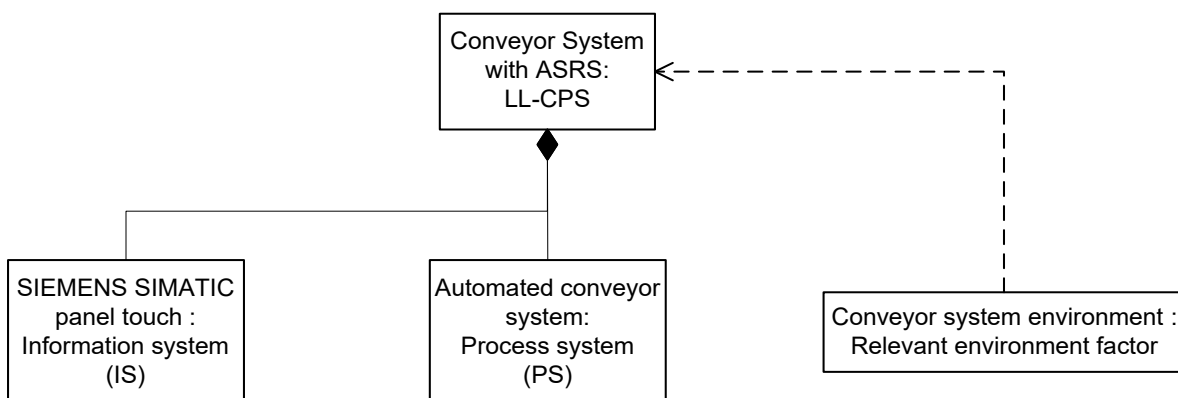


Figure 20: System decomposition of the conveyor system with ASRS with BDD

The Figure 20 shows the decomposition of the conveyor system with ASRS as an example. The conveyor system with ASRS is decomposed into a Siemens Simatic panel touch system as the information system and an automated conveyor system as the process system. This LL-CPS is closely linked with the conveyor system environment and exchanges the information or wares through the system interfaces.

Figure 21 shows the interface specification between the conveyor system with ASRS and its system environment with an IBD. The relevant environment factors comprises two important parts: another CPSs and workers. In this sample, the painting and dry system is another LL-CPS and it is connected with the conveyor system with ASRS using material flow interfaces and connectors. Worker 1 sorts the wares running on the buffer belt in the conveyor system using HMI and material flow interface. Worker 2 reads and gives the color information by using a HMI to this LL-CPS.

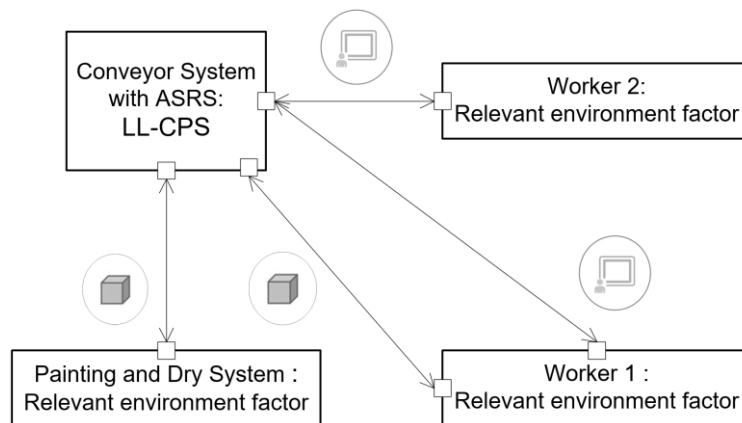


Figure 21: Interface specification between the conveyor system and its system environment

### Description of process system

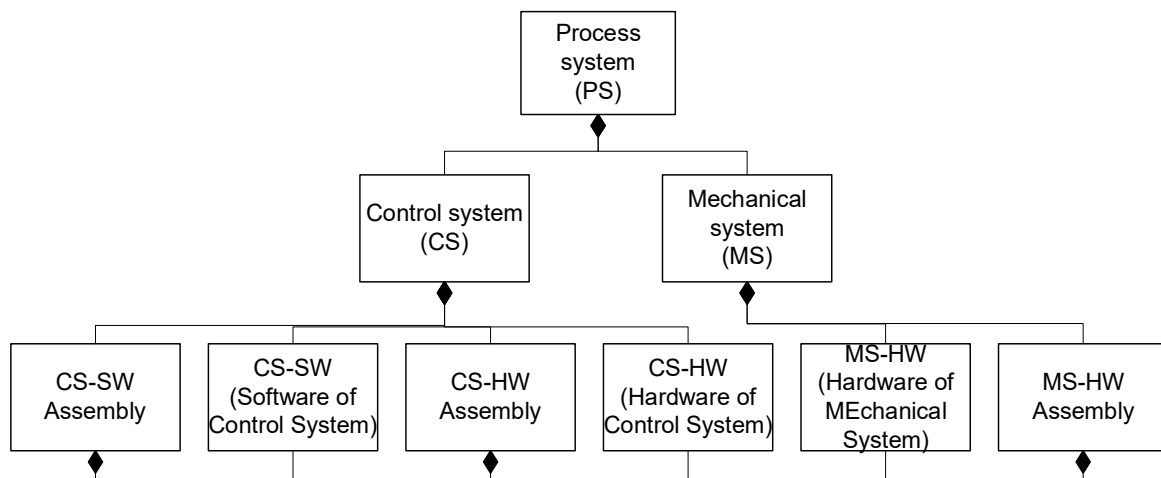


Figure 22: System decomposition of process system with BDD

### Chapter 3 - Problem Statement and Analysis with Example

At the second level of this hierarchical structure, the information system and the process system will be continually decomposed. A PS (see Figure 22) is divided into a set of control systems (CS) and a set of mechanical systems (MS). The control system is continually decomposed into a set of software of control system (CS-SW), a set of software assembly (CS-SW Assembly), a set of hardware of control system (CS-HW) and a set of hardware assembly (CS-HW Assembly). In a process system, the variables and information are shared between components, which is different than in an information system.

With the laboratory model as an example, Figure 23 shows a BDD of the automated conveyor system as the process system in LL-CPS. It comprises a SIEMENS Simatic PCS7 as a control system and the conveyor models as a mechanical system.

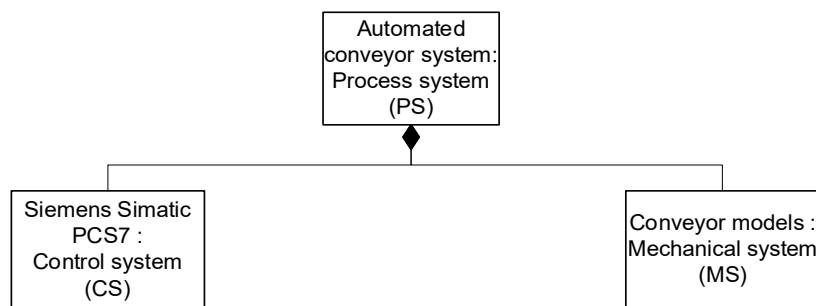


Figure 23: BDD of process system in the sample

The mechanical system in this laboratory model comprises four conveyor belts, a buffer belt and a warehouse, etc.

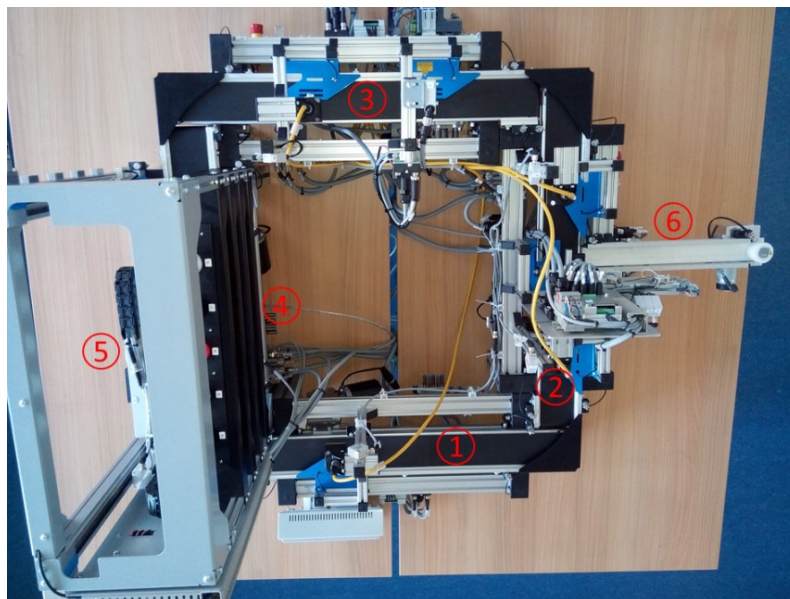


Figure 24: Top view of the laboratory model

Figure 24 shows the top view of the laboratory model, whereby the four conveyor belts are chained in a cycle to transport the wares in circulation (turn counter clockwise in order ①②③④). Above conveyor belt ① there is a RFID read/write sensor, which can be used to read/write production information from/into wares. There is a photoelectric sensor (light barrier) over the conveyor belt ③, which is used for detecting the wares when they get up the conveyor belt ③. The warehouse ⑤ is constructed with conveyor belt ④, and all of the wares will be transported from the warehouse to the conveyor belt ④ with a gripper robot or phase reversal. The buffer belt ⑥ with a pusher bridges the conveyor belt ② to other CPSs.

Figure 25 illustrates the decomposition of the mechanical system in the laboratory model. This mechanical system/mechanics is composed of three MS-HW assemblies: from left to right are the ASRS warehouse, four conveyor belts, and one buffer belt. The ASRS warehouse comprises warehouse and gripper robot. The four conveyor belts comprise belts, belt-motors, R/W sensors and PH (photoelectric) sensors. Moreover, the buffer belt comprises belts, belt-motors, and pushers.

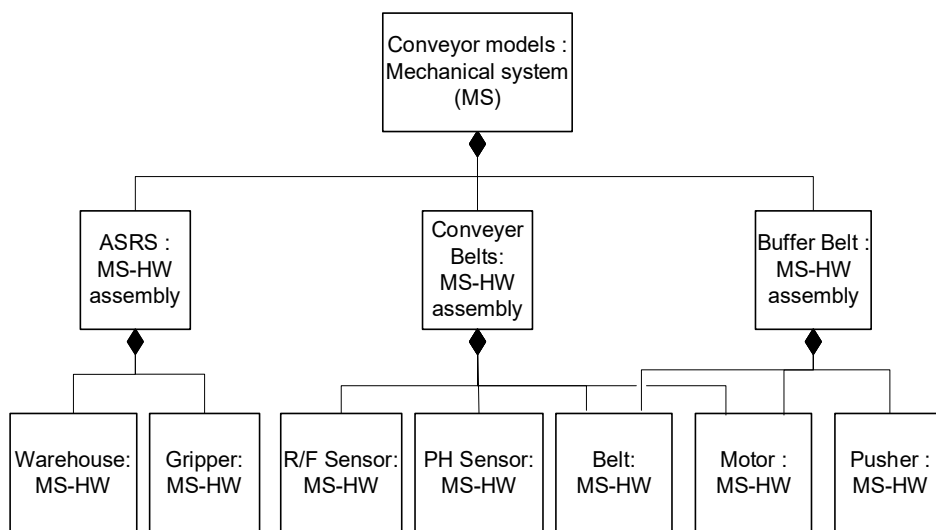


Figure 25: BDD of the mechanical system in the sample

With an internal block diagram of the mechanical system, the combination of interfaces, connectors, and components is represented in Figure 26.

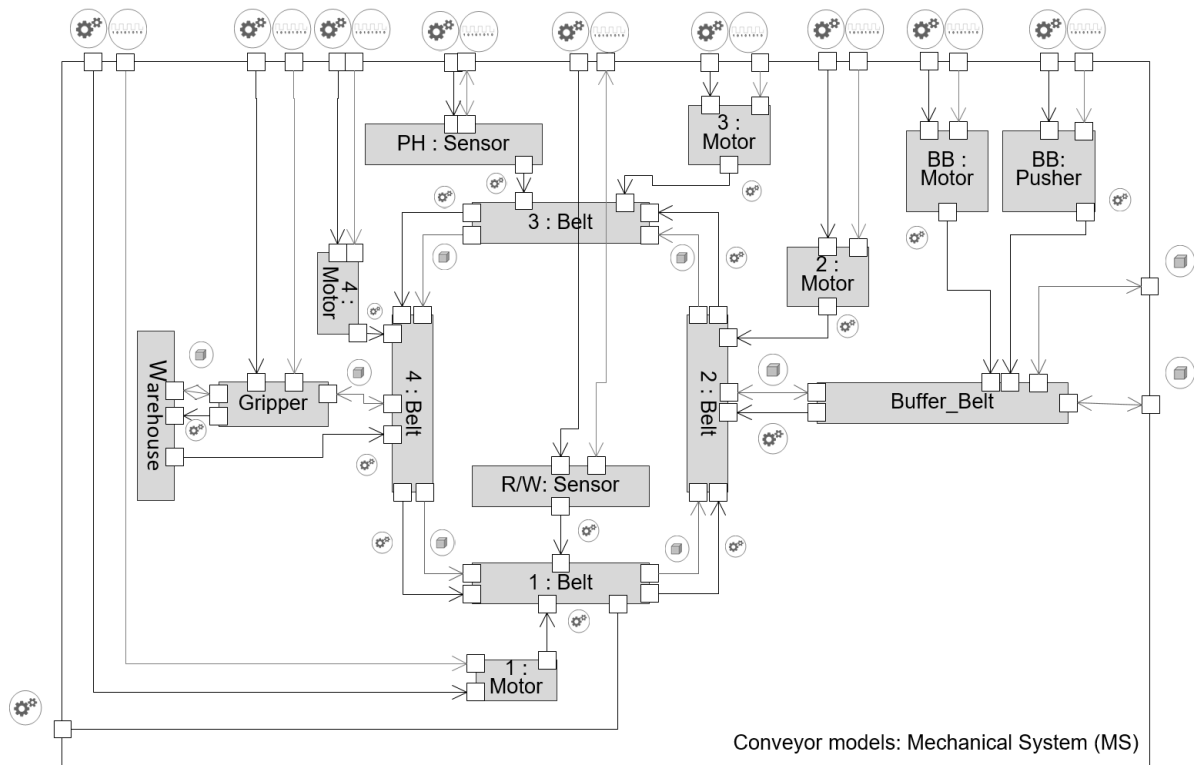


Figure 26: IBD of the mechanical system in the sample

The control system is the other important part in the process system in a LL-CPS. It manages, commands or regulates the behavior of connected mechanical systems to implement some tasks or processes. For sequential and combinational logic, the PLC is very necessary, like a computer numerical control (CNC) and robot controllers (RC). Figure 27 shows a BDD for the decomposition of control system in the laboratory model. This control system comprises different CS-HW assemblies and CS-SW assemblies. The CS-HW assembly PLC hardware platform comprises CS-HWs: from left to right are the data store, timer, program memory, BUS, etc. The CS-SW assembly STEP 7 comprises CS-SWs: FC 10, OB1, etc.

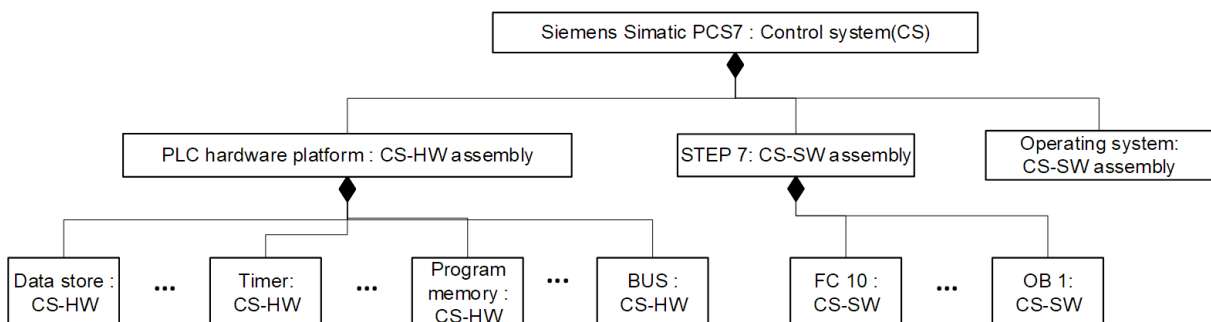


Figure 27: BDD of control system in sample

In this laboratory model, the CS-HWs and CS-HW assemblies are integrated in two PLC hardware platforms. The automation of this conveyor system with ASRS is under the control



Chapter 3 - Problem Statement and Analysis with Example

of these two Siemens Simatic PCS7 300 PLC systems [42]. Figure 28 shows a picture of one integrated PLC hardware platform in the control system of the laboratory model.

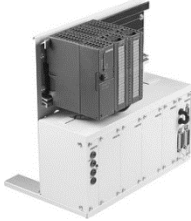


Figure 28: Siemens Simatic PCS7 300: Hardware assembly in control system

The tasks in the programmable control system are not fulfilled only by combination of several hardware components, which need the specific software programs. Figure 29 shows the IEC 61131-3 standards-based programming languages of PLC, which are classified by the language characteristics. The structured text (ST in Siemens S7 called Structured Control Language: SCL) and instruction list (IL) are two graphical languages. The function block diagram (FBD) and ladder diagram/logic (LD) are two classical text languages. The sequential function chart (SFC in Siemens S7 named Graph) is a mixed of the above two [43].

The ST is a high-level language, which is block structured. The function calls and variables are defined by common elements. The syntax of this language elements is very similar to Pascal. ST provides good mathematical operands for program logic. Figure 30 shows an example of an FBD program in the laboratory model.

The IL is a low level language, which is mainly used for the logic connection between control inputs and outputs. The main features of IL are that the operators have only one operand and the syntax of the language is based on assembly language. This language provides a table view that allows a lot of data to be well organized, updated and displayed. On the other side, for large-scale and complex program IL is difficult to read.

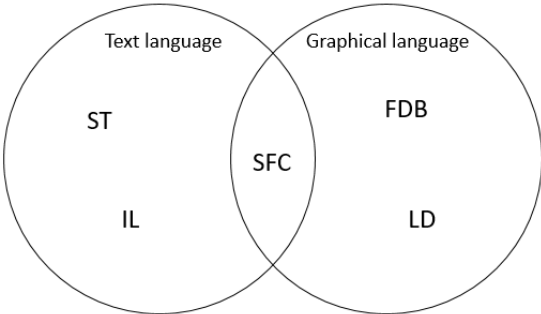


Figure 29: IEC 61131-3 standards-based programming languages of PLC

## Chapter 3 - Problem Statement and Analysis with Example

```
// Ist-Werte
IF Werkstueckerkannt=true THEN
  IF zahlerRot_soll <> zahlerRot_ist THEN
    zahlerRot_ist:= zahlerRot_ist +1;
  ELSIF zahlerGelb_soll <> zahlerGelb_ist THEN
    zahlerGelb_ist:= zahlerGelb_ist+1 ;
  ELSIF zahlerBlau_ist<> zahlerBlau_soll then
    zahlerBlau_ist:= zahlerBlau_ist +1;
  END_IF;

  IF (zahlerRot_soll+zahlerGelb_soll+zahlerBlau_soll) = (zahlerRot_ist+zahlerGelb_ist+zahlerBlau_ist) THEN
    Produktionsziel_erreicht:= true;
  END_IF;
END_IF;
```

Figure 30: A ST program example

The FBD is a graphical language, which can describe the functions between the input variables and output variables. The function is understood as a set of elementary blocks. The connection lines are using to connect the input and output variables. This language is a very good fit for electrical engineers. Figure 31 shows an example of FBD program in the laboratory model.

The LD represents the programs by using a graphical diagram, which is based on circuit diagrams of relay logic hardware. This language has a clear logical relationships that provides an easy structure to read. For a large-scale and complex program, LD is impractical and it has no mathematical operands possible.

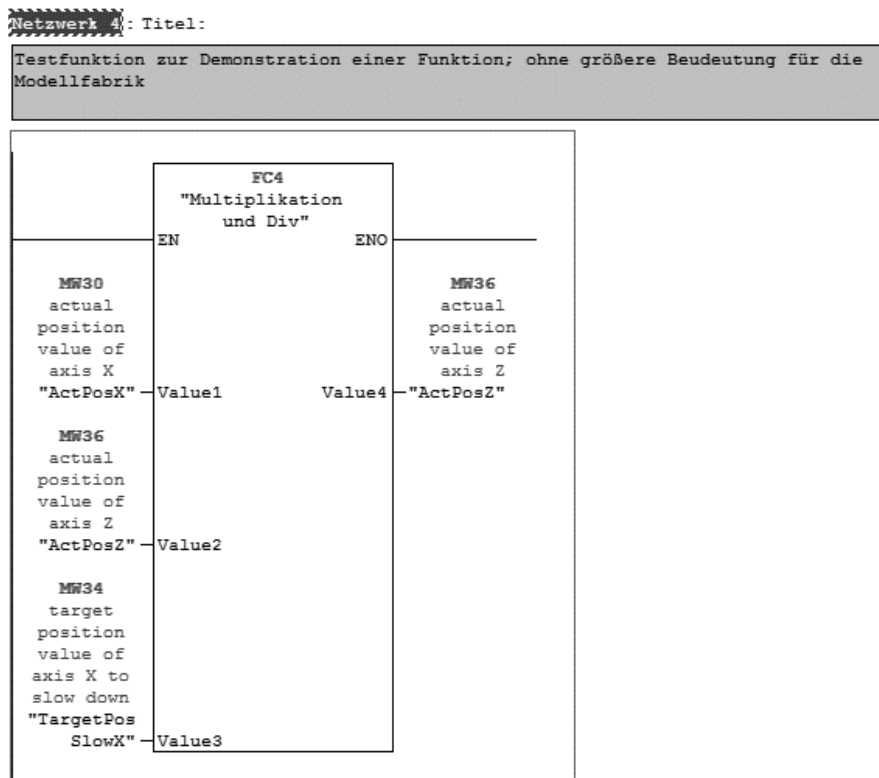


Figure 31: A FBD program example

## Chapter 3 - Problem Statement and Analysis with Example

The SFC is a graphical language, which can be used to program processes that can be split into steps. The control steps are associated with each other by conditions of the handoffs in this flow control. Between the control steps are transitions that are linked to input bits. The SFC is described as a Petri net and it is easy for process design and error analysis. In normal conditions, this programming language needs a strong program capacity.

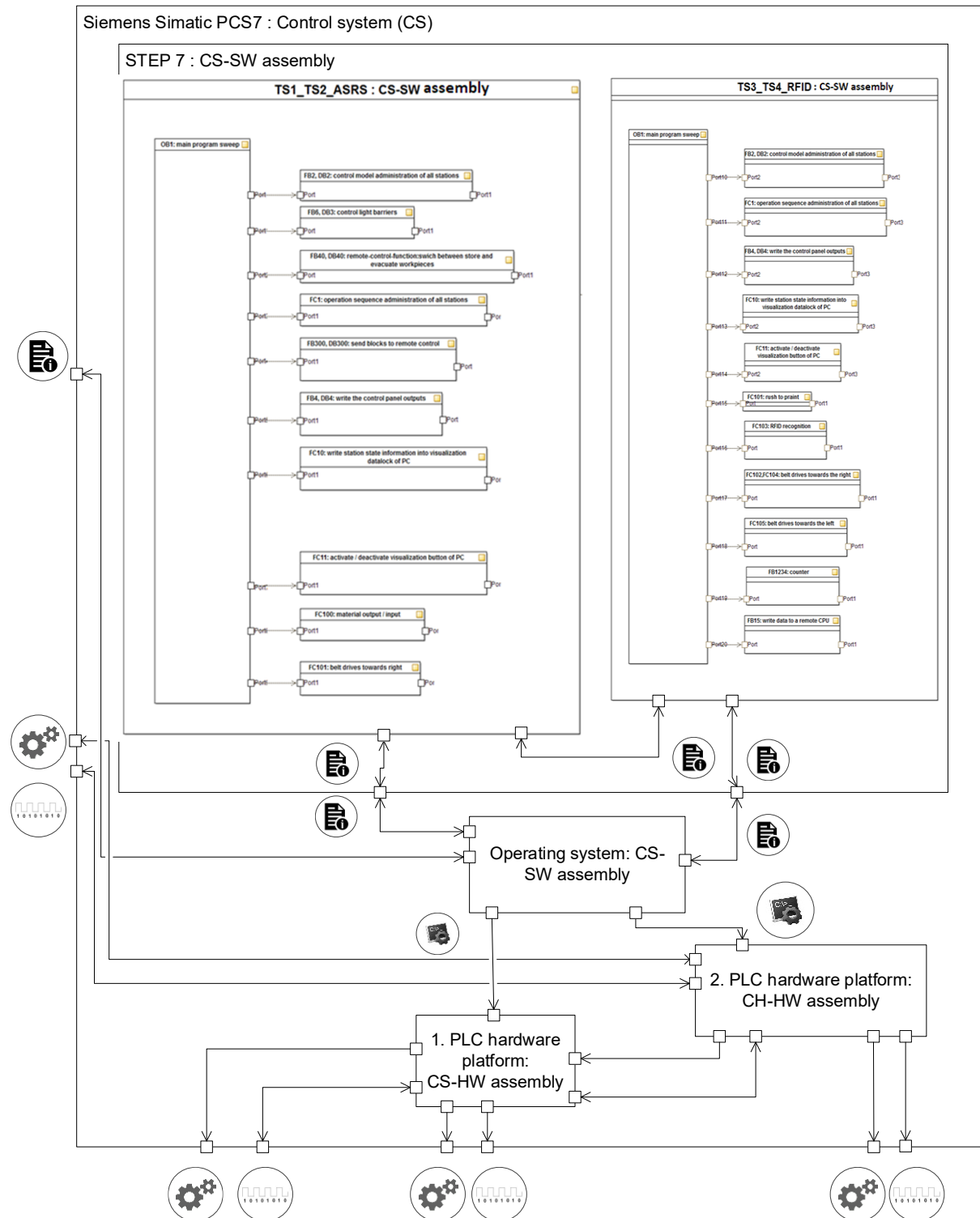


Figure 32: IBD of control system in sample

Figure 32 illustrates the system decomposition of the control system in the laboratory model with an IBD. The CS-HW assembly has two instances, which are two connected PLC hardware platforms. They provide the executions of requirements or functions for their connected software and software assemblies: Moreover, they control the linked mechanical components to implement the software-established processes at the physical level.

The CS-SW assembly has two instances. They are the operating system and the STEP 7 programs, which comprise two assemblies: TS1\_TS2\_ASRS and TS3\_TS4\_RFID. These two software assemblies comprise different function block, which are connected together as in a function block diagram.

**Description of information system**

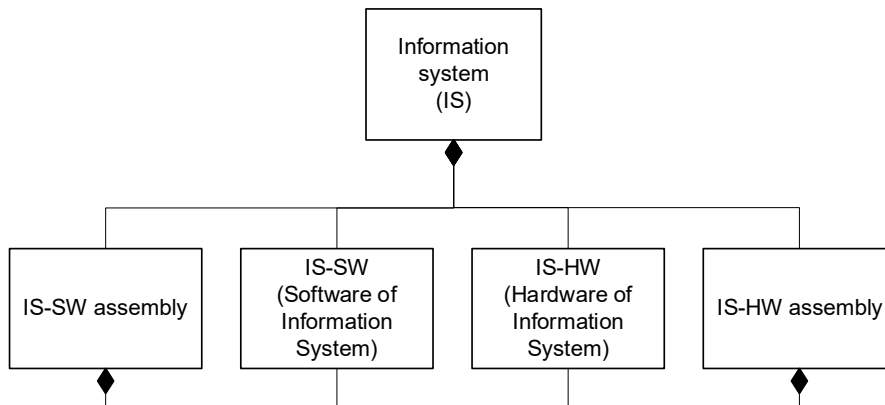


Figure 33: System decomposition of Information system with BDD

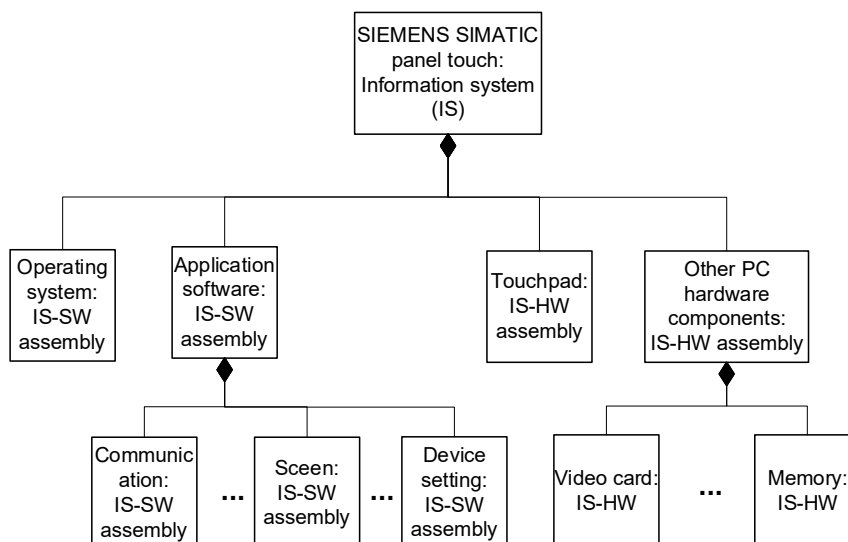


Figure 34: BDD of information system in sample

### Chapter 3 - Problem Statement and Analysis with Example

An information system is decomposed into a set of software (IS-SW), a set of software assembly (IS-SW Assembly), a set of hardware (IS-HW) and a set of hardware assembly (IS-HW Assembly) (see Figure 33). They deal with each other in the collection, processing, organization and storage of data and information. In the work of Boell and Cecez-Kecmanovic, the “Information systems (IS) involve a variety of information technologies (IT) such as computers, software, databases, communication systems, the Internet, mobile devices and much more, to perform specific tasks, interact with and inform various actors in different organizational or social contexts” [44]. Compared with the process system, the IS is not a real time system and it has an open, dynamic and distributed system structure. Figure 34 shows a BDD for the decomposition of the information system in the laboratory model. A Siemens Simatic Panel Touch (see Figure 35) is modeled as the IS in this sample. It comprises a touchpad and integrates other PC hardware components as two hardware assemblies, and an operating system and an application software as two software assemblies. This IS has a human-machine interface, which allows workers read and control the functions in this LL-CPS.

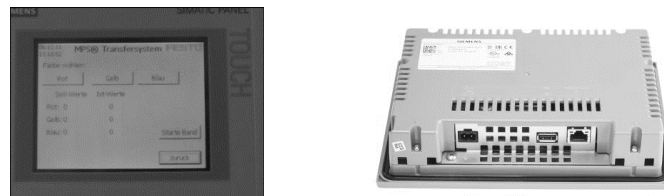


Figure 35: SIEMENS SIMATIC PANEL TOUCH

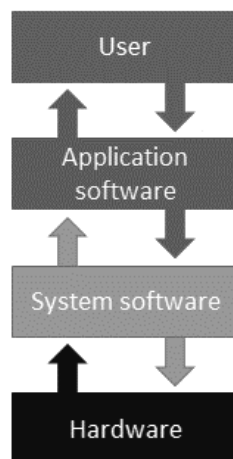


Figure 36: System software and application software relation

The IS-SWs and IS-SW assemblies in this sample are divided into two groups: operating system software and application software. The operating system software manages the resources of the computer hardware. The application software takes the functions, tasks and activities from the user and implements them during the operating system software and then the

### Chapter 3 - Problem Statement and Analysis with Example

hardware. Figure 36 shows the relationships between the user, application software, (operating) system software and hardware. In Figure 37, an internal block diagram describes the system combination of the IS in the laboratory model.

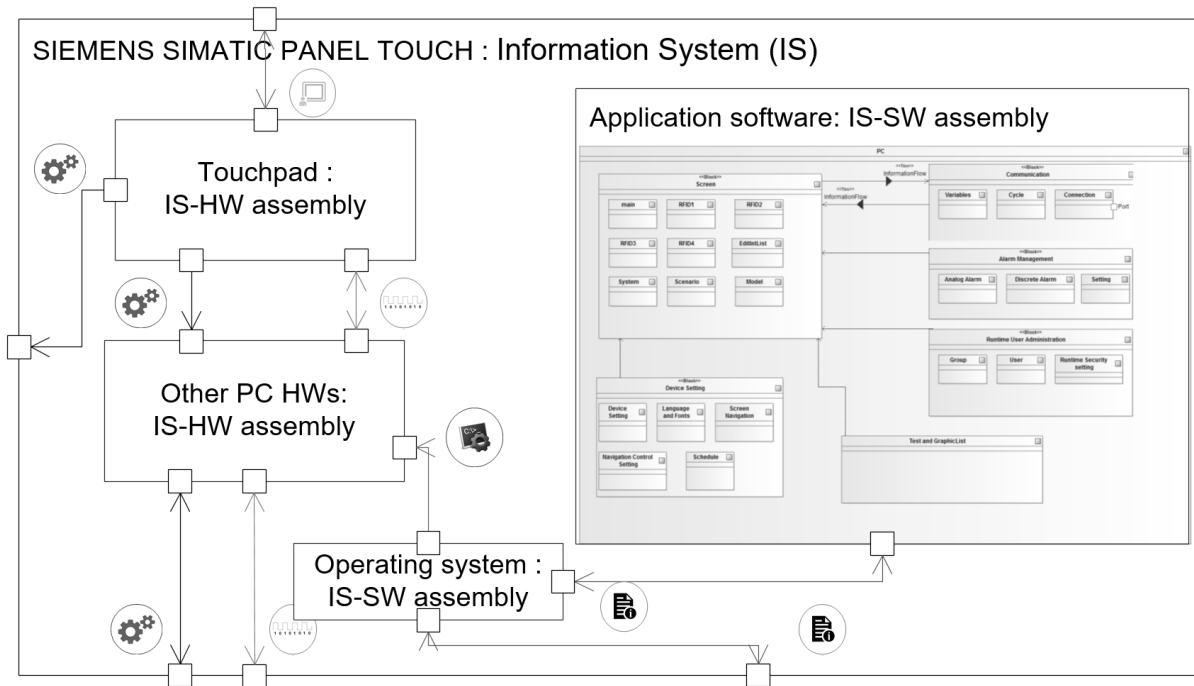


Figure 37: IBD of information system in sample

### Summary of system decomposition

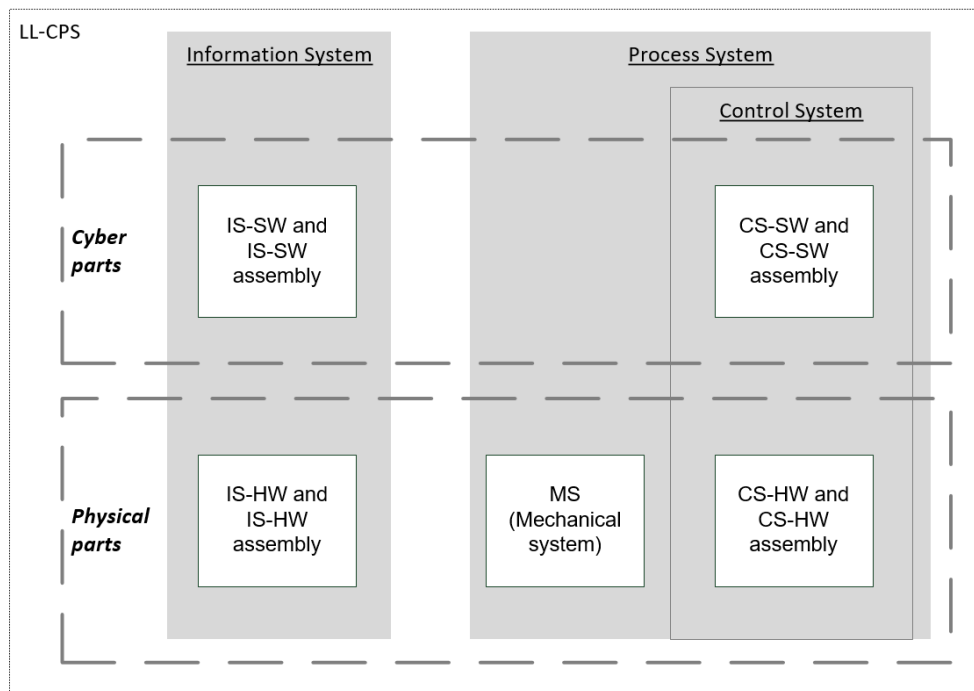


Figure 38: System structure for a LL-CPS

### Chapter 3 - Problem Statement and Analysis with Example

The decomposed components in the LL-CPS are working either closely together on one task or completely independently of each other. From the aspect of attributes, all of the software components like the IS-SW, IS-SW assembly, CS-SW and CS-SW assembly, constitute the cyber part of the LL-CPS. The IS-HW, IS-HW assembly, CS-HW, CS-HW assembly and MS-HW constitute the physical part of the LL-CPS. On the other hand, the different software, hardware and assemblies constitute the information system and process system in this LL-CPS (see Figure 38). Figure 39 shows an IBD representing the decomposition of the conveyor system with ASRS as a sample.

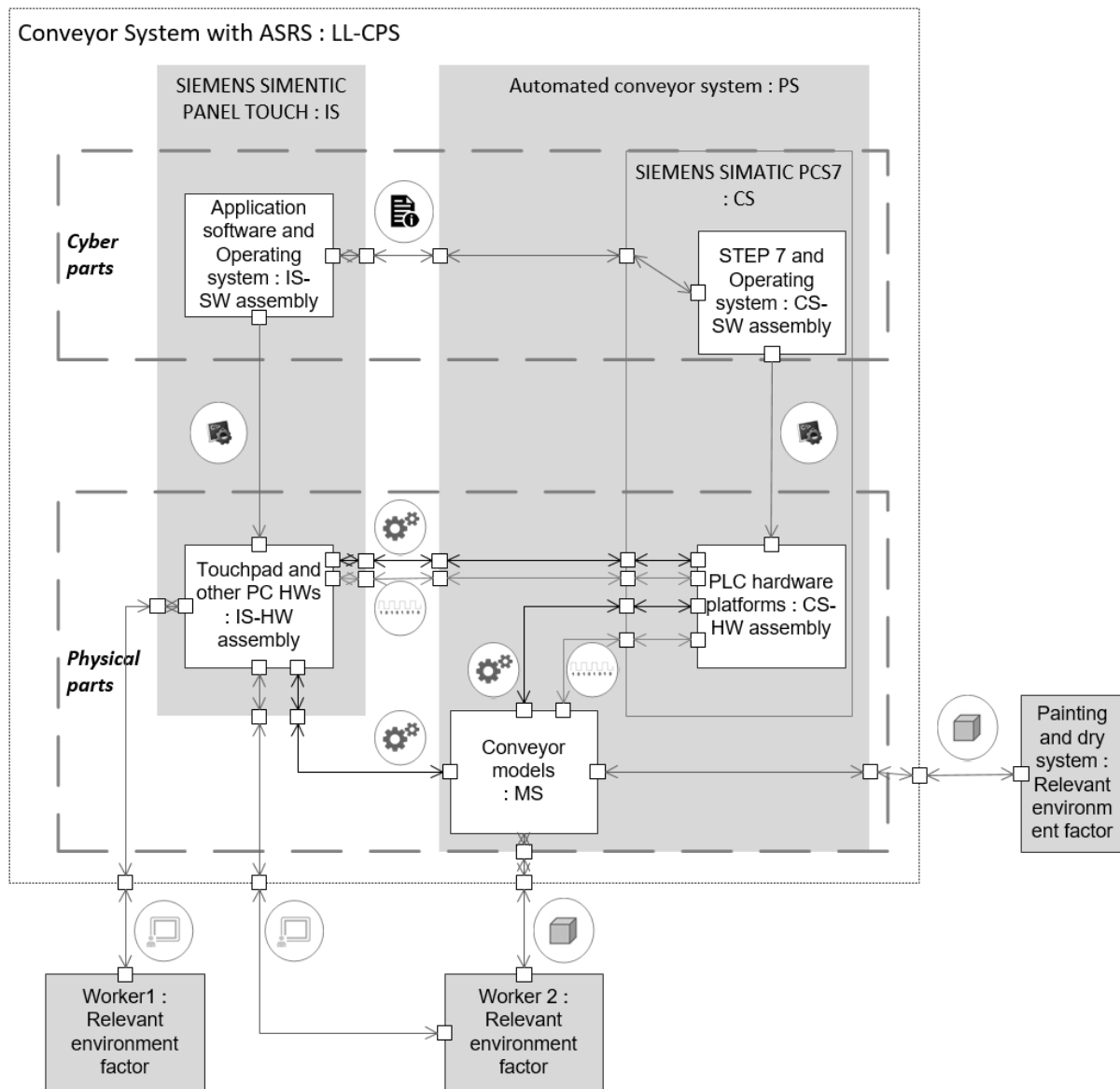


Figure 39: System decomposition of the conveyor system with ASRS

## 3.2 Managed evolution scenario of LL-CPS

### 3.2.1 Problems of the ongoing LL-CPS

A conveyor system with ASRS has been described and analysed as an ongoing LL-CPS. From the perspective of production efficiency, this ongoing conveyor system with ASRS is not perfect, because its production time can be reduced. The manual work of the workers in the existing conveyor system could cause an increase in production time, and an automated machine definitely has higher production efficiency since no thinking is needed by the machine. Furthermore, the worker who stands by the buffer belt repeatedly performs the same task (sort the wares), which increases the risk of making mistakes. In many factories, this is a main reason for the poor product quality.

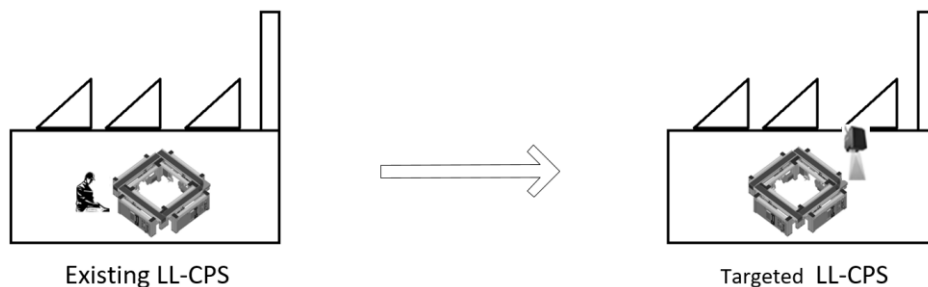


Figure 40: Managed evolution of a LL-CPS

In conclusion, this existing conveyor system with ASRS is not production time effective and quality assurance. According to the problems of the existing LL-CPS, a targeted status of this LL-CPS is defined (see Figure 40). In this thesis, the development of LL-CPS is managed, which means that one system evolution step is defined from the existing status to the targeted status of a LL-CPS. The targeted status must be cleanly defined.

### 3.2.2 The targeted status of this LL-CPS

The new work processes in the targeted status of this LL-CPS are based on the existing LL-CPS. However, it is characterized by the high degree of the ability of automatization.

- Extract, register color and painting Color:  
In the targeted status of this LL-CPS, the wares are taken out with the gripper robot from the warehouse. After obtaining painting information, the wares will be transported through the buffer belt to the painting and dry hall. These processes are as same as the processes in the existing LL-CPS.



### Chapter 3 - Problem Statement and Analysis with Example

- Read color:**  
 In this process, there are some differences compared with the existing status. There is no worker standing by the buffer belt to sort the wares that come back from the painting hall. Instead of the worker, a new RFID read sensor is procured and installed on the conveyor belt. It is used to read the color information from the wares.
- Test number:**  
 This process is as same as the process test number in the existing status of this LL-CPS. The retrieved wares will be counted by using a photoelectric sensor according to the given information from the engineer in the register color process.
- Retrieve:**  
 The wares will be retrieved through the gripper robot with the color information, which is read with the RFID read sensor in the read color process, in the predefined location or floor in the warehouse.

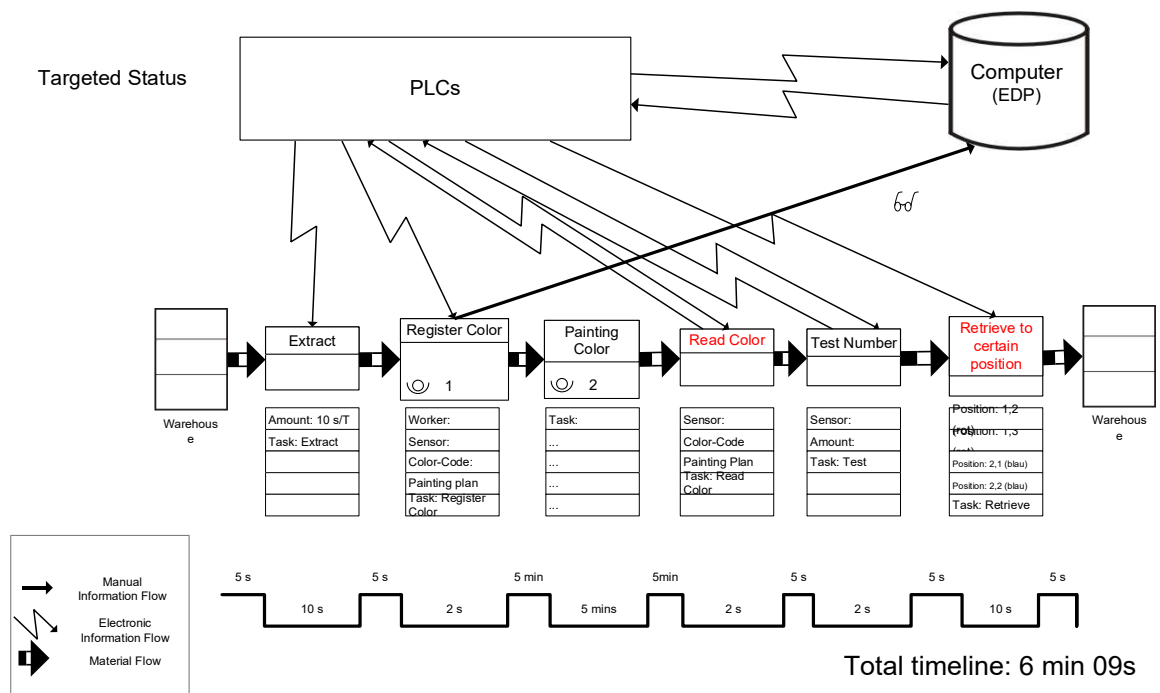


Figure 41: Targeted status of this LL-CPS

Figure 41 describes the targeted status of this LL-CPS with a VSM model. During this development, the manual sort work is removed, which can improve the ability of automatization and the product quality. In addition, the production time is reduced from 8 minutes 7 seconds to 6 minutes 9 seconds.

### 3.3 State of the art and existing approaches for managed evolution of LL-CPSs

The problems during the evolution of CPS have been recognized by many researchers. In this section, the important related works will be divided into three research fields: system modeling, formal modeling of the system evolution and modeling and optimizing the cost of reconfiguration.

#### 3.3.1 Cyber physical system modeling

**Deynet** [45] specifies the cyber-physical system based on a module architecture in his diploma thesis. By using of a pilot project in car as an example, Deynet introduced two methods for the system specification: the interfaces specification and the perspective specification. The interfaces specification emphasizes the description of the standardizing and specification of the interface types between the modules. Once the interface types are determined, the types must be kept, although the number of interfaces in the determined types can be continually specified. Figure 42 show an example of the interfaces specification in Deynet’s work.

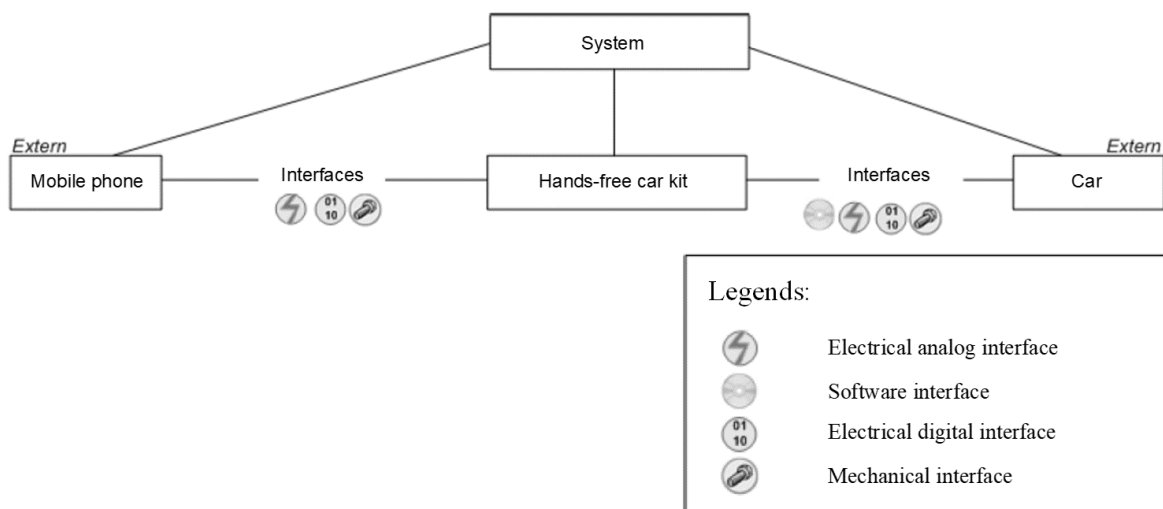


Figure 42: The interfaces specification for a pilot system in car

All of the interfaces in this system are specified into four different types: the electrical analog interface, the software interface, the electrical digital interface and the mechanical interface. Such interfaces can be used for a further specification in a sub-system. For example, the hands-free car kit is specified into a cradle, baseplate, ECU (A mounted box under the interior trim of car) and microphone (see Figure 43).

### Chapter 3 - Problem Statement and Analysis with Example

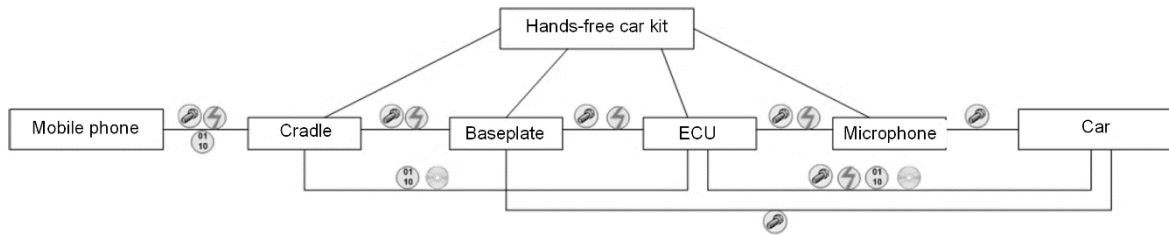


Figure 43: The interfaces specification for the Hands-free car kit segment

The perspective specification is focusing on the system specification at different system levels. At the each level, there is its own individual view and interface types of the overall system. The advantage of this specification is that each interface type appears only once in one system level. This makes the documenting of the system specification very clear and compact. In addition, the interdisciplinary specification is inevitable.

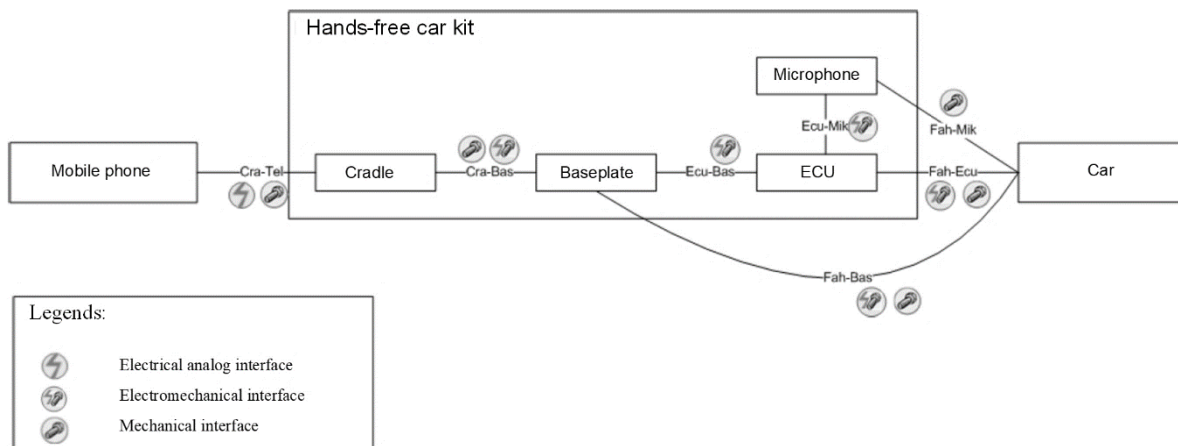


Figure 44: The perspective specification for the Hands-free car kit segment in consideration only of mechanical, electrical analog and electromechanical interfaces

Figure 44 shows a perspective specification for the hands-free car kit segment, in which the specification only focuses on the mechanical, electrical analog and electromechanical interfaces. The name of the line represents the two connecting blocks, like Cra-Tel for a cradle and mobile phone (Telephone).

**Bartelt et al.** [46] defined a cyber-physical system based on the following characteristics. First, a CPS is a system of systems can be decomposed into information systems and control systems. Between them, there is an intelligent interface to join the different properties from the two kinds of systems. Secondly, the information systems and control system are decomposed in modular building blocks named components. The intelligent interface is also decomposed to fine-grained intelligent interfaces to network the components between the information system and control system. Figure 45 shows the decomposition of a CPS with smart interfaces.

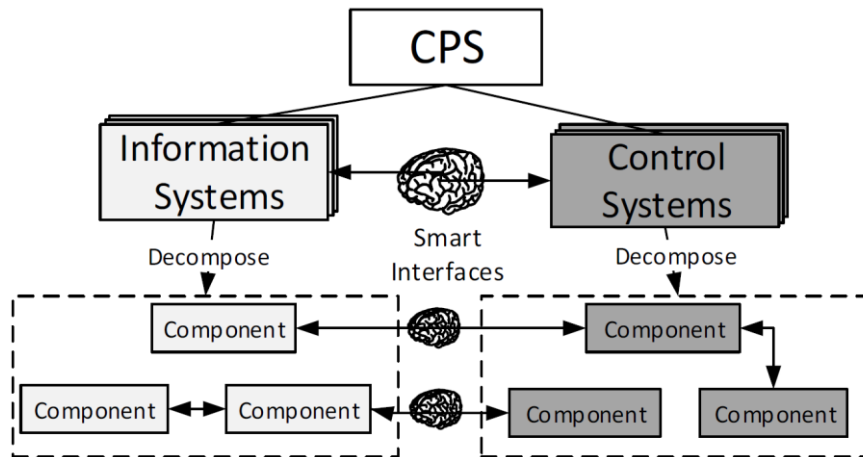


Figure 45: The CPS (decomposed) with smart interfaces

**Larsen et al.** [47] described an approach using the example of a small unmanned aerial vehicle to model the cyber-physical systems and enable integration of multiple models and tools in a consistent tool chain. A CPS is decomposed to the discrete-event (DE) models of computational processes and the continuous-value and continuous-time (CT) formalisms of physical engineering [48]. The VDM-RT (an extension of the Vienna Development Method's modelling language with features for object orientation, concurrency and real-time computation, including the distribution of processes to virtual CPUs) is used to notate the DE models and the 20-sim (a package for modelling and simulating complex physical systems) to notate the CT models. The architecture of DE models is presented here using SysML. A co-modeling framework Crescendo is introduced, whereby the interfaces between DE and CT models identify the shared features of the two constituent models. P. Larsen was working on an integrated tool chain (INFO-CPS tool chain) to make a co-simulation of more than two simulation tools possible. The CPS architecture on INFO-CPS is expressed using SysML, which allows cyber and physical elements to be identified such that each of these elements corresponds with a constituent model. However, there is a need for integrated development methods that span from requirements through to analyzing the results of simulation. Another need is for efficiently managing the traceability of design artefacts to analyze the changes impact and evidence the dependability of CPS development.

In a cyber-physical system, the physical systems are inherently described by non-causal continuous-time equations. On the other hand, the cyber-based information and communication systems are based on the notion of causality and discrete-time semantics. In another work of **Simko et al.** [49], the formalization of composition and interactions in the two worlds is regarded as the new challenge for the model-based engineering of cyber-physical system. A CPS-specific modeling language (CyPhyML) is developed and used to define the structure and behavior of physical and computational components. It supports the non-causal and causal modeling and facilitates hierarchical composition. Simko et al. formalized a CyPhyML model as a tuple, which comprises a set of components, a set of component

assemblies, a set of design elements, a union of the sets of ports, a containment function for design elements and component assemblies, a port containment function, a set of power flow and a set of information flow (see Figure 46). The union of the sets of ports is specified into eight different types: the rotational mechanical power ports, the translational mechanical power ports, the multi-body power ports, the hydraulic power ports, the thermal power ports, the electrical power ports, the continuous time input signal ports and the continuous time output signal ports. Furthermore, all of the power ports are encapsulated into a union and all of the signal ports into another union. The power flow links any power ports together, and the information flow links any signal ports together. Through the mathematically rigorous and unambiguous formal specifications of CPS, the structural and behavioral specifications can be written using the same modeling language, whereby both can be used for deductive reasoning. The use of these formalizations for model checking, deductive reasoning and correctness proofs will be a matter of future work of Simko.

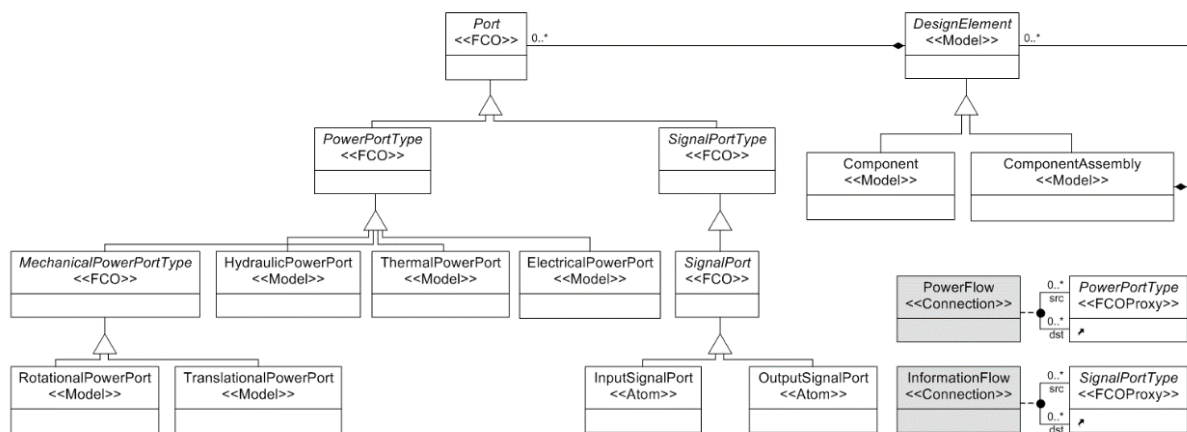


Figure 46: The generic modeling environment meta-model for the composition sub-language of CyPhyML

In general, a complex cyber-physical system is typically modeled from different disciplines for developing and evaluating design alternatives within the context of formalisms that are relevant to selected aspects of the system. Each modeling aspect highlights certain features and occludes others to make analysis tractable and to focus on particular performance attributes [50]. **Bhave et al.** introduced a base architecture of a CPS, which contains detail to convey the nature of information and physical quantities flowing between components. A quadrotor is modeled using multi-domain models, namely a physical model, software model, hardware model and control model, which represent the same cyber-physical system from the physical, software, hardware and control design domain perspectives. Figure 47 shows the conceptual relationship between system models, views and the base architecture.

Continuing with the case of a quadrotor, **Bhave et al.** built the encapsulation-based relation  $R_{V_X}^X$  from model X to view  $V_X$ , and then the encapsulation/refinement-based relation  $R_{BA}^{V_X}$  between view  $V_X$  and the base architecture of CPS. On the other hand, model Y can be also transformed to the base architecture and with model X in the same level. The relation  $R_{V_X}^X$  is defined as a one-to-one or one-to-many maps. The relation  $R_{BA}^{V_X}$  is defined as a combination of one-to-many and many-to-one maps, although the many-to-many maps are not allowed. In Bhave et al's work, several research issues are not solved, including the rules to determine the encapsulations, as well as the combination of multiple connectors into a single one. Finally, the consistency during the transformation will need to be prescribed.

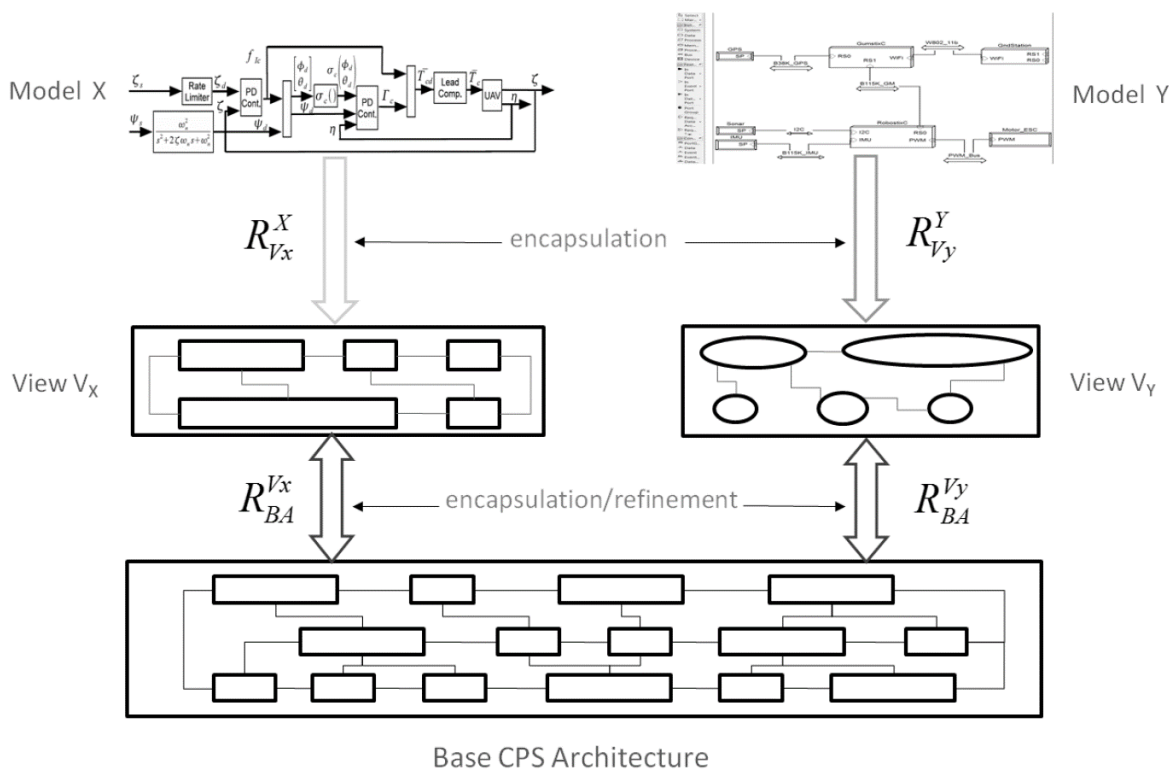


Figure 47: The conceptual relationship between system models, views and the base architecture of CPS

In the book of **Tiller** [51], a modeling approach was introduced for the multi-domain models, which are characterized by the fact that they have components belonging to different engineering domains. Hereby, a conveyor belt system was modeled from the mechanical and electrical domains. The mechanical domain is associated with the electronic domain during the position and speed sensors. The effective inertia between the mechanical components must be formulated for a combination of the two rigidly-connected inertias.

### 3.3.2 Formal modeling of system evolution

A synergy between the systems modeling languages SysML and Modelica, which is a standard for modeling the continuous dynamics of systems in terms of hybrid discrete-event and differential algebraic equation systems, is the core work of **Johnson et al.** [52], in which the use of a triple graph grammar (TGG) to keep a bi-directional mapping between these SysML constructs and the corresponding Modelica models and create Modelica models from SysML models is the highlight of this work (see Figure 48 [41] (as cited in [52])). The SysML models and the Modelica models can be represented in an abstract syntax, which is defined by a metamodel. These metamodels are represented as graphs. The mappings between the SysML blocks and Modelica Classes are represented as a set of correspondence relationships. For instance, at the syntax level, a SysML block (in the SysML Metamodel graph) is mapping to a Modelica class (in the Modelica Metamodel graph) using a relationship entity block2class. These correspondence relationships with source and target points compose to a correspondence graph to represent the correspondences between SysML and Modelica models. Through the code generation technology, the created Modelica models to code are generated automatically.

While Johnson et al. defined formal meta-level mappings for relating the different modeling representations SysML and Modelica, the changes of design during system engineering cannot be traced from SysML to Modelica.

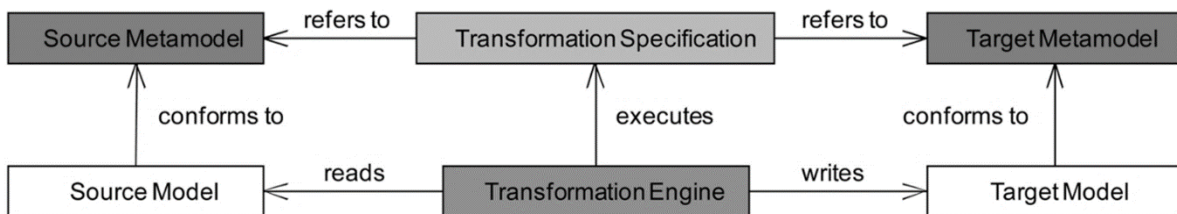


Figure 48: Triple Graph Grammar Formalism

The work of **Youness et al.** [53] focused on the system development in a multi-modeling domains. The models in different modeling domains need to be composed for validation of partial models and synchronization tasks, which is an error-prone activity. Therefore, a traceability mechanism and a formalization of the model composition operation and the corresponding traceability were introduced to support a composition in multi-modeling domains. The formalization of composition operation comprises the formalization of composition operators, composition specification, composition rules, the execution of composition specification, the composition rule activation, model elements in a sub-set, the explicit and implicit rule call and target equivalence resolution. The formalization of model composition traces comprises the formalization of traces structuring and traces generation.

Nevertheless, the formalization of Youness et al. is not complete for every situation. For example, the formalization can be extended to generate more complete traces. In addition, the formalization of composition operation and model composition traces is tedious and complex.

**Padilla** [54] introduced a middleware dedicated to CPS to manage the components deployment and the dynamic reconfiguration on the software layer of a cyber physical system (see Figure 49). When new models or the new services or new creation of new bindings between the existing components need to be deployed, this middleware can propose an adoption of the *model@runtime* paradigm to the specific constraints of CPSs. The *model@runtime* is a model-based application to provide an automated synthesis to support component interoperability. The models in CPS are abstracted to nodes, which need to be dynamically reconfigured and redeployed to meet the CPS evolution and user preferences. The following figure illustrates the architecture of reconfiguration on each node of the CPS. First, new models are received to define the new targeted status of the CPS. Second, the set of local adaptations are defined to reach this new status. Finally, the various local adaptations are enacted on the node. The set of local adaptations comprise integrations of new pieces of codes, instantiations or removing of existing components and channels used to bind them, or reconfiguration of the value of any attribute.

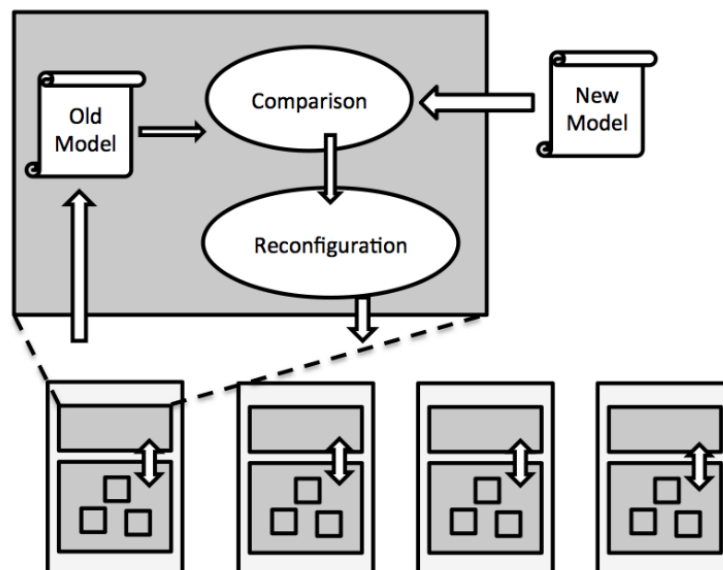


Figure 49: The architecture of reconfiguration on each node of the CPS

However, this initial work only focuses on the software components deployment and the dynamic reconfiguration on the software layer, whereas it does not consider the components on the hardware layer of CPS.



### 3.3.3 Modeling and optimizing the costs of reconstruction

**Orabi et al.** [55] developed a model for the reconstruction costs of damaged transportation networks. In this model, the total reconstruction costs comprised the direct costs ( $DC$ ) and indirect cost ( $IC$ ). The non-construction related costs, e.g. road user and business disruption, are not included in this model. The direct cost  $DC$  includes the cost of resources for the reconstruction and they are calculated using the equation below. The indirect cost  $IC$  includes time-dependent costs.

$$\text{Total reconstruction costs} = DC + IC$$

$$DC = \sum_{m=1}^M \sum_{x=1}^X R_m^x dc_r$$

$$IC = \sum_{m=1}^M d_m ic_m$$

The variable  $d_m$  represents the duration of each project ( $m$ ). The variable  $R_m^x$  represents the resource requirements for the activity ( $x$ ) of project ( $m$ ). The variable  $ic_m$  is the indirect cost unit rate for project ( $m$ ). The  $dc_r$  is the unit cost of resource ( $r$ ). The total number of projects is expressed with  $M$ .

## 3.4 Research questions of this thesis

Concluding from the problem analysis and literature research, the main research questions are formulated as follows.

**Research question 1:** How can a LL-CPS and its managed evolution be formally modelled and described?

**Research question 2:** How can the changes in the managed evolution of a LL-CPS be formally derived?

**Research question 3:** How can an approach be developed for the managed evolution of a LL-CPS, in respect of a local minimum of the costs of reconstruction for implementation and the controlled risks in ongoing operations?

**Research question 4:** How can the developed approach be demonstrated and evaluated?



## 4 Formal Descriptions and Transformations of Managed Evolution of LL-CPSs

### Content

---

- 4.1 Formal description for VSM
    - 4.1.1 Formal semantical foundation
    - 4.1.2 Model-based description
    - 4.1.3 Semantical mapping
    - 4.1.4 Concrete modeling
  - 4.2 Formal description for IBD
    - 4.2.1 Formal semantical foundation
    - 4.2.2 Model-based description
    - 4.2.3 Semantical mapping
    - 4.2.4 Concrete modeling
  - 4.3 Formal mapping relation from VSM to IBD
    - 4.3.1 Formal semantical foundation
    - 4.3.2 Model-based description
  - 4.4 Formal managed evolution of LL-CPSs
    - 4.4.1 Formal semantical foundation
    - 4.4.2 Model-based description
- 

Before introducing an approach to reduce the development risks and ascertaining the local optimal cost of reconstruction for the managed evolution of LL-CPSs, a formal description mechanism will first be introduced to represent the managed evolution of LL-CPS and the model transformations.

In mathematics, computer science, and linguistics, a formal description is an abstract description that comprises a set of symbol, letters or tokens together with a set of specification rules. The formal description can be textual or graphical, but is often a mix of both [56]. This abstract description allows engineering systems by concentrating first on its core functionalities while deferring secondary concerns like details of the final execution platform. The details discarded earlier are retrieved later on at a lower level of abstraction or in the instance.

In this formal description mechanism, the process-oriented modeling method VSM and the component-oriented modeling method IBD, which have been introduced in chapter 3, are applied as two different model-based descriptions to represent the same LL-CPSs on a description layer. This description layer is named the model-based description layer. The

formal description of all VSM models on a model-based description layer is represented with a set of models  $M_{VSM}$ . All IBD models are formed with a set of models  $M_{IBD}$  (See Figure 50). If all models on the model-based description layer are formed with a set  $M$  (see Definition 19), then the  $M_{VSM}$  and  $M_{IBD}$  are subsets of  $M$  and disjoint sets.

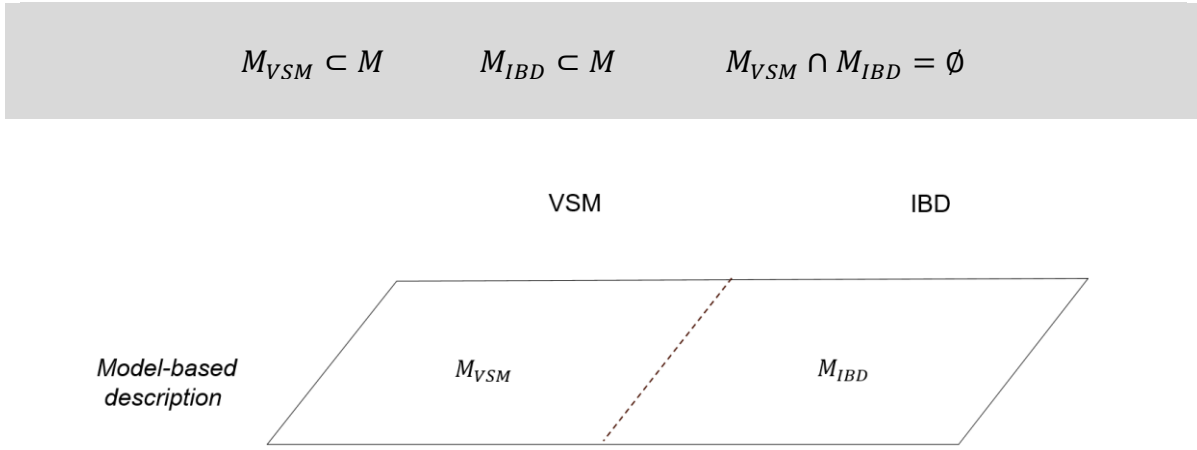


Figure 50: VSM and IBD models on model-based description layer

The VSM and IBD models can be formed on a uniform description layer named formal semantical foundational layer. The models on this layer are formed with a uniform description: a graph-structure. This set VSM graphs is formed with a set  $G_{VSM}$ . The set of IBD graphs is formed with a set  $G_{IBD}$  (See Figure 51). If all of the graphs on the formal semantical foundation layer are formed with a set of models  $G_{in}$  (see Definition 4), then the  $G_{VSM}$  and  $G_{IBD}$  are subsets of  $G_{in}$  and disjoint sets.

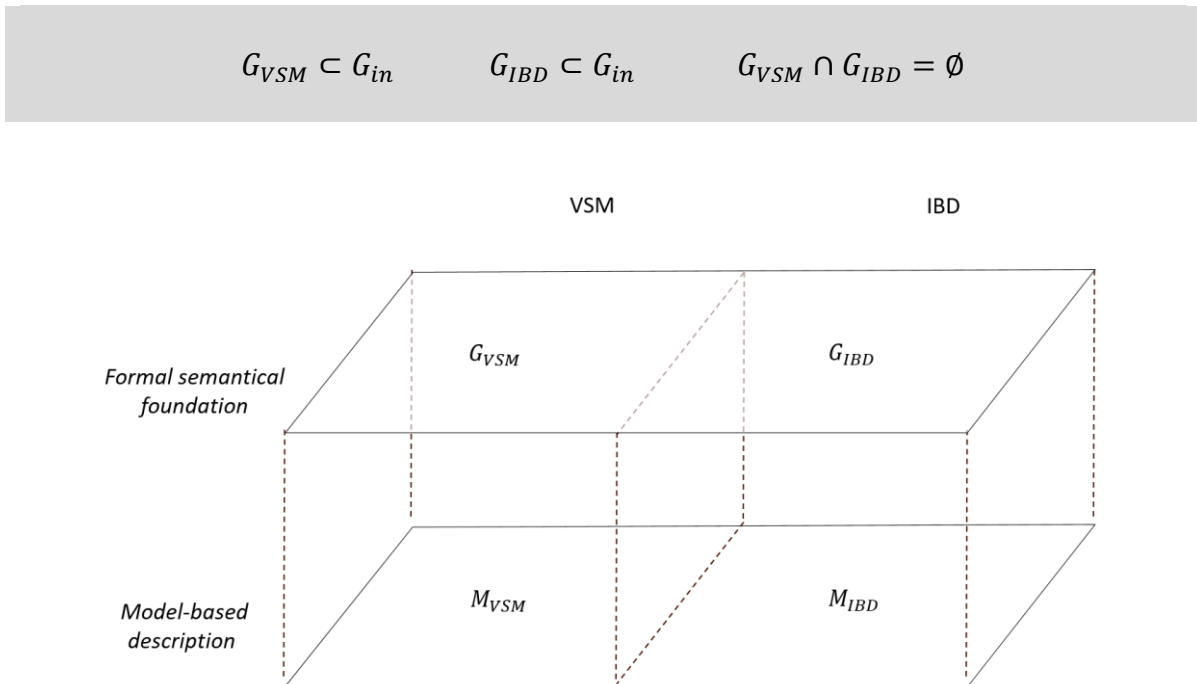


Figure 51: VSM and IBD models on two description layers

The managed evolution of a LL-CPS can be represented with a sequence of system statuses. If an ongoing LL-CPS is defined as the existing status, its goal LL-CPS can be defined as its next status: the targeted status of this LL-CPS. When the targeted status is implemented, it can be used as a new existing status. Based on this new existing status, the next new LL-CPS can be continually developed to the next targeted status. Therefore, the managed evolution process of a LL-CPS can be expressed with a continual iteration from its existing status to its targeted status (See Figure 52).

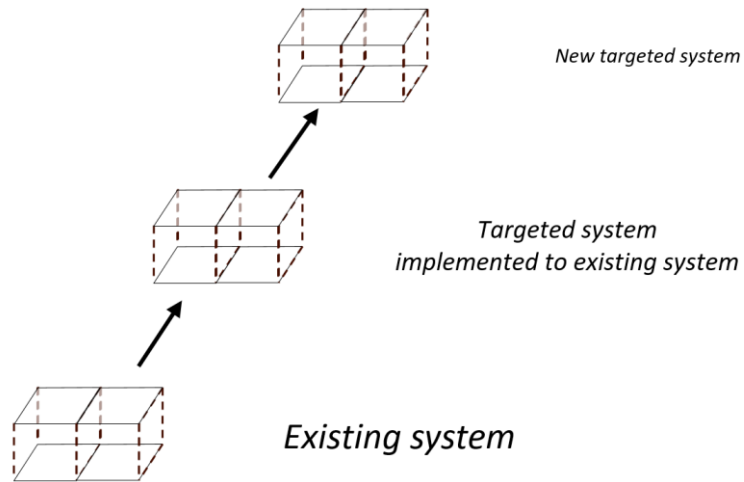


Figure 52: The managed evolution of a LL-CPS

Figure 53 shows a cube model in a three-dimensional axes system, on which the LL-CPSs are described with two modeling methods: VSM and IBD. Each modeling method has two description layers: the model-based description layer and formal semantical foundation layer. The managed evolutions of these LL-CPSs are described with the developments from their existing statuses to targeted statuses. This cube model is divided into eight areas, each of which represents the LL-CPSs by different description layers, modeling methods and statuses.

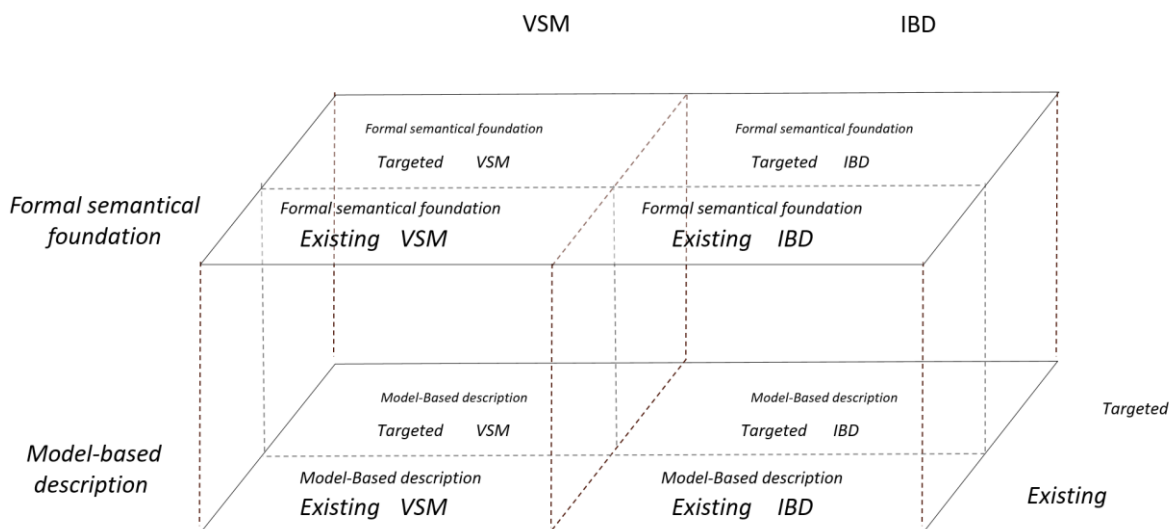


Figure 53: Cube model with eight areas

This cube model with eight areas provides a basic frame for the model transformation, model derivation and system managed evolution on two description layers. A predicate is formed with a Boolean-valued function  $status$  to represent the statuses of VSM and IBD models during the managed evolution of LL-CPSs. A model  $m$  can be a VSM model or IBD model. The function  $status$  maps this model to a status: existing or targeted. The existing status means that this model is representing an ongoing system. The targeted status means that this model is representing a targeted system.

Definition 27

$$status := (M_{VSM} \cup M_{IBD}) \rightarrow \{existing, targeted\}$$

$$m \in M_{VSM} \cup M_{IBD}$$

All graphs for the ongoing LL-CPSs are formed with the  $G_{existing} \subset G_{in}$ , and all models with the set  $M_{existing} \subset M$ . All graphs for the targeted LL-CPSs are formed with the  $G_{targeted} \subset G_{in}$ , and all models with the set  $M_{targeted} \subset M$ . The  $G_{existing}$  and  $G_{targeted}$  are two disjoint sets and the sets  $M_{existing}$  and  $M_{targeted}$  are also disjoint.

$$G_{existing} \subset G_{in} \quad M_{existing} \subset M$$

$$G_{targeted} \subset G_{in} \quad M_{targeted} \subset M$$

$$G_{existing} \cap G_{targeted} = \emptyset$$

$$M_{existing} \cap M_{targeted} = \emptyset$$

All VSM models for the ongoing LL-CPSs can be formed with a set of models  $M_{VSM,existing}$ . The targeted statuses of these LL-CPSs are formed with a set of models  $M_{VSM,targeted}$ . They are two subsets of the set  $M_{VSM}$  and disjoint sets on account of status. On the formal semantical foundation layer, the ongoing LL-CPSs and their targeted statuses are formed with a set of graphs  $G_{VSM,existing}$  and a set of graphs  $G_{VSM,targeted}$ . These sets are disjoint and subsets of the set of VSM graphs  $G_{VSM}$ .

$$M_{VSM,existing} \subset M_{existing} \quad M_{VSM,targeted} \subset M_{targeted}$$

$$M_{VSM,existing}, M_{VSM,targeted} \subset M_{VSM}$$

$$G_{VSM,existing} \subset G_{existing} \quad G_{VSM,targeted} \subset G_{targeted}$$

$$G_{VSM,existing}, G_{VSM,targeted} \subset G_{VSM}$$

On the other hand, all IBD models for the same ongoing LL-CPSs are formed with a set of models  $M_{IBD,existing}$ . The same targeted statuses of these LL-CPSs are formed with a set of models  $M_{IBD,targeting}$ . These two sets are subsets of the set of IBD models  $M_{IBD}$  and they are two disjoint sets on account of status. On the formal semantical foundation layer, they are represented with the sets of graphs  $G_{IBD,existing}$  and  $G_{IBD,targeted}$  (See Figure 54). These sets are subsets of the set of IBD graphs  $G_{IBD}$  and disjoint sets on account of status.

$$\begin{aligned}
 M_{IBD,existing} &\subset M_{existing} & M_{IBD,targeted} &\subset M_{targeted} \\
 M_{IBD,existing}, M_{IBD,targeted} &\subset M_{IBD} \\
 G_{IBD,existing} &\subset G_{existing} & G_{IBD,targeted} &\subset G_{targeted} \\
 G_{IBD,existing}, G_{IBD,targeted} &\subset G_{IBD}
 \end{aligned}$$

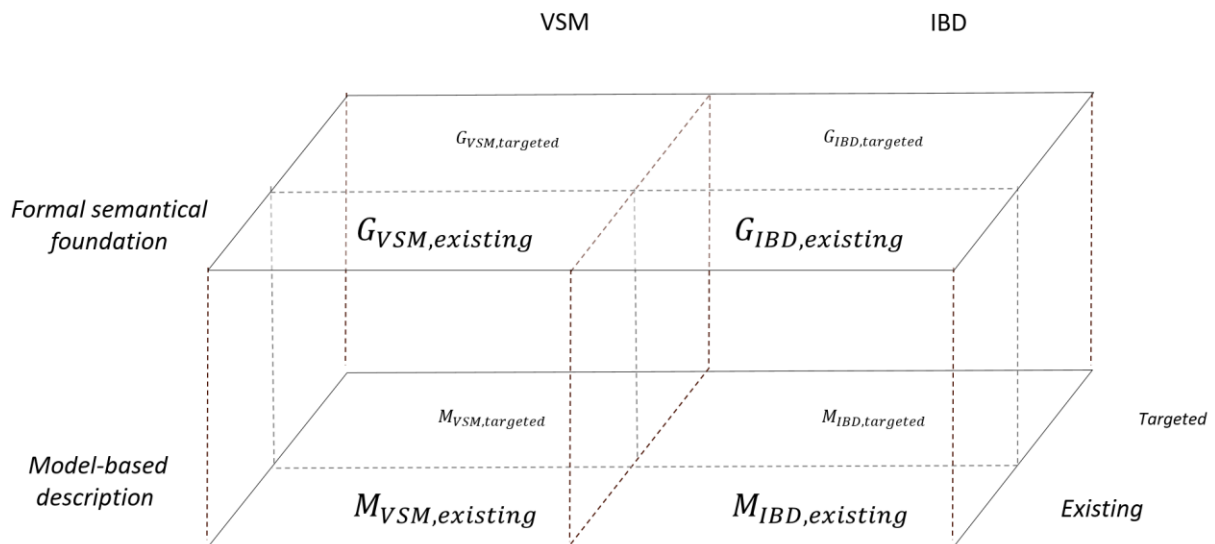


Figure 54: Models and graphs in cube model

There are transformations between the models on model-based description layer and the graphs on the formal semantical foundation layer. They are described with a set of equivalent transformation functions. They keep the models and graphs for the same LL-CPSs structurally and behaviorally identical (See Figure 55). The transformation functions *semv* and *conv* will be introduced in sections 4.1.3 and 4.1.4, the function *semi* will be introduced in section 4.2.3 and the function *coni* in section 4.2.4.

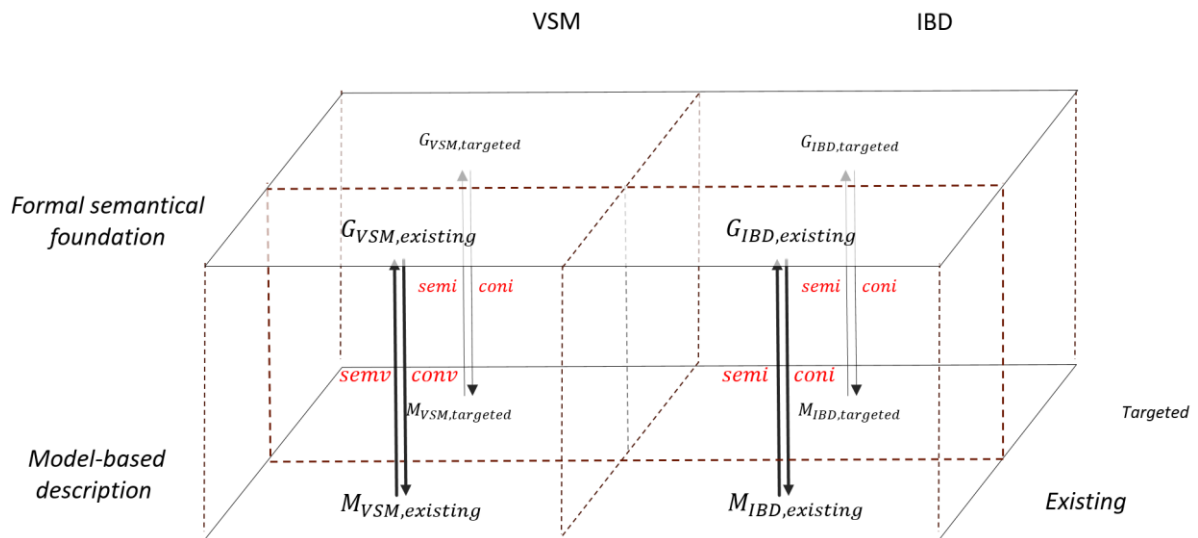


Figure 55: Model descriptions transformations between two description layers

The mapping relationship from a set of VSM graphs to a set of IBD graphs on the formal semantical foundation layer is represented with a set of mapping functions  $\{h_i\}$ . A set of functions  $\{hm_i\}$  represents the mapping relationship from a set of VSM models to a set of IBD models on the model-based description layer in Figure 56. These mappings will be introduced in section 4.3.

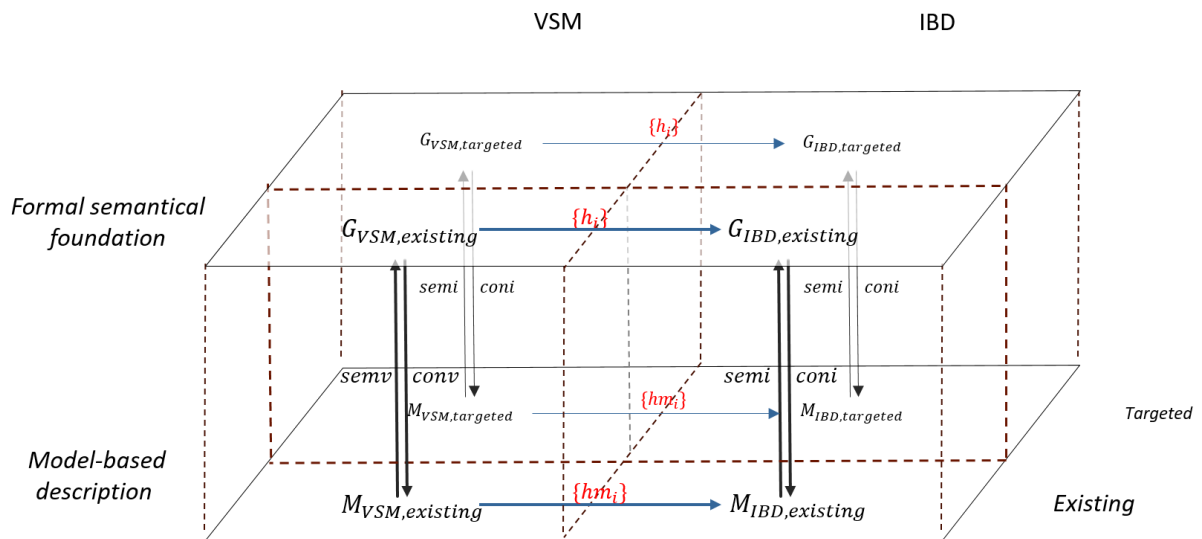


Figure 56: Mapping functions from VSM to IBD side

The managed evolution from the ongoing LL-CPSs to their targeted statuses is formed with a set of unidirectional functions  $\{f_i\}$  on the formal semantical foundation layer and a set of unidirectional functions  $\{fm_i\}$  on the model-based description layer (See Figure 57). Section 4.4 introduces the formalizations of these evolution functions.



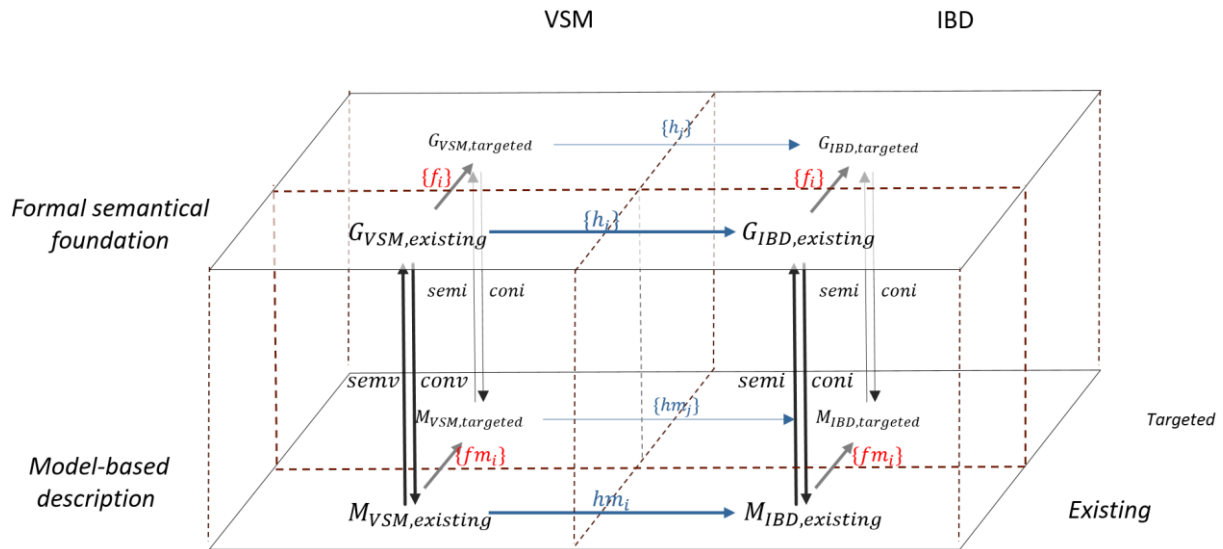


Figure 57: Evolution functions for managed evolution of LL-CPSs

Figure 58 summarizes all of the formal descriptions, model transformations, model mapping relationships and managed evolution functions for the managed evolution of LL-CPSs in the cube model and positions the sections in this chapter into this formal description mechanism.

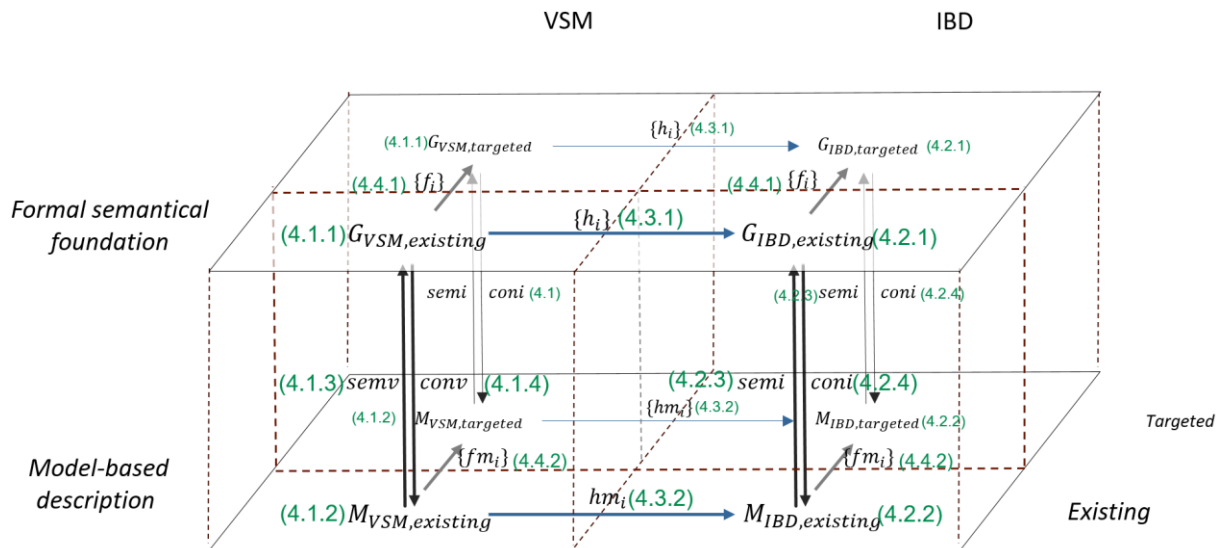


Figure 58: Positions of sections into modeling system

## 4.1 Formal description for VSM

The VSM as a process-oriented modeling method is applied in this section to model the LL-CPSs on the model-based description layer. All of these VSM models are formed as a set of models  $M_{VSM}$ . On the formal semantical foundation layer, they are formed as a set of graphs  $G_{VSM}$ . The models in  $M_{VSM}$  and the graphs in  $G_{VSM}$  model the same LL-CPSs. The transformations between models and graphs are described with the functions *semv* and *conv* (See Figure 59).

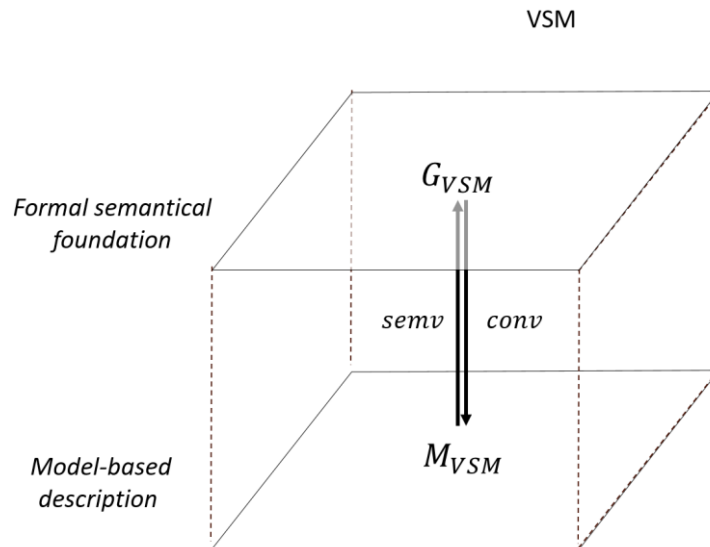


Figure 59: Formal descriptions and descriptions transformation for VSM

### 4.1.1 Formal semantical foundation

The formal semantical foundation is the deep structure of a modeling, which is represented by using a mathematical foundation, Metamodel, graphical representation and example in this section.

#### 4.1.1.1 Mathematical foundation

A graph  $g_{VSM}$  in the set  $G_{VSM}$  is defined as a **directed connected graph structure** (see Definition 4). It comprises a set of vertices  $V_{VSM}$  and a set of edges  $E_{VSM}$ . One edge links two vertices together and can be represented by using a 2-tuple of its source vertex and target vertex. The connect relation from the edge to its source vertex is represented with the function  $src_{VSM}$ , and to its target vertex is represented with the function  $tgt_{VSM}$ . Every vertex and edge has attributes to save information. These attributes are mappings from vertices and edges to a key-value structure comprising a set of strings  $Key_{VSM}$  and a set of strings  $Value_{VSM}$ . The mapping from the keys to values is represented with a hash function.

Definition 28

$$g_{VSM} := (V_{VSM}, E_{VSM}, src_{VSM}, tgt_{VSM}, attributes_{VSM}, Key_{VSM}, Value_{VSM})$$

$$src_{VSM} := src \upharpoonright_{E_{VSM} \rightarrow V_{VSM}}$$

$$tgt_{VSM} := tgt \upharpoonright_{E_{VSM} \rightarrow V_{VSM}}$$

$$E_{VSM} = V_{VSM} \times V_{VSM}$$

$$attributes_{VSM} := (V_{VSM} \cup E_{VSM}) \rightarrow (Key_{VSM} \rightarrow Value_{VSM})$$

$Key_{VSM} :=$  a set of strings

$Value_{VSM} :=$  a set of strings

$$g_{VSM} \in G_{VSM}$$

Every vertex must have a unique identifier (ID). The ID of an edge comprises the IDs of its source vertex and target vertex. There is an injective function  $id_{GVSM}$  to map the identifiers of every vertex and edge to strings.

Definition 29

$$id_{GVSM} := (V_{VSM} \cup E_{VSM}) \rightarrow IDs$$

$IDs :=$  a set of strings

4.1.1.2 Metamodel

Every vertex can connect with any number of edges, although one edge must connect only with two vertices: one source vertex and one target vertex. Every vertex and edge can have any number of attributes or have not attribute if no attributes are given (see Figure 60).

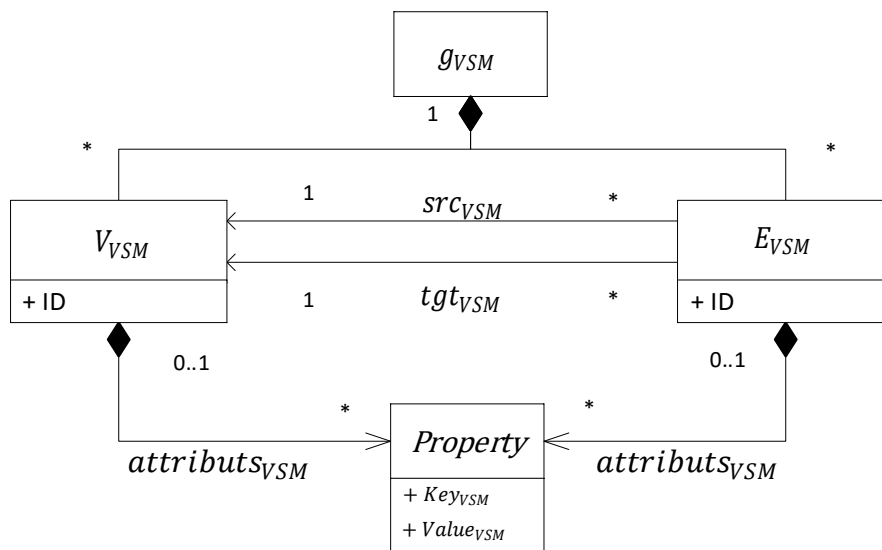


Figure 60: A Metamodel for the formal semantical foundation of VSM model

### 4.1.1.3 Graphical representation

The Table 1 shows the symbols, types, mathematical model element, Metamodel element and description of any graph element in a  $g_{VSM}$ .

Symbol	Type	Mathematical model element	Metamodel element	Descriptions
	Vertex	$v_{VSM} \in V_{VSM}$	$V_{VSM}$	Vertex in graph
	Edge	$e_{VSM} \in E_{VSM}$	$E_{VSM}$	Edge in graph
	Attributes	$Key_{VSM} \rightarrow Value_{VSM}$	<div style="border: 1px solid black; padding: 5px;"> <i>Property</i>                      + <math>Key_{VSM}</math>                      + <math>Value_{VSM}</math> </div>	Attributes of vertices and edges
	Connect relation begin	$src_{VSM}$	$\leftarrow src_{VSM}$	Connect relation for one edge and its source vertex
	Connect relation end	$tgt_{VSM}$	$\leftarrow tgt_{VSM}$	Connect relation for one edge and its target vertex

Table 1. Graphical representation of the formal semantical foundation of the VSM model

### 4.1.1.4 Example

Figure 61 illustrates a VSM graph  $g_a$  in the set  $G_{VSM}$ . Here, two edges and three vertices are linked together as a directed connected graph structure. The id of any edge is expressed with the id of its source vertex to the id of its target vertex. For instance, the edge (a,b) represents a directed edge with the type "Input" from vertex a to vertex b. The attributes are attached to every vertex and edge to save the description information, not only the type, but also the other information like full name, time and costs, etc.

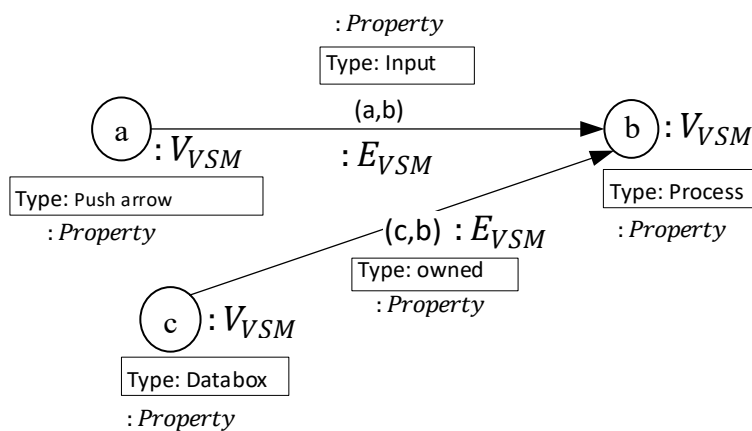


Figure 61: An example for the formal semantical foundation of VSM model

## 4.1.2 Model-based description

On the model-based description layer, a set of VSM models  $M_{VSM}$  describe a set of LL-CPSs, which is introduced by using a mathematical foundation, Metamodel, graphical representation and example.

### 4.1.2.1 Mathematical foundation

Every VSM model  $m_{VSM}$  in the set  $M_{VSM}$  is defined with **an integrated system in a network structure** (see Definition 19) to describe an integrated LL-CPS. It comprises a set of model elements  $ME_{VSM}$  and a set of relation elements  $A_{VSM}$ . They are combined together by using the connection functions  $beg_{VSM}$  and  $end_{VSM}$  to an integrated LL-CPS. As a standard modeling language, the description information of every model element and relation element in VSM is standardly structured. It can be in textual and/or graphical form.

Definition 30

$$m_{VSM} := (ME_{VSM}, A_{VSM}, beg_{VSM}, end_{VSM})$$

$$beg_{VSM} := beg|_{A_{VSM} \rightarrow ME_{VSM}}$$

$$end_{VSM} := end|_{A_{VSM} \rightarrow ME_{VSM}}$$

$$A_{VSM} := ME_{VSM} \times ME_{VSM}$$

$$m_{VSM} \in M_{VSM}$$

In VSM, the model elements and relation elements will be continually specified by different types. The set of model elements  $ME_{VSM}$  comprises a set of process elements  $P_{VSM}$ , a set of flow elements  $F_{VSM}$  and a set of comment elements  $C_{VSM}$ . They are disjoint sets. The  $P_{VSM}$  comprises two disjoint sets  $P_{VSM}^{Process}$  and  $P_{VSM}^{Info}$ . The  $P_{VSM}^{Process}$  represents the set of model elements of production and logistic processes in VSM. The  $P_{VSM}^{Info}$  represents the set of model elements of data processing in VSM.

$$ME_{VSM} := \{P_{VSM} \dot{\cup} F_{VSM} \dot{\cup} C_{VSM}\}$$

$$P_{VSM} := \{P_{VSM}^{Process} \dot{\cup} P_{VSM}^{Info}\}$$

$$P_{VSM}^{Process} := \left\{ \begin{array}{l} P_{VSM}^{Process-process} \dot{\cup} P_{VSM}^{Process-stock} \\ \dot{\cup} P_{VSM}^{Process-shipment} \dot{\cup} P_{VSM}^{Process-CuSu} \end{array} \right\}$$

$$P_{VSM}^{Info} := \{P_{VSM}^{Info-EDV} \dot{\cup} P_{VSM}^{Info-control}\}$$

$$F_{VSM} := \{F_{VSM}^{EInfo} \dot{\cup} F_{VSM}^{MInfo} \dot{\cup} F_{VSM}^{Material}\}$$

$$C_{VSM} := \left\{ \begin{array}{l} C_{VSM}^{Databox} \dot{\cup} C_{VSM}^{Operator} \\ \dot{\cup} C_{VSM}^{Timeline-CVA} \dot{\cup} C_{VSM}^{Timeline-NVA} \dot{\cup} C_{VSM}^{Gosee} \end{array} \right\}$$

$$A_{VSM} := \{ A_{VSM}^{In} \dot{\cup} A_{VSM}^{Out} \dot{\cup} A_{VSM}^{Owned} \}$$

Every flow element in the VSM model is imaged with directed flow, where the arrow shows the flow direction. The set of flow elements  $F_{VSM}$  comprises the electronic information flows in set  $F_{VSM}^{EInfo}$ , the manual information flows in set  $F_{VSM}^{MInfo}$  and the material flows in set  $F_{VSM}^{Material}$ , and they are disjoint sets. The set of comment elements  $C_{VSM}$  comprises five disjoint sets by different types. The set  $C_{VSM}^{Databox}$  represents the comment elements in the type of data box. The elements in the set  $C_{VSM}^{Operator}$  make explanatory notes on the model element with the information about workers. The comment elements with type timeline CVA in the set  $C_{VSM}^{Timeline-CVA}$  are used to describe the value added times, as well as the elements in the set  $C_{VSM}^{Timeline-NVA}$  for the non-value added times (NVA). The set  $C_{VSM}^{Gosee}$  represents a set of comment elements with the type of go see. The set of relation elements  $A_{VSM}$  comprises a set of owned relation elements  $A_{VSM}^{Owned}$  and a set of the input relation elements  $A_{VSM}^{In}$  and a set of output relation elements  $A_{VSM}^{Out}$ . They are also disjoint sets.

There is a function to map the types of model elements and relation elements to strings, which describe the type of the corresponding model and relation elements.

## Definition 31

$$type := (ME_{VSM} \cup A_{VSM}) \rightarrow strings$$

$$\forall x \in (ME_{VSM} \cup A_{VSM}) \exists type(x) = \left\{ \begin{array}{l} \text{"Process"} \quad \text{if } x \in P_{VSM}^{Process-process} \\ \text{"Data box"} \quad \text{if } x \in C_{VSM}^{Databox} \\ \text{"Worker"} \quad \text{if } x \in C_{VSM}^{Operator} \\ \text{"EDV"} \quad \text{if } x \in P_{VSM}^{Info-EDV} \\ \text{"Production Control"} \quad \text{if } x \in P_{VSM}^{Info-control} \\ \text{"Inventory hedge"} \quad \text{if } x \in P_{VSM}^{Process-stock} \\ \text{"Timeline - NVA"} \quad \text{if } x \in C_{VSM}^{Timeline-NVA} \\ \text{"Timeline - CVA"} \quad \text{if } x \in C_{VSM}^{Timeline-CVA} \\ \text{"External Shipment"} \quad \text{if } x \in P_{VSM}^{Process-shipment} \\ \text{"Supplier"} \quad \text{if } x \in P_{VSM}^{Process-CuSu} \\ \text{"Go See"} \quad \text{if } x \in C_{VSM}^{Gosee} \\ \text{"Electronic Information arrow"} \quad \text{if } x \in F_{VSM}^{EInfo} \\ \text{"Manual Information arrow"} \quad \text{if } x \in F_{VSM}^{MInfo} \\ \text{"Material arrow"} \quad \text{if } x \in F_{VSM}^{Material} \\ \text{"Input"} \quad \text{if } x \in A_{VSM}^{In} \\ \text{"Output"} \quad \text{if } x \in A_{VSM}^{Out} \\ \text{"Owned relation"} \quad \text{if } x \in A_{VSM}^{Owned} \end{array} \right.$$

The IDs of every VSM model element and relation element are defined with an injective function  $id_{MVSM}$ . The ID of a relation element is represented with its beginning model element and target model element.

Definition 32

$$id_{MVSM} := (ME_{VSM} \cup A_{VSM}) \rightarrow IDs$$

$IDs := a\ set\ of\ strings$

#### 4.1.2.2 Metamodel

A VSM model  $m_{VSM}$  is formed with a metamodel in Figure 60, which describes not only the system structure, but also the specifications of a VSM model by types.

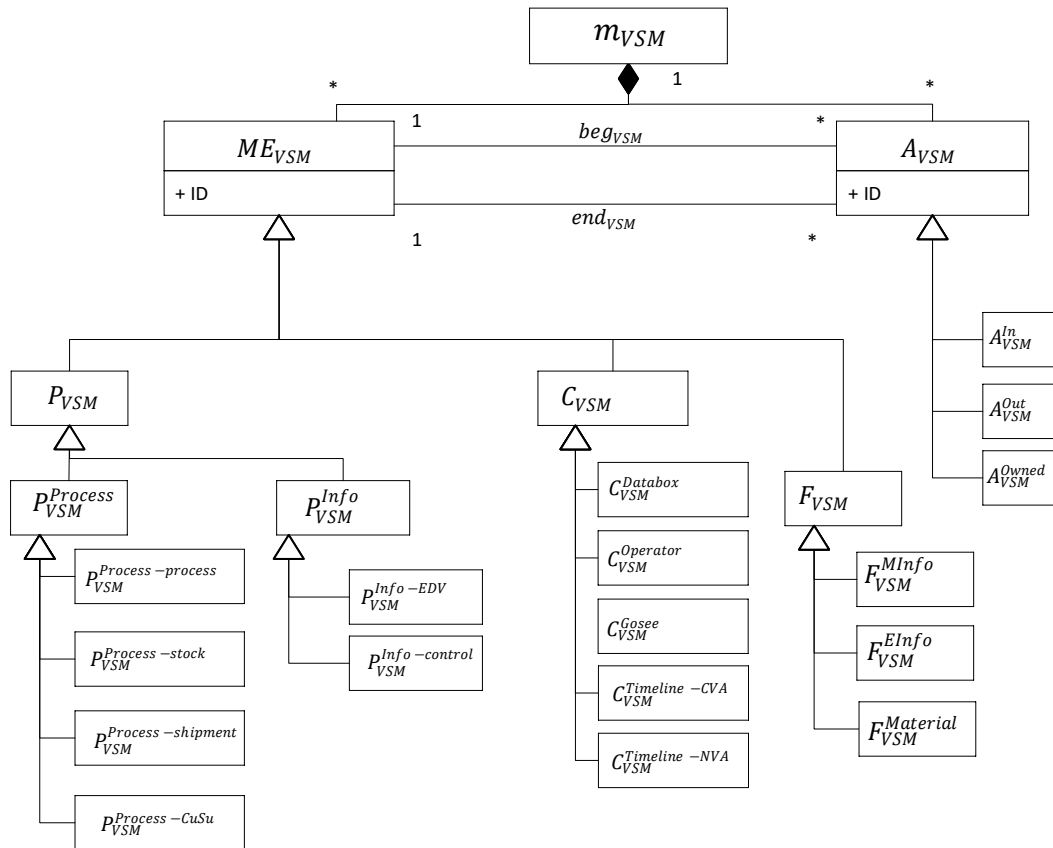


Figure 62: A Metamodel for specification of VSM model by types

The specified model elements and relation elements in any model  $m_{VSM}$  have to satisfy the combination rules forming with metamodels in Figure 63 and Figure 64 to guarantee the system integration. In this metamodel, the multiplicity describes the quantitative connection

relationship. It is interpreted as a subset of the natural numbers. The 1 is {1}, the \* is the set of all natural numbers.

One connection relation element has only two connected model elements: one is its source model element and the other is its target model element. Any process element can have any number of relation elements. Any flow element can have maximal two relation elements (see Figure 63).

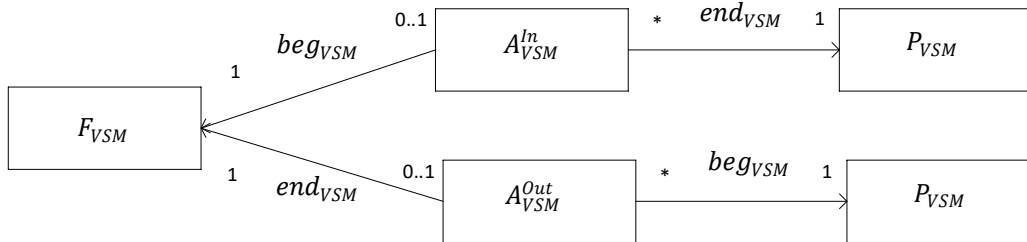


Figure 63: The Metamodel for connection relation in VSM model

An owned relation element in the set  $A_{VSM}^{Owned}$  links one comment element and its owner element together. One process or flow element can have any number of comment elements, although one comment element only allows belonging to one model element. The other connection or owned relations are not allowed.

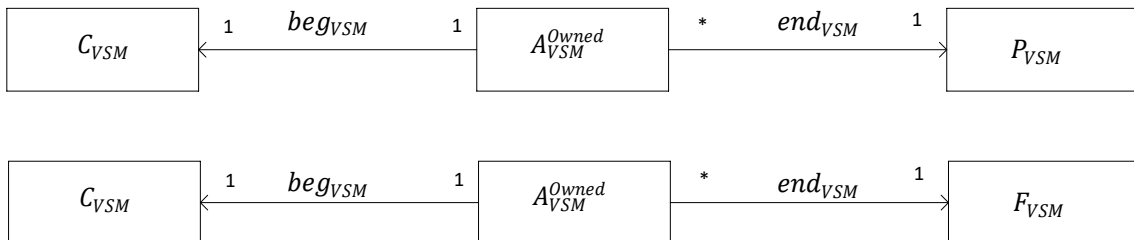


Figure 64: A Metamodel for owned relation in VSM model

#### 4.1.2.3 Graphical representation

The Table 2 shows the specified model elements and the relation elements in VSM by using the symbol, type, mathematical model element, metamodel element and descriptions.

Symbol	Type	Mathematical model element	Metamodel element	Descriptions
	Process	$P_{VSM}^{Process-process}$		A process, operation, machine or department.
	Data box	$C_{VSM}^{Databox}$		It goes under other model elements and describe the corresponding model element.



## Chapter 4 - Formal Descriptions and Transformations of Managed Evolution of LL-CPSs


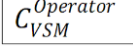

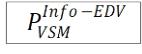

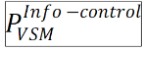

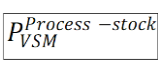

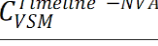

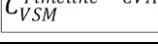

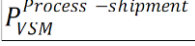

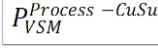

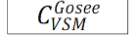

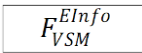

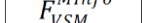

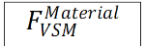

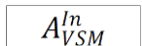

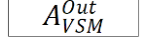
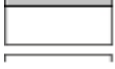
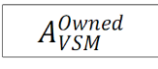
	Worker	$C_{VSM}^{Operator}$		It represents a worker and shows the number of workers required to process.
	EDV	$P_{VSM}^{Info-EDV}$		Electronic/digital information or data for production planning and production control.
	Production Control	$P_{VSM}^{Info-control}$		This box represents a central production scheduling or control department, person, system or operation.
	Inventory hedge	$P_{VSM}^{Process-stock}$		It represents a stock against problems such as downtime, to protect the system against sudden fluctuations in customer orders or system failures.
	Timeline-NVA	$C_{VSM}^{Timeline-NVA}$		It shows non-value added times (NVA).
	Timeline-CVA	$C_{VSM}^{Timeline-CVA}$		It shows value added times (CVA).
	External Shipment	$P_{VSM}^{Process-shipment}$		It represents the shipments from suppliers or to customers using external transport
	Supplier	$P_{VSM}^{Process-CuSu}$		It represents the supplier when in the upper left, and the customer when in the upper right, the usual end point for material
	Go See	$C_{VSM}^{Gosee}$		It represents the gathering of information through visual means.
	Electronic Information arrow	$F_{VSM}^{EInfo}$		It represents electronic flow.
	Manual Information arrow	$F_{VSM}^{MInfo}$		This arrow shows general flow of information from memos, reports, or conversation. Frequency and other notes may be relevant.
	Material arrow	$F_{VSM}^{Material}$		It represents the transportation of material from one process to the next process.
	Input	$A_{VSM}^{In}$		This represents the input relation between the model elements.
	Output	$A_{VSM}^{Out}$		This represents the output relation between the model elements.
	Owned relation	$A_{VSM}^{Owned}$		This represents the owned relation between the model elements.

Table 2. The graphical representation for the model-based description of VSM

#### 4.1.2.4 Example

The Figure 65 illustrates a VSM model  $m_a \in M_{VSM}$ , which comprises one process element, three flow elements and a number of comment elements. The process element sort links conveyer belt 1, conveyer belt 2 and info 1 together as an integrated LL-CPS.

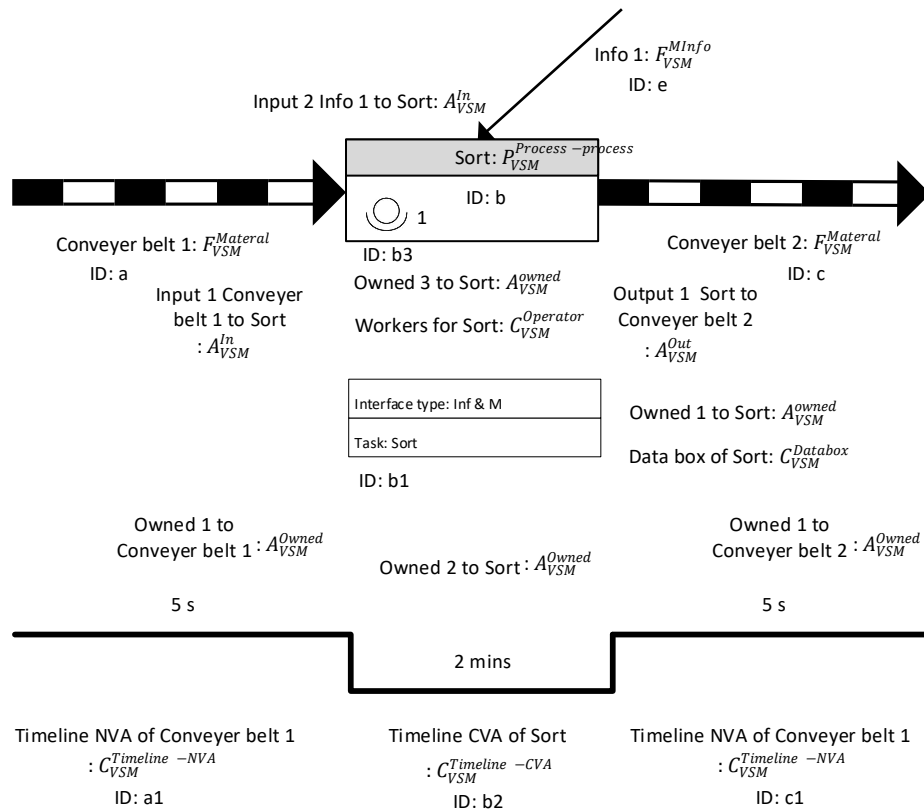


Figure 65: An example for the model-based description of a VSM model

### 4.1.3 Semantical mapping

The VSM graphs on the formal semantical foundation layer are understood as the equivalent descriptions of the VSM models on the model-based description layer. A semantical mapping represents the transformation from VSM models to the equivalent descriptions in the graph-structure. This transformation must keep that the VSM graphs are consistent with the VSM models.

#### 4.1.3.1 Mathematical foundation

A function  $semv$  is defined as a unidirectional bijective transformation function (see Definition 22): one-to-one (**injective**) and onto (**surjective**). This function represents a mapping relationship named semantical mapping from a set of models  $M_{VSM}$  to a set of graphs  $G_{VSM}$ .

## Definition 33

$$semv := M_{VSM} \rightarrow G_{VSM}$$

The function  $semv$  is an **equivalent mapping function** for a model  $m$  in  $M_{VSM}$  to a graph  $g$  in  $G_{VSM}$ , if it meet three requirements:

$$(\forall m_{VSM} \in M_{VSM}) \exists \{g_{VSM} \in G_{VSM} \mid semv(m_{VSM}) = g_{VSM}\}$$

1. Elements identify: For any model  $m_a$  in the set  $M_{VSM}$ , there exists a graph  $g_{VSM}$  in the set  $G_{VSM}$ , which satisfies the transformation function  $semv(m_{VSM}) = g_{VSM}$ . For any model element  $me$  in model  $m_{VSM}$ , exists a vertex  $v$ . The number of all model elements in  $m_{VSM}$  must be equal to the number of all vertices in  $g_{VSM}$ . The ID of vertex  $v$  is mapped to the ID of model element  $me$  with a mapping function:  $semvid := \{id\} \rightarrow \{id\}$ .

$$(\forall me \in ME_{VSM}) \exists (v \in V_{VSM}) \wedge |ME_{VSM}| = |V_{VSM}| \wedge semvid(id_{MVSM}(me)) = id_{GVSM}(v)$$

2. Structure identify: For any relation element  $a$  in  $m_{VSM}$ , there exists an edge  $e$  in  $g_{VSM}$ . The number of all relation elements in  $m_{VSM}$  must be equal to the number of all edges in  $g_{VSM}$ . The ID of the source vertex of  $e$  is mapped to the ID of the beginning model element of  $a$ , and the ID of the target vertex of  $e$  is mapped to the ID of the ending model element of  $a$ .

$$\begin{aligned} & (\forall a \in A_{VSM}) \exists (e \in E_{VSM}) \wedge |A_{VSM}| = |E_{VSM}| \\ & \wedge semvid(id_{MVSM}(beg_{VSM}(a))) = id_{GVSM}(src_{VSM}(e)) \\ & \wedge semvid(id_{MVSM}(end_{VSM}(a))) = id_{GVSM}(tgt_{VSM}(e)) \end{aligned}$$

3. Information identify: For any vertex or edge  $x$  in  $V_{VSM}$  and  $E_{VSM}$ , if its ID is mapped to the ID of a model or relation element  $y$ , the attribute of  $x$  is assigned with the string "Type" as its key and string  $type(y)$  in its value.

$$\begin{aligned} & (\forall y \in (ME_{VSM} \cup A_{VSM}) \exists x \in (V_{VSM} \cup E_{VSM}) \wedge semvid(id_{MVSM}(y)) = id_{GVSM}(x)) \\ & \rightarrow (attributes_{VSM}(x)("Type") = type(y)) \end{aligned}$$

If model and relation elements have other description information, they can be saved in the attributes of vertices or edges with the key-value structure too, for example the information of name. The function  $\delta$  is a mapping function to map the model and

relation elements to strings. A set of these functions is represented with  $\{\delta_i\}$ . A parameter  $s_i$  represents a string and it is combined with the  $\delta_i$  to a key-value structure according to the index  $i$  in that  $s_i$  as the key and  $\delta_i$  as the value.

$$(\forall y \in (ME_{VSM} \cup A_{VSM}) \exists x \in (V_{VSM} \cup E_{VSM}) \wedge semvid(id_{MVSM}(y)) = id_{GVSM}(x)) \rightarrow (attributes_{VSM}(x)(s_i) = \delta_i(y))$$

$$\delta_i := (ME_{VSM} \cup A_{VSM}) \rightarrow strings$$

$$s_i \in Key_{in}$$

$$\delta_i(y) \in Value_{in}$$

### 4.1.3.2 Graphical representation

Table 3 shows a comparison between the model-based description of VSM and formal semantical foundation of VSM by symbol, type and Metamodel element.

Model-based description of VSM			Formal semantical foundation of VSM		
Symbol	Type	Metamodel element	Sybol	Type	Metamodel element
	Process	$P_{VSM}^{Process-process}$		Vertex	$V_{VSM}$
	Data box	$C_{VSM}^{Databox}$		Vertex	$V_{VSM}$
	Worker	$C_{VSM}^{Operator}$		Vertex	$V_{VSM}$
	EDV	$P_{VSM}^{Info-EDV}$		Vertex	$V_{VSM}$
	Production Control	$P_{VSM}^{Info-control}$		Vertex	$V_{VSM}$
	Inventory hedge	$P_{VSM}^{Process-stock}$		Vertex	$V_{VSM}$
	Timeline-NVA	$C_{VSM}^{Timeline-NVA}$		Vertex	$V_{VSM}$
	Timeline-CVA	$C_{VSM}^{Timeline-CVA}$		Vertex	$V_{VSM}$
	External Shipment	$P_{VSM}^{Process-shipment}$		Vertex	$V_{VSM}$
	Supplier	$P_{VSM}^{Process-CuSu}$		Vertex	$V_{VSM}$
	Go See	$C_{VSM}^{Gosee}$		Vertex	$V_{VSM}$
	Electronic Information arrow	$F_{VSM}^{EInfo}$		Vertex	$V_{VSM}$
	Manual Information arrow	$F_{VSM}^{MInfo}$		Vertex	$V_{VSM}$

	Material arrow	$F_{VSM}^{Material}$		Vertex	$V_{VSM}$
	Input	$A_{VSM}^{In}$		Edge	$E_{VSM}$
	Output	$A_{VSM}^{Out}$		Edge	$E_{VSM}$
	Owned relation	$A_{VSM}^{Owned}$		Edge	$E_{VSM}$

Table 3. Comparison table for semantical mapping for VSM model

Figure 66 shows a graphical representation of the semantical mapping from a model in  $M_{VSM}$  to a graph in  $G_{VSM}$ .

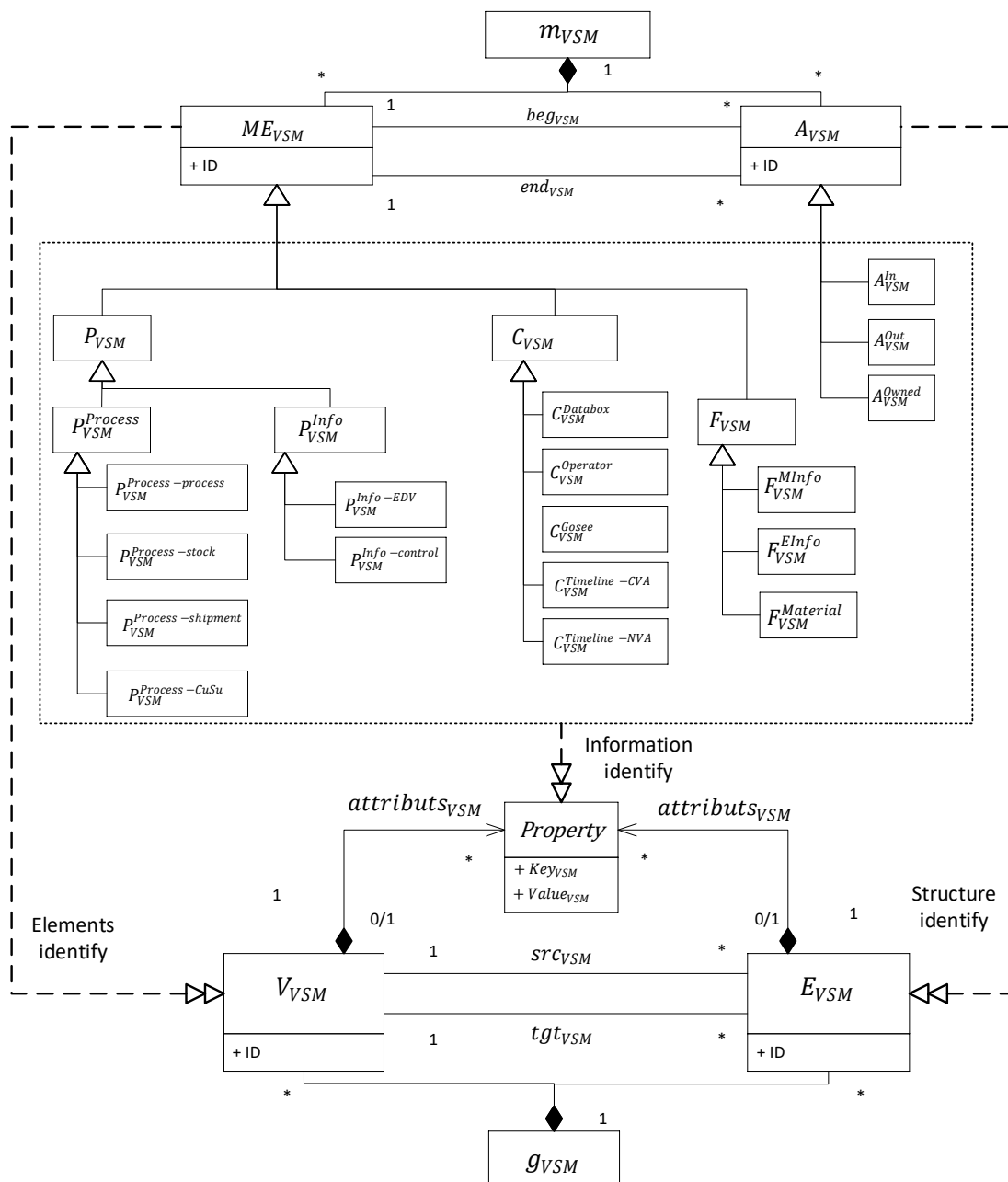


Figure 66: Graphical representation for semantical mapping of VSM model

### 4.1.3.3 Example

Figure 67 illustrates a semantical mapping transformation from a VSM model  $m_x \in M_{VSM}$  to a graph  $g_x \in G_{VSM}$ .

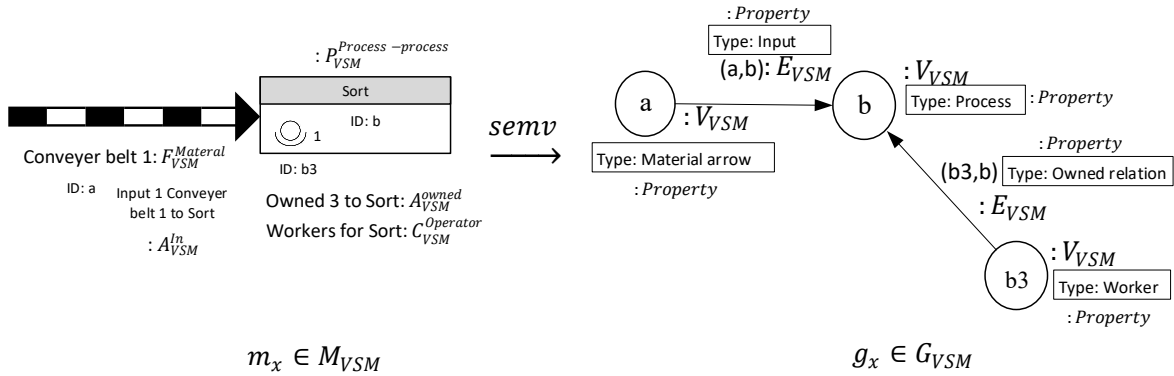


Figure 67: An example of semantical mapping for a VSM model

In Table 4, all model elements in  $m_x$  are transformed to vertices in  $g_x$ , and all relation elements are transformed into edges. The types of model elements are transformed to attributes in the corresponding vertices.

Conveyer belt 1 $\rightarrow v_a$ and $semvid(v_a) = \text{ID of Conveyer belt 1: a}$
Type of  for Conveyer belt 1 $\rightarrow$ (Type: Material arrow) in $v_a$
Process sort $\rightarrow v_b$ and $semvid(v_b) = \text{ID of Process sort: b}$
Type of  for process sort $\rightarrow$ (Type: Process) in $v_{b3}$
Worker of Sort $\rightarrow v_{b3}$ and ID of $v_{b3}$ is mapped to the ID of Worker of Sort: b3
Type of  in process sort $\rightarrow$ (Type: worker) in $v_{b3}$
Input 1 Conveyer belt 1 to sort $\rightarrow (v_a, v_b)$ and ID of $(v_a, v_b)$ is mapped to the ID of Input 1 Conveyer belt 1 to sort: (a,b)
Type of Input 1 Conveyer belt 1 to sort $\rightarrow$ (Type: Input) in $(v_a, v_b)$
Owned 3 to sort $\rightarrow (v_{b3}, v_b)$ and ID of $(v_{b3}, v_b)$ is mapped to the ID of Owned 3 to sort: (b3,b)
Type of Owned 3 to sort $\rightarrow$ (Type: Owned relation) in $(v_{b3}, v_b)$

Table 4. Transformations of model/relation elements in a VSM model

### 4.1.4 Concrete modeling

The concrete modeling is defined as a transformation function  $conv$  from a set of graphs  $G_{VSM}$  to a set of models  $M_{VSM}$ .

#### 4.1.4.1 Mathematical foundation

This function  $conv$  is a unidirectional **bijective** transformation function and instances every graph in a set  $G_{VSM}$  to VSM model in a set  $M_{VSM}$  (see Definition 22).

Definition 34

$$conv := G_{VSM} \rightarrow M_{VSM}$$

Like with the transformation function  $semv$ , the function  $conv$  is an **equivalent mapping function** if it meet three requirements: elements identify, structure identify and information identify.

$$(\forall g_{VSM} \in G_{VSM}) \exists \{m_{VSM} \in M_{VSM} \mid conv(g_{VSM}) = m_{VSM}\}$$

1. Elements identify: For any graph  $g_{VSM}$  in the set  $G_{VSM}$ , there exists a model  $m_{VSM}$  in the set  $M_{VSM}$ , which satisfies the transformation function  $conv(g_{VSM}) = m_{VSM}$ . For any vertex  $v$  in graph  $g_{VSM}$ , there exists a model element  $me$ . The number of all vertices in  $g_{VSM}$  must be equal to the number of all model elements in  $m_{VSM}$ . The ID of model element  $me$  must be mapped to the ID of vertex  $v$  with a mapping function:  $convid := \{id\} \rightarrow \{id\}$ .

$$(\forall v \in V_{VSM}) \exists (me \in ME_{VSM}) \wedge |V_{VSM}| = |ME_{VSM}| \\ \wedge convid(id_{GVSM}(v)) = id_{MVSM}(me)$$

2. Structure identify: For any edge  $e$  in  $E_{VSM}$ , there exists a relation element  $a$  in  $A_{VSM}$ . The number of all relation elements in  $m_{VSM}$  must be equal to the number of all edges in  $g_{VSM}$ . The ID of the beginning model element of  $a$  is mapped to the ID of the source vertex of  $e$ , and the ID of the ending model element of  $a$  is mapped to the ID of the target vertex of  $e$ .

$$(\forall e \in E_{VSM}) \exists (a \in A_{VSM}) \wedge |A_{VSM}| = |E_{VSM}| \wedge convid(id_{GVSM}(v)) = id_{MVSM}(me) \\ \wedge convid(id_{GVSM}(src_{VSM}(e))) = id_{MVSM}(beg_{VSM}(a)) \\ \wedge convid(id_{GVSM}(tgt_{VSM}(e))) = id_{MVSM}(end_{VSM}(a))$$

3. Information identify: For any model or relation element  $y$ , if its ID is mapped to the ID of a vertex or edge  $x$  in graph  $g_{VSM}$ , then the string in the attribute with the key "Type"

of this vertex or edge  $x$  will be reformed and assigned to the string of  $type(y)$  in the corresponding model or relation element  $y$ .

$$((\forall x \in V_{VSM} \cup E_{VSM} \exists y \in ME_{VSM} \cup A_{VSM}) \wedge convid(id_{GVSM}(x)) = id_{MVSM}(y)) \rightarrow (attributes_{VSM}(x)("Type") = type(y))$$

The other attributes in vertices and edges can be transformed to the description information of model and relation elements like the type information.

$$((\forall x \in V_{VSM} \cup E_{VSM} \exists y \in ME_{VSM} \cup A_{VSM}) \wedge convid(id_{GVSM}(x)) = id_{MVSM}(y)) \rightarrow (attributes_{VSM}(x)(s_i) = \delta_i(y))$$

$$\delta_i := (ME_{VSM} \cup A_{VSM}) \rightarrow strings$$

$$\delta_i(y) \in Value_{in}$$

$$s_i \in Key_{in}$$

#### 4.1.4.2 Graphical representation

Table 5 shows a graphical representation of concrete modeling, which can be understood as an inverse transformation of semantical mapping.


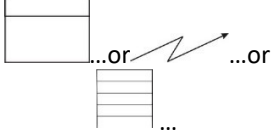

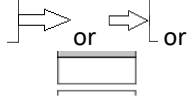
Formal semantical foundation of VSM			Model-based description of VSM		
Sybol	Type	Metamodel element	Sybol	Type	Metamodel element
	Vertex	$V_{VSM}$		Model element	$ME_{VSM}$
	Edge	$E_{VSM}$		Relation element	$A_{VSM}$

Table 5. Comparison table for concrete modeling for VSM graph

The graphical representation for concrete modeling in Figure 68 shows how a VSM graph is transformed to a VSM model.



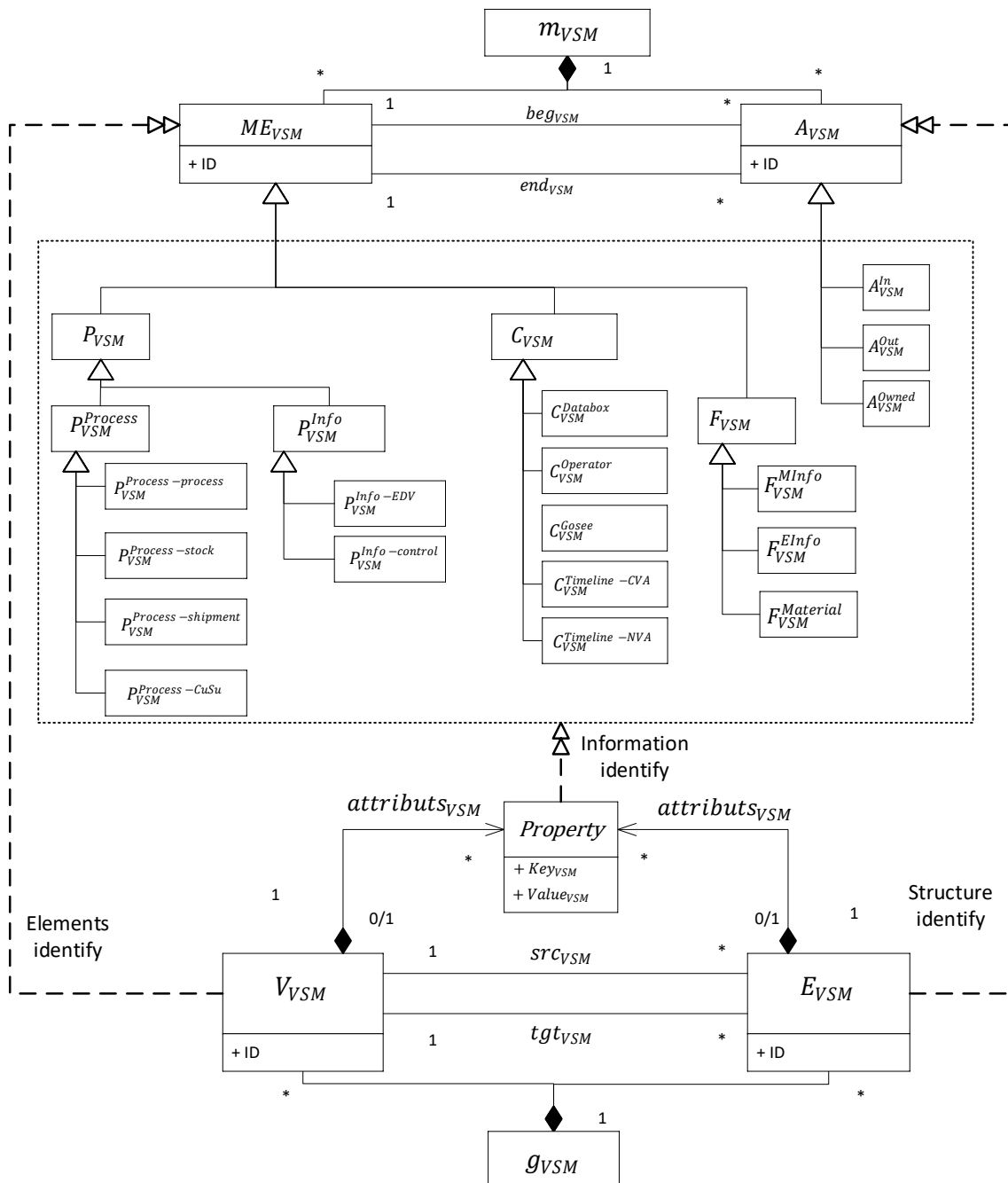


Figure 68: Graphical representation of concrete modeling for VSM graph

#### 4.1.4.3 Example

Figure 69 illustrates a concrete modeling transformation from graph  $g_x$  to model  $m_x$ .

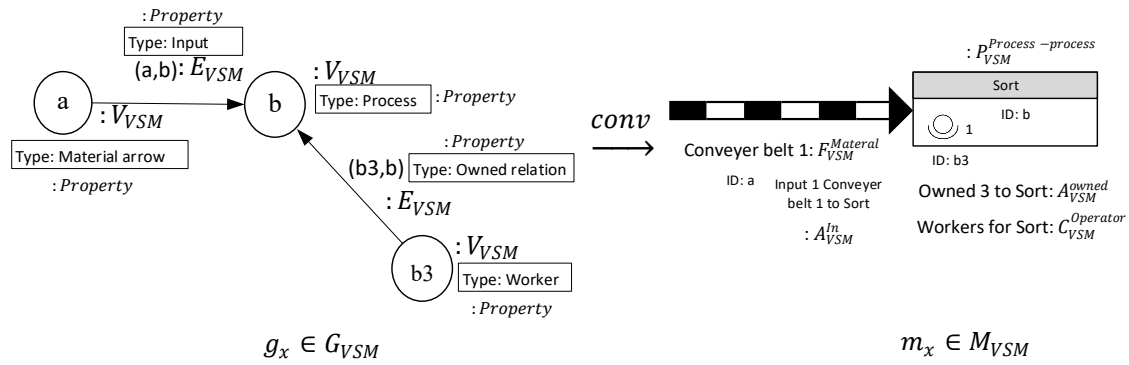


Figure 69: An example of concrete modeling for a VSM graph

## 4.2 Formal description for IBD

In this section, the IBD is used as a component-oriented modeling method to describe LL-CPSs. A set of IBD models  $M_{IBD}$  describes a set of LL-CPSs on model-based description layer. A set of graphs  $G_{IBD}$  represents the same LL-CPSs on the formal semantical foundation layer. The transformations between the models in  $M_{IBD}$  and the graphs in  $G_{IBD}$  are represented with the mapping functions *semi* and *coni* (See Figure 70).

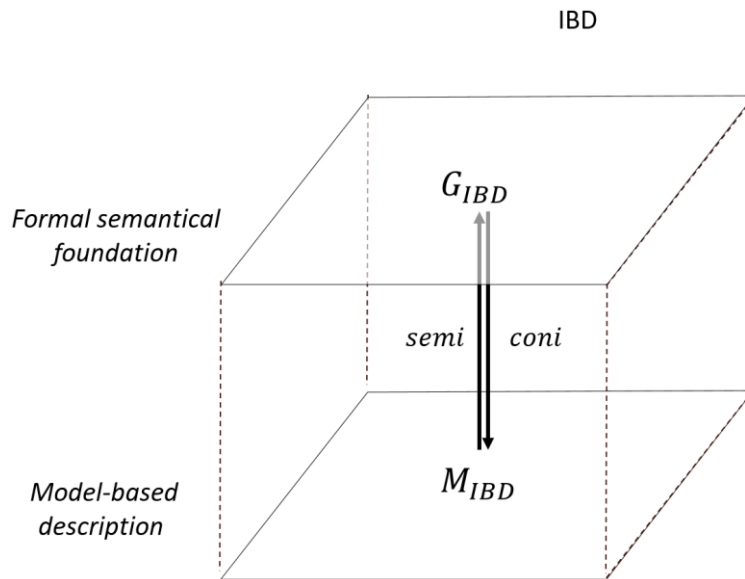


Figure 70: Formal descriptions and descriptions transformation for IBD model

### 4.2.1 Formal semantical foundation

The IBD graphs on the formal semantical foundation layer are introduced in this section by using of mathematical foundation, metamodel, graphical representation and example.

### 4.2.1.1 Mathematical foundation

Every graph  $g_{IBD}$  in the set  $G_{IBD}$  is defined as a **directed connected graph structure** (see Definition 4). It comprises a set of vertices  $V_{IBD}$  and a set of edges  $E_{IBD}$ . The functions  $src_{IBD}$  and  $tgt_{IBD}$  link every edge with its source vertex and target vertex together. Every vertex and edge have attributes that save the given information with a key-value structure, which comprises a set of string  $Key_{IBD}$  and a set of string  $Value_{IBD}$ . The mapping from the keys to values are represented with a hash function.

Definition 35

$$g_{IBD} := (V_{IBD}, E_{IBD}, src_{IBD}, tgt_{IBD}, attributes_{IBD}, Key_{IBD}, Value_{IBD})$$

$$src_{IBD} := src \mid_{E_{IBD} \rightarrow V_{IBD}}$$

$$tgt_{IBD} := tgt \mid_{E_{IBD} \rightarrow V_{IBD}}$$

$$E_{IBD} = V_{IBD} \times V_{IBD}$$

$$attributes_{IBD} := (V_{IBD} \cup E_{IBD}) \rightarrow (Key_{IBD} \rightarrow Value_{IBD})$$

$$Key_{IBD} := \text{a set of values}$$

$$Value_{IBD} := \text{a set of values}$$

$$g_{IBD} \in G_{IBD}$$

Every vertex and edge in  $g_{IBD}$  have a unique identifier (ID). The ID of an edge comprises the IDs of its source vertex and target vertex. An injective function  $id_{G_{IBD}}$  maps the identifiers of every vertex and edge to strings.

Definition 36

$$id_{G_{IBD}} := (V_{IBD} \cup E_{IBD}) \rightarrow IDs$$

$$IDs := \text{a set of strings}$$

### 4.2.1.2 Metamodel

Every vertex can connect with any number of edges, although one edge only allows connecting with one source vertex and one target vertex. The attributes in any vertex and edge are used to save the description information (see Figure 71).

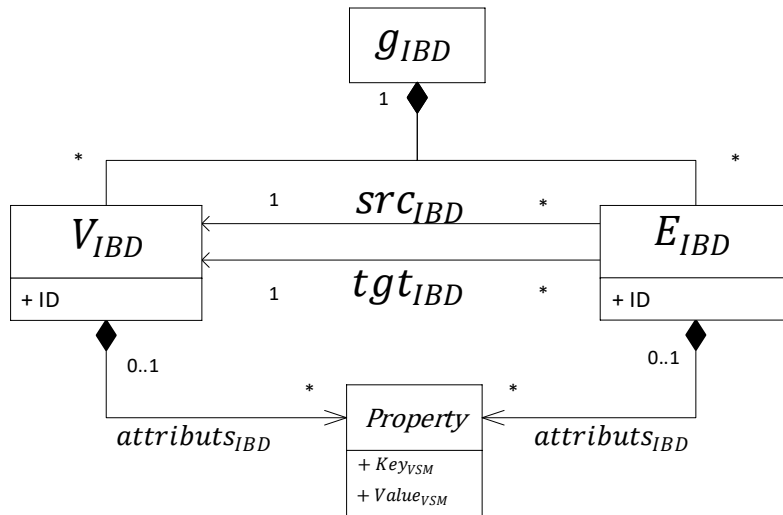


Figure 71: A Metamodel for formal semantical foundation of IBD model

### 4.2.1.3 Graphical representation

The graphical representation in Table 6 shows the symbol, name, mathematical model element, metamodel element and description of every graph elements in a IBD graph  $g_{IBD}$ .

Symbol	Type	Mathematical model element	Metamodel element	Descriptions
	Vertex	$v_{IBD} \in V_{IBD}$		Vertex in graph
	Edge	$e_{IBD} \in E_{IBD}$		Edge in graph
	Attributes	$Key_{IBD} \rightarrow Value_{IBD}$		Attributes of vertices and edges
	Connect relation begin	$src_{IBD}$		Connect relation for one edge and its source vertex
	Connect relation end	$tgt_{IBD}$		Connect relation for one edge and its target vertex

Table 6. Graphical representation of the formal semantical foundation of IBD model

### 4.2.1.4 Example

Figure 72 illustrates an IBD graph  $g_b \in G_{IBD}$ . One edge links two vertices together to a connected graph. The attributes of vertices and edges are used to save the description information.

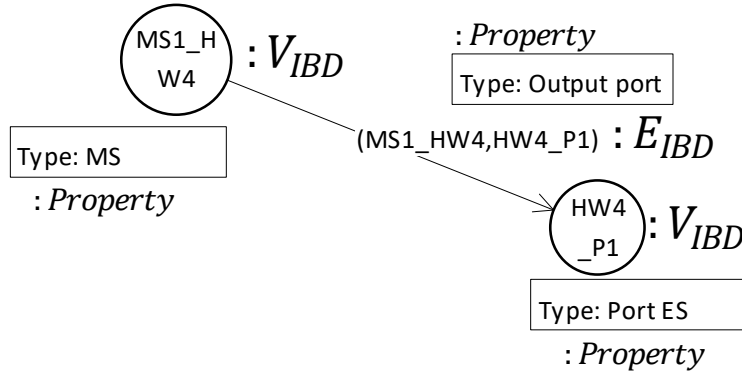


Figure 72: Example of the formal semantical foundation of IBD model

## 4.2.2 Model-based description

On the model-based description layer, a set of IBD models  $M_{IBD}$  model a set of LL-CPSs.

### 4.2.2.1 Mathematical foundation

Any IBD model  $m_{IBD} \in M_{IBD}$  is defined with **an integrated system in a network structure** (see Definition 19) and comprises a set of model elements  $ME_{IBD}$  and a set of relation elements  $A_{IBD}$ . The model elements comprise a set of blocks  $N_{IBD}$  and a set of ports (interfaces)  $P_{IBD}$ . The specifications of the blocks and ports (interfaces) have been introduced in sections 2.4.1.2 and 3.1.2. The model elements and relation elements connect to each other by using of the connection functions  $beg_{IBD}$  and  $end_{IBD}$  and build an integrated system.

Definition 37

$$m_{IBD} := (ME_{IBD}, A_{IBD}, beg_{IBD}, end_{IBD})$$

$$beg_{IBD} := beg \mid_{A_{IBD} \rightarrow ME_{IBD}}$$

$$end_{IBD} := end \mid_{A_{IBD} \rightarrow ME_{IBD}}$$

$$A_{IBD} := ME_{IBD} \times ME_{IBD}$$

$$m_{IBD} \in M_{IBD}$$

The model elements and relation elements in IBD model can be continually specified by different types. The set of model elements  $ME_{IBD}$  comprises a set block elements  $N_{IBD}$  and a set of port elements  $P_{IBD}$ . They are disjoint sets. The  $N_{IBD}$  comprises three disjoint sets: the set of blocks for information systems  $N_{IBD}^{IS}$ , the set of blocks for process systems  $N_{IBD}^{PS}$  and the set of relevant environment factors  $N_{IBD}^{RF}$ . The  $N_{IBD}^{IS}$  is specified by the types of software and hardware. The  $N_{IBD}^{PS}$  is specified into  $N_{IBD}^{CS}$  and  $N_{IBD}^{MS}$ . The  $N_{IBD}^{CS}$  is continually specified into

$N_{IBD}^{CS-HW}$  and  $N_{IBD}^{CS-SW}$ . The set of relevant environment factors  $N_{IBD}^{RF}$  is specified into two disjoint sets:  $N_{IBD}^{RF-H}$  and  $N_{IBD}^{RF-CPS}$ .

The interface specification for the set  $P_{IBD}$  is introduced in section 3.1.2.1. The  $P_{IBD}^{HMI}$  represents a set of HMI interfaces in an IBD model. The  $P_{IBD}^M$  represents a set of material interfaces. The  $P_{IBD}^{ES}$  represents a set of electrical signal interfaces. The  $P_{IBD}^C$  represents a set of constructive interfaces. The  $P_{IBD}^S$  represents a set of software interfaces. The  $P_{IBD}^E$  represents a set of execute interfaces.

The relation elements can be specified into three disjoint sets:  $A_{IBD}^{Con}$  for all of the connection relations between the port elements,  $A_{IBD}^{Input}$  and  $A_{IBD}^{Output}$  for the connection relations between port elements and block elements.

$$\begin{aligned}
 ME_{IBD} &:= \{ N_{IBD} \dot{\cup} P_{IBD} \} \\
 N_{IBD} &:= \{ N_{IBD}^{IS} \dot{\cup} N_{IBD}^{PS} \dot{\cup} N_{IBD}^{RF} \} \\
 N_{IBD}^{IS} &:= \{ N_{IBD}^{IS-HW} \dot{\cup} N_{IBD}^{IS-SW} \} \\
 N_{IBD}^{PS} &:= \{ N_{IBD}^{CS} \dot{\cup} N_{IBD}^{MS} \} \\
 N_{IBD}^{CS} &:= \{ N_{IBD}^{CS-HW} \dot{\cup} N_{IBD}^{CS-SW} \} \\
 N_{IBD}^{MS} &:= \{ N_{IBD}^{MS-HW} \} \\
 N_{IBD}^{RF} &:= \{ N_{IBD}^{RF-H} \dot{\cup} N_{IBD}^{RF-CPS} \} \\
 P_{IBD} &:= \{ P_{IBD}^{HMI} \dot{\cup} P_{IBD}^M \dot{\cup} P_{IBD}^{ES} \dot{\cup} P_{IBD}^C \dot{\cup} P_{IBD}^S \dot{\cup} P_{IBD}^E \} \\
 A_{IBD} &:= \{ A_{IBD}^{Con} \dot{\cup} A_{IBD}^{Input} \dot{\cup} A_{IBD}^{Output} \}
 \end{aligned}$$

The identifiers of every IBD model element and relation element are defined with an injective function  $id_{MIBD}$ . The ID of a relation element is represented with its beginning model element and target model element.

**Definition 38**

$$id_{MIBD} := (ME_{IBD} \cup A_{IBD}) \rightarrow IDs$$

$IDs := a \text{ set of strings}$

There is a function *category* to map the types of model elements and relation elements in IBD model to strings, which characterize the type of the corresponding model and relation elements.

Definition 39

$$category := (ME_{IBD} \cup A_{IBD}) \rightarrow strings$$

$$\forall x \in (ME_{IBD} \cup A_{IBD}) \exists category(x) = \begin{cases} "IS - HW" & \text{if } x \in N_{IBD}^{IS-HW} \\ "IS - SW" & \text{if } x \in N_{IBD}^{IS-SW} \\ "CS - HW" & \text{if } x \in N_{IBD}^{CS-HW} \\ "CS - SW" & \text{if } x \in N_{IBD}^{CS-SW} \\ "MS - HW" & \text{if } x \in N_{IBD}^{MS-HW} \\ "RF - H" & \text{if } x \in N_{IBD}^{RF-H} \\ "RF - CPS" & \text{if } x \in N_{IBD}^{RF-CPS} \\ "Port HMI" & \text{if } x \in P_{IBD}^{HMI} \\ "Port E" & \text{if } x \in P_{IBD}^E \\ "Port ES" & \text{if } x \in P_{IBD}^{ES} \\ "Port C" & \text{if } x \in P_{IBD}^C \\ "Port S" & \text{if } x \in P_{IBD}^S \\ "Port M" & \text{if } x \in P_{IBD}^M \\ "Input port" & \text{if } x \in A_{IBD}^{Input} \\ "Output port" & \text{if } x \in A_{IBD}^{Output} \\ "Connection" & \text{if } x \in A_{IBD}^{Con} \end{cases}$$

4.2.2.2 Metamodel

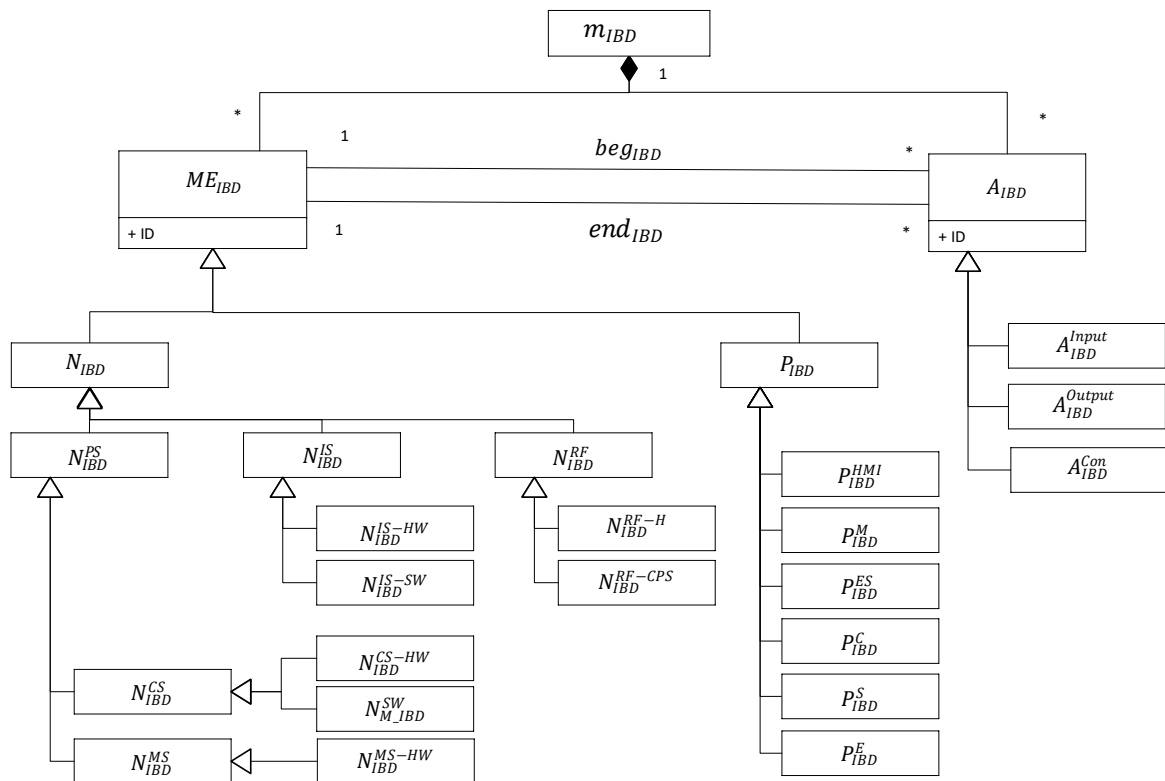


Figure 73: A Metamodel for IBD Model

The metamodel in Figure 73 describes the system structure of model  $m_{IBD}$ . Every model element must have an identifier and can have any number of relation elements, although every relation element only allows connecting with one model element as source and one as target. The ID of relation element is a tuple comprising the IDs from its beginning model element and ending model element.

The specified model elements and relation elements in any model  $m_{VSM}$  have to satisfy the combination rules forming with metamodels in Figure 74 to Figure 80, which are used to guarantee the system integration.

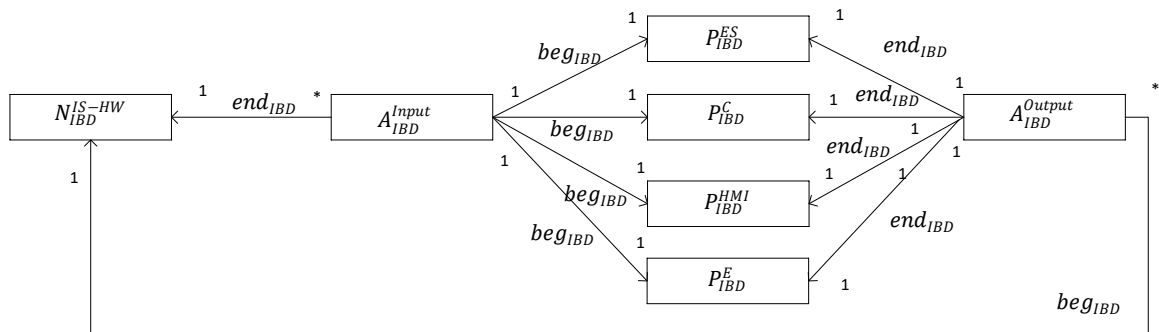


Figure 74: Combination rules between IS-HW model elements and ports

The block elements and port elements are connected to each other with the input and output relation elements. Every block element allows having any number of input and output port elements, although one port element only allows belonging to one block element.

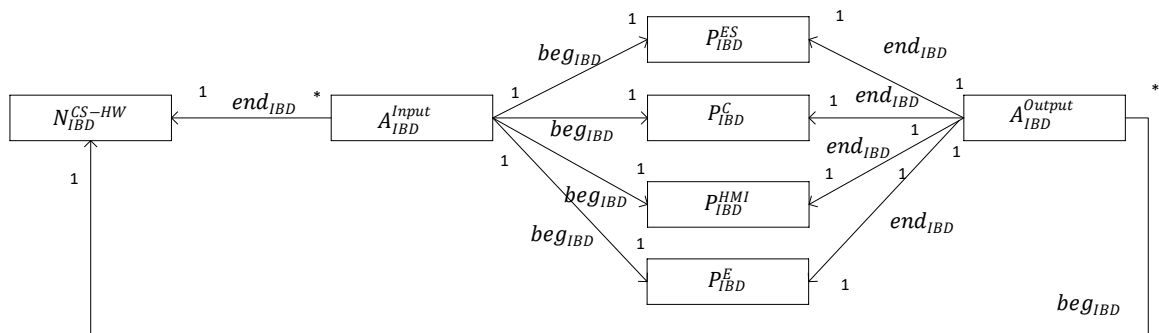


Figure 75: Combination rules between CS-HW model elements and ports

For example, Figure 74 and Figure 75 show the hardware blocks in the information system can have any number of ports by the types of HMI (HMI), electrical signal (ES), constructive (C) and execute (E), like the hardware blocks in the control system. The software blocks in the information system can have any number of ports by the types of software (S) and execute (E), like the software blocks in the control system that are shown with Figure 76 and Figure 77. The hardware blocks in the mechanic system can have any number of ports by the types of HMI (HMI), material flow (M), constructive (C) and execute (E), shown in Figure 78. The



human blocks can have any number of ports with HMI (HMI) type and material flow (M) type. The CPS block can have any number of ports in any type (see Figure 79).

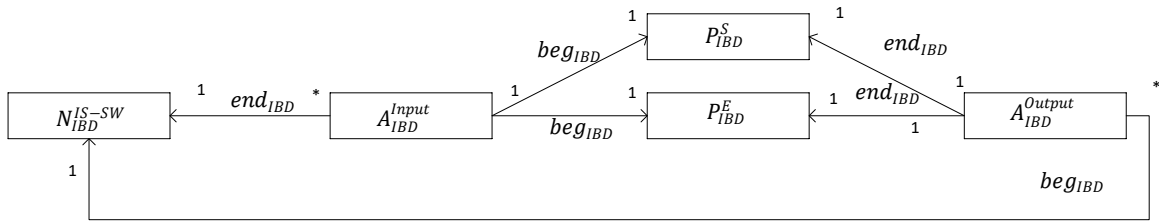


Figure 76: Combination rules between IS-SW model elements and ports

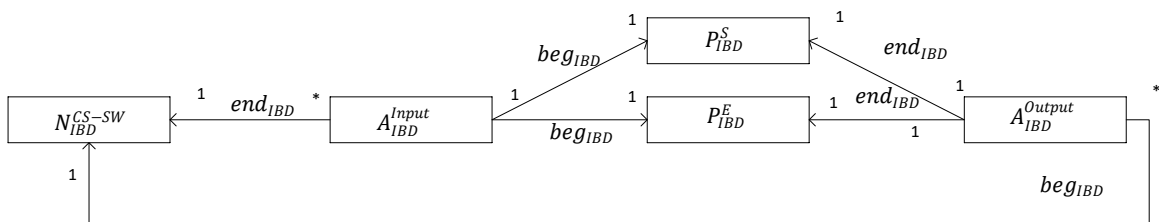


Figure 77: Combination rules between CS-SW model elements and ports

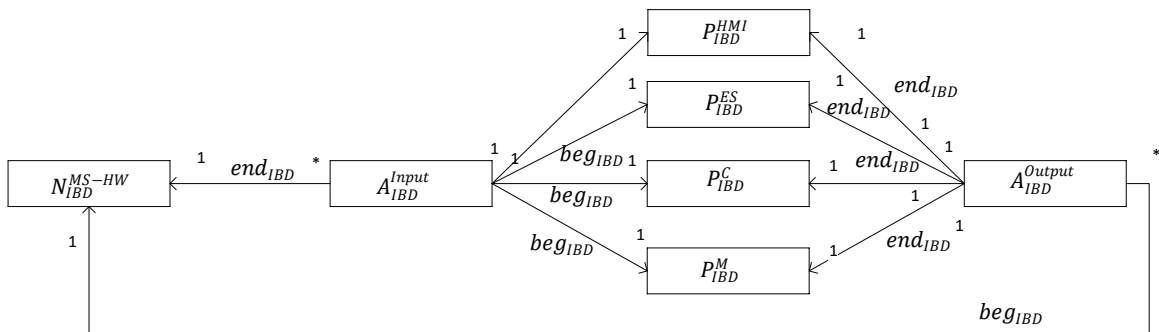


Figure 78: Combination rules between MS-HW model elements and ports

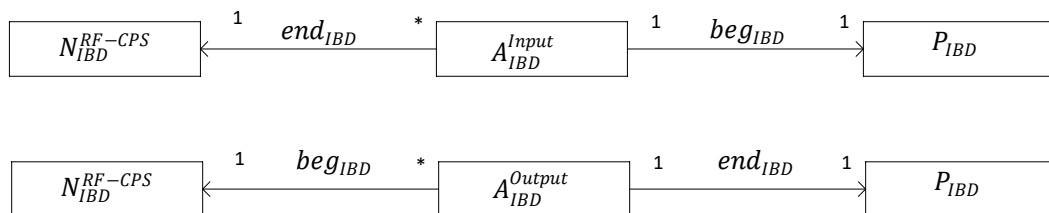


Figure 79: Combination rules between RF-CPS model elements and ports

Only two ports with the same type can be connected by using a connection relation element, and a connection relation element can just links two ports together (see Figure 80).

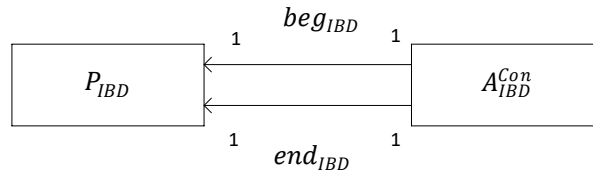


Figure 80: A Metamodel for connection relations between ports in IBD model

The other connection in IBD model are not allowed. Accordingly, that block and block cannot connect directly, port and port are not allowed to connect without connector to each other.

#### 4.2.2.3 Graphical representation

The symbol, type, mathematical model element, metamodel element and description for every model element in IBD model are shown in Table 7.

Symbol	Type	Mathematical model element	Metamodel element	Descriptions
	Block	$N_{IBD}$		It represents an entity, which is conceptual in nature during the initial phase of development but will be defined as part of the development process.
	Port	$P_{IBD}$		It is defined as the specified interaction point on a block.
	Connection	$A_{IBD}^{Con}$		It represents a connection relation.
	Input port	$A_{IBD}^{Input}$		It represents an input relation.
	Output port	$A_{IBD}^{Output}$		It represents an output relation.

Table 7. Graphical representation of the model elements in IBD model

#### 4.2.2.4 Example

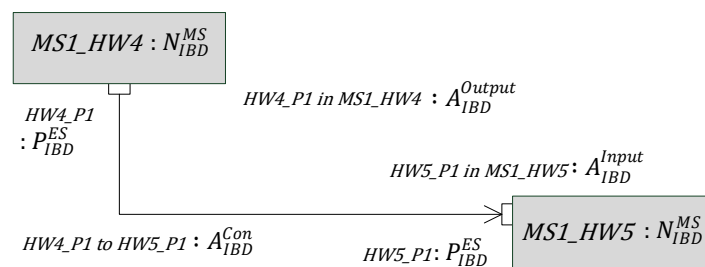


Figure 81: An example of IBD Model

Figure 81 illustrates one model in  $M_{IBD}$ . The block MS1\_HW4 has one port HW4\_P1, which connects to the port HW5\_P1 of the block MS1\_HW5.

### 4.2.3 Semantical mapping

The transformation function  $semi$  is defined as an unidirectional **bijective** transformation function from IBD models in  $M_{IBD}$  to IBD graphs in  $G_{IBD}$  (see Definition 22). This transformation function can be used for an equivalent transformation of a set of IBD models on the model-based description layer to a set of graphs on the formal semantical foundation layer.

#### 4.2.3.1 Mathematical foundation

The domain of function  $semi$  is a set of IBD models  $M_{IBD}$  and the codomain is a set of graphs  $G_{IBD}$ .

Definition 40


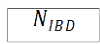

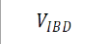

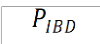


$$semi := M_{IBD} \rightarrow G_{IBD}$$

Like the transformation function  $semv$ , the function  $semi$  is an **equivalent mapping function** for a model  $m_{IBD}$  in  $M_{IBD}$  to a graph  $g_{IBD}$  in  $G_{IBD}$ , if it meet three requirements: elements identify, structure identify and information identify (see section 4.1.3.1). The domain and codomain of function are  $M_{IBD}$  and  $G_{IBD}$ .

$$(\forall m_{IBD} \in M_{IBD}) \exists \{g_{IBD} \in G_{IBD} \mid semi(m_{IBD}) = g_{IBD}\}$$

#### 4.2.3.2 Graphical representation

Table 8 shows a comparison of model/relation elements in an IBD model on the model-based description layer and on the formal semantical foundation layer.

Model-based description of IBD			Formal semantical foundation of IBD		
Symbol	Name	Metamodel element	Sybol	Name	Metamodel element
	Block			Vertex	
	Port			Vertex	







	Connection	$A_{IBD}^{Con}$		Edge	$V_{IBD}$
	Input	$A_{IBD}^{Input}$		Edge	$E_{IBD}$
	Output	$A_{IBD}^{Output}$		Edge	$E_{IBD}$

Table 8. Comparison table for semantical mapping of elements in an IBD model

Figure 82 shows a graphical representation for the semantical mapping from an IBD model  $m_{IBD}$  to an IBD graph  $g_{IBD}$ .

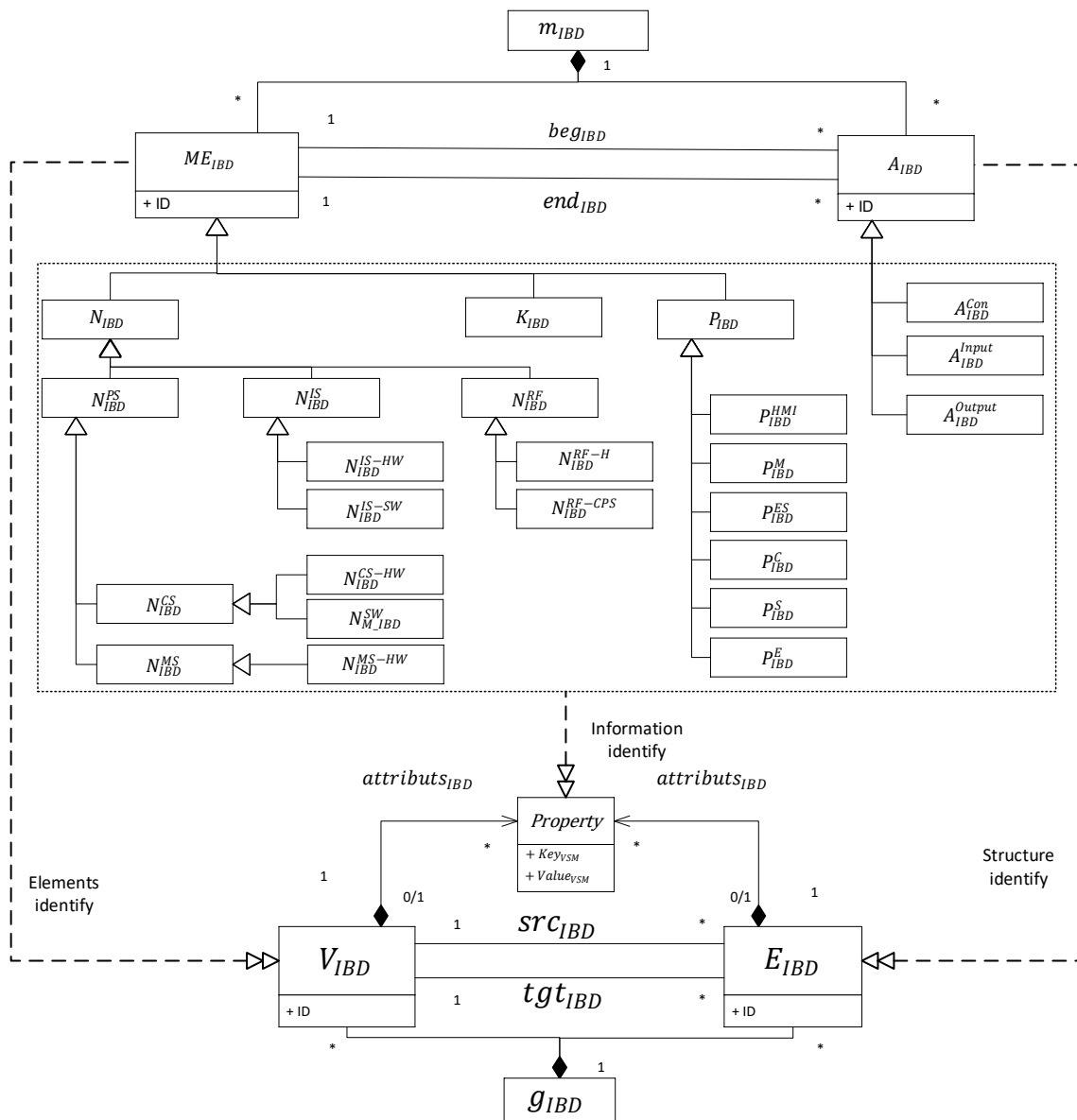


Figure 82: Graphical representation of the semantical mapping for IBD models

### 4.2.3.3 Example

Figure 83 illustrates an example of semantical mapping from one IBD model  $m_b \in M_{IBD}$  to a graph  $g_b \in G_{IBD}$ .

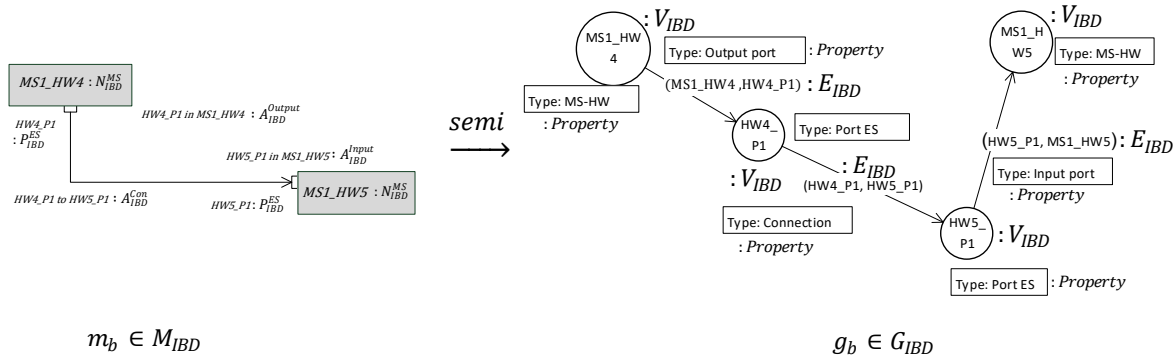


Figure 83: Example of semantical mapping of IBD

## 4.2.4 Concrete modeling

The concrete modeling is described with a unidirectional bijective transformation function *coni* (see Definition 22) from a set of graphs on the formal semantical foundation layer to a set of IBD models on the model-based description layer.

### 4.2.4.1 Mathematical foundation

The domain of function *coni* is a set of IBD graphs and its codomain is a set of IBD models.

Definition 41

$$coni := G_{IBD} \rightarrow M_{IBD}$$

The function *coni* is an **equivalent mapping function**, if it meet three requirements: elements identify, information identify and structure identify (see section 4.1.4.1).

$$(\forall g_{IBD} \in G_{IBD}) \exists \{m_{IBD} \in M_{IBD} \mid coni(g_{IBD}) = m_{IBD}\}$$

### 4.2.4.2 Graphical description

The elements comparing by using of the symbol, type and Metamodel element between an IBD graph on formal semantical foundation layer and an IBD model on model-based description layer is illustrated with the Table 9.

Formal semantical foundation of IBD			Model-based description of IBD		
Symbol	Type	Metamodel element	Symbol	Type	Metamodel element
	Vertex	$V_{IBD}$		Model element	$ME_{IBD}$
	Edge	$E_{IBD}$		Relation element	$A_{IBD}$

Table 9. Comparison table for concrete modeling of elements in an IBD graph

The graphical representation in Figure 84 shows, how an IBD graph is reverted to an IBD model by using concrete modeling.

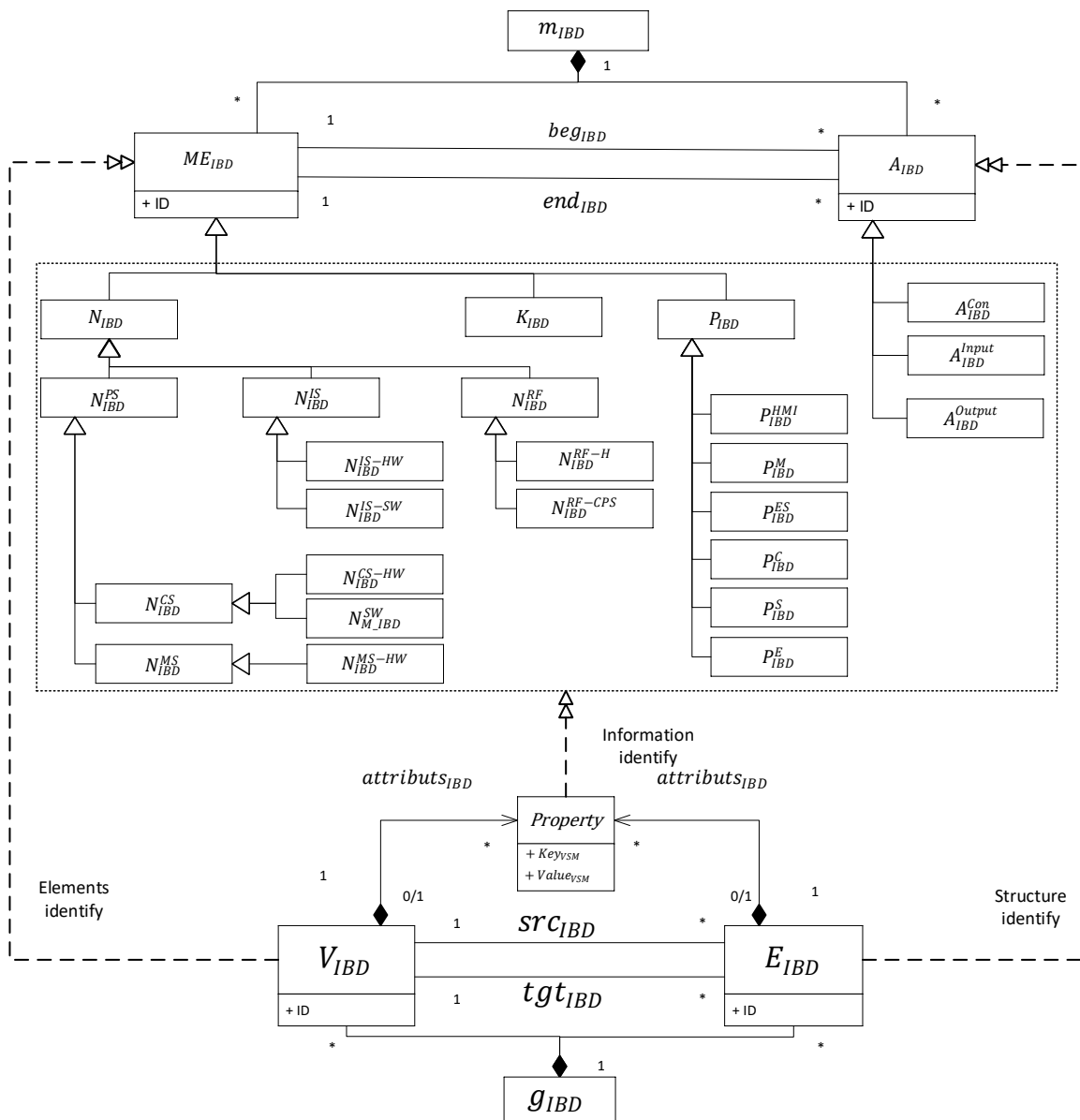


Figure 84: Graphical representation of the concrete modeling for IBD graphs

### 4.2.4.3 Example

Figure 85 illustrates a concrete modeling transformation from an IBD graph  $g_b$  to an IBD model  $m_b$ .

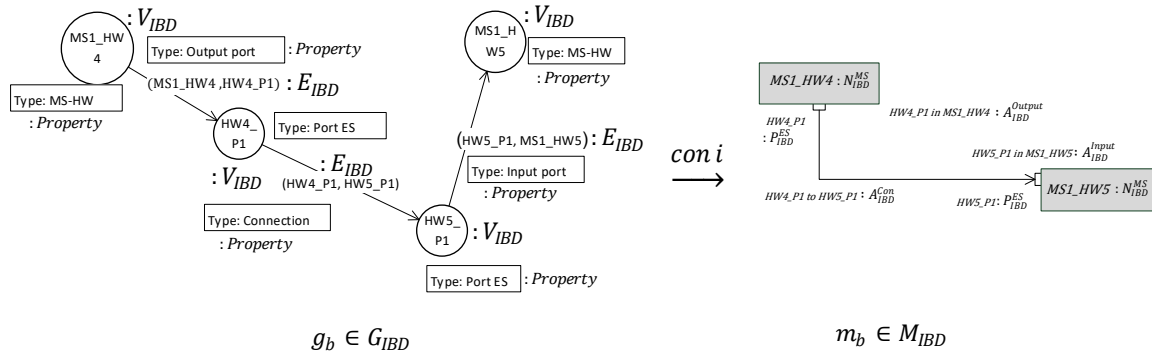


Figure 85: Example of concrete modeling of an IBD graph

## 4.3 Formal mapping relation from VSM to IBD

The mapping relationship from one VSM graph to one IBD graph is represented with a unidirectional mapping function  $h_i$  on the formal semantical foundation layer. On the model-based description layer, the mapping relationship from one VSM model to one IBD model is represented with a unidirectional mapping function  $hm_i$  (see Definition 23). A set of VSM graphs  $G_{VSM}$  is mapped to a set of IBD graphs  $G_{IBD}$  using a set of functions  $\{h_i\}$ , and the mappings for a set of VSM models  $M_{VSM}$  to a set of IBD models  $M_{IBD}$  is formed with a set of functions  $\{hm_i\}$  (See Figure 86).

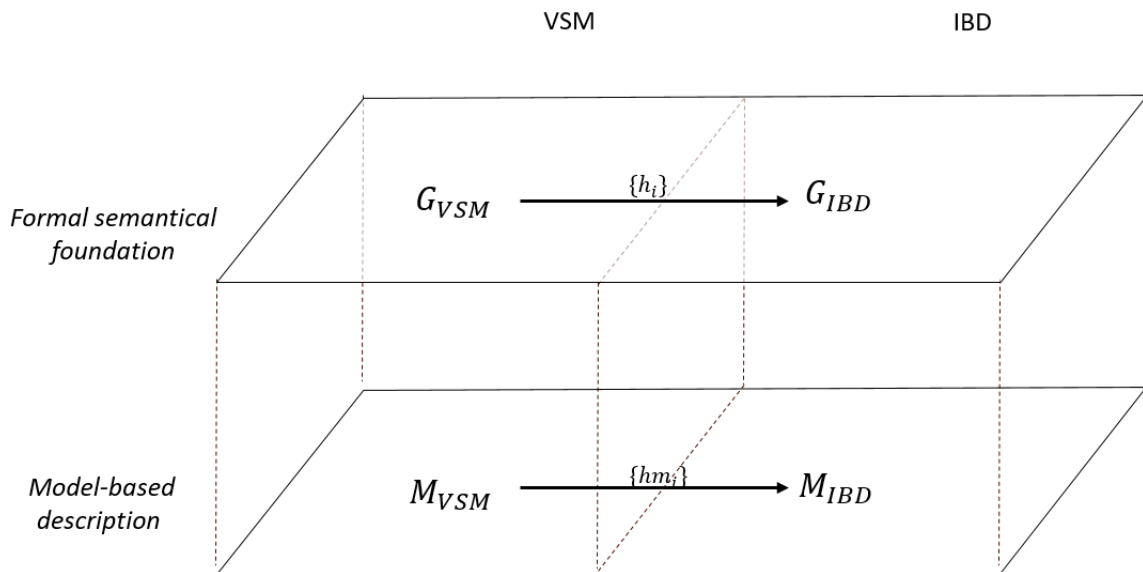


Figure 86: The mapping functions from VSM areas to IBD areas

The functions  $create_G$  and  $create_M$  represent that a VSM graph can be mapped with a set of functions  $\{h_i\}$  and a VSM model can be mapped with a set of functions  $\{hm_i\}$ .

Definition 42

$$create_G := G_{VSM} \rightarrow P(\{h_i\})$$

Definition 43

$$create_M := M_{VSM} \rightarrow P(\{hm_i\})$$

### 4.3.1 Formal semantical foundation

On the formal semantical foundation layer, a set of mapping functions  $h_i$  maps a set of VSM graphs to a set of IBD graphs.

#### 4.3.1.1 Mathematical foundation

Every mapping function  $h_i$  is a one-to-one mapping from a VSM graph to an IBD graph.

Definition 44

$$h_i := G_{VSM} \rightarrow G_{IBD}$$

In graph theory, the mapping relationship between graphs, in essence, is the mapping relationships between the vertex sets and edge sets in different graphs [57] [16]. If a VSM graph is mapped to a IBD graph with mapping function  $h_i$ , there exists a multivalued mapping function  $hv$ , for every vertex in this VSM graph. This function  $hv$  is a **one-to-many and surjective** mapping function.

Definition 45

$$hv := V_{VSM} \rightarrow \mathcal{P}(V_{IBD})$$

$$\forall h_i(g_y) = g_d \exists hv := V_y \rightarrow \mathcal{P}(V_d)$$

$$g_y \in G_{VSM}$$

$$g_d \in G_{IBD}$$

If a VSM graph  $g_y$  is mapped to an IBD graph  $g_d$ , the function  $hv$  maps every vertex in  $g_y$  to a set of vertices in  $g_d$ . A vertex  $v$  in a VSM graph  $g_y$  can be mapped with a set of multivalued mapping functions. Every mapping result comprises a set of vertices in  $g_d$ , where this set of these vertices must be a subgraph in  $g_d$  (see Definition 7).



$$(\forall v \in V_y) \exists \{v_j \in V_d \mid hv(v) = \{v_j\} \wedge g\{v_j\} \subseteq g_d\}$$

$$j = 1, \dots, n \quad n \in \mathbb{N}$$

One vertex  $v_j \in V_d$  can be mapped by different  $v$  in  $V_y$ . For instance, the vertex 8 in  $g_{d,1}$  in Figure 87 is mapped by the vertices a and c in  $g_y$ .

Because the mapping function  $pv$  in the definition of the path morphism in Definition 13 is also a mapping relationship from a set of vertices to a set of vertices. So the mapping function  $hv$  can be also used to in the path morphism and the walk morphism (see Definition 14).

$$pm(p(v_1, v_n)) := \{p_j(\{v_i\}, \{v_j\}) \mid p_j(v_i, v_j) \subseteq g_2\}$$

$$g_y \in G_{VSM} \text{ and } g_d \in G_{IBD}$$

$$p(v_1, v_n) \subseteq g_y$$

$$hv := V_{VSM} \rightarrow \mathcal{P}(V_{IBD})$$

$$hv(v_1) := \{v_i\} \quad v_i \in V_d$$

$$hv(v_n) := \{v_j\} \quad v_j \in V_d$$

$$i, j = 1, \dots, n \quad n \in \mathbb{N}$$

A **one-to-many** (multivalued) **and surjective** mapping function  $he$  maps edges in  $g_y$  to edges in  $g_d$ .

Definition 46

$$he := E_{VSM} \rightarrow \mathcal{P}(E_{IBD})$$

$$\forall h_i(g_y) = g_d \exists he := E_y \rightarrow \mathcal{P}(E_d)$$

Every edge in a VSM graph  $g_y$  is allowed to be mapped with a set of multivalued mapping functions  $he_i$ .

$$(\forall e \in E_y) \exists \{e_i \in E_d \mid he(e) = \{e_i\}\}$$

$$i = 1, \dots, n \quad n \in \mathbb{N}$$

### 4.3.1.2 Example

An example for the mapping relationship from a VSM graph  $g_y$  to an IBD graph  $g_{d,1}$  is illustrated in Figure 87. In order to clearly show the mapping relationships, all of the attributes of every vertex and edges are described as invisible.

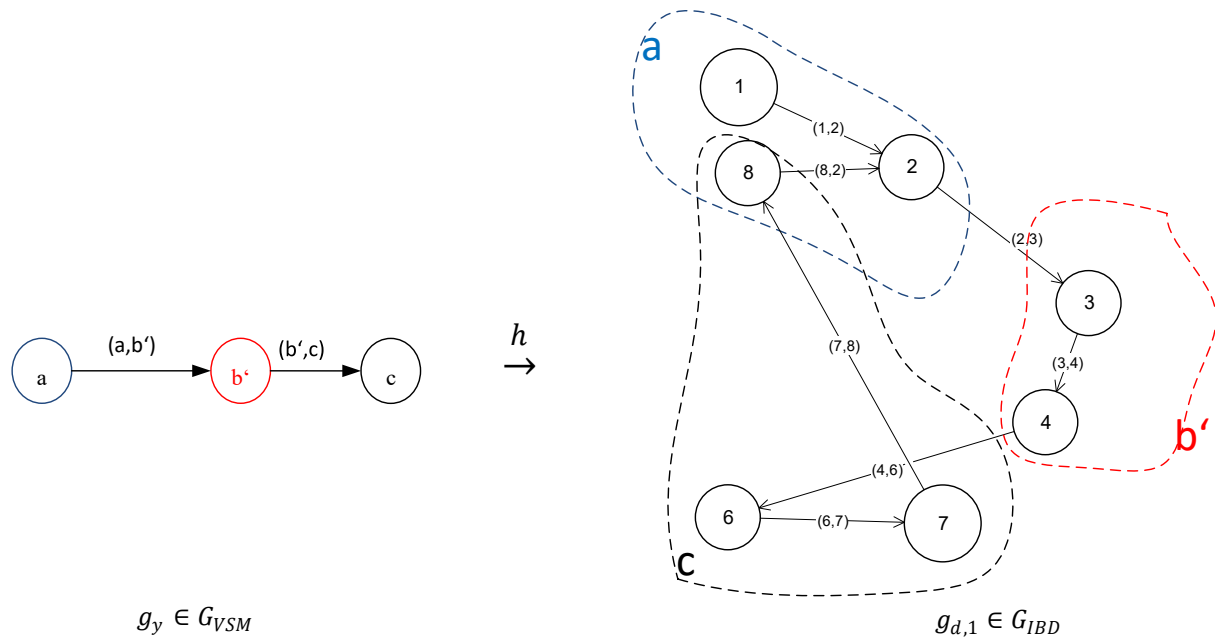


Figure 87: An example of mapping relationship  $h$

Table 10 shows the mapping relationships  $hv$  and  $he$  for every vertex and edge in graph  $g_y$  in this example. Vertex 8 is shared with a and c. Edge  $(b', c)$  maps to two edges  $(4,6)$  and  $(6,7)$  in IBD graph  $g_{d,1}$ .

$v \in V_{VSM}$	$hv(v)$
a	$\{v_1, v_2, v_8\}$
b'	$\{v_3, v_4\}$
c	$\{v_6, v_7, v_8\}$
$e \in E_{VSM}$	$he(e)$
$(a,b')$	$\{(2,3)\}$
$(b',c)$	$\{(4,6), (6,7)\}$

Table 10. Mapping relationship of every vertex and edge

### 4.3.2 Model-based description

On the model-based description layer, the mapping relationship from a VSM model to a IBD model is formed with a mapping function  $hm_i$ .

#### 4.3.2.1 Mathematical foundation

This  $hm_i$  is defined as a **one-to-one** mapping from a set of VSM models  $M_{VSM}$  to a set of IBD models  $M_{IBD}$ .

Definition 47

$$hm_i := M_{VSM} \rightarrow M_{IBD}$$

If a VSM model  $m_y$  is mapped to an IBD model  $m_d$  with mapping function  $hm_i$ , there is a **one-to-many and surjective** mapping function  $hmm$  to map every model elements in  $m_y$  to a set of model elements in  $m_d$ .

Definition 48

$$hmm := ME_{VSM} \rightarrow \mathcal{P}(ME_{IBD})$$

$$\forall hm_i(m_y) = m_d \exists hmm := ME_y \rightarrow \mathcal{P}(ME_d)$$

$$m_y \in M_{VSM}$$

$$m_d \in M_{IBD}$$

In this mapping, every model elements in this set must be connected as an integrated sub-model in model  $m_d$  (see Definition 20) and one model element  $me_i \in ME_d$  can be shared for using by different  $me$  in  $ME_y$ .

$$(\forall me \in ME_y) \exists \{me_i \in ME_d \mid hmm(me) = \{me_j\} \wedge m\{me_j\} \leq m_d\}$$

$$j = 1, \dots, n \quad n \in \mathbb{N}$$

A **one-to-many and surjective** mapping function  $hma$  maps any relation element in  $m_y$  to a set of relation elements in  $m_d$ .

Definition 49

$$hma := A_{VSM} \rightarrow \mathcal{P}(A_{IBD})$$

$$\forall hm_i(m_y) = m_d \exists hma := A_y \rightarrow \mathcal{P}(A_d)$$

Every relation element in a VSM model  $m_y$  is allowed to be mapped with a set of multivalued mapping functions  $hma_i$ . One relation element in  $m_d$  can be also shared by different relation elements in  $m_y$ .

$$(\forall a \in A_y) \exists \{a_i \in A_d \mid hma(a) = \{a_i\}\}$$

$$i = 1, \dots, n \quad n \in \mathbb{N}$$

### 4.3.2.2 Example

The Figure 88 illustrates the example for a mapping  $hm$  from a VSM model  $m_y$  to an IBD model  $m_d$ . In order to clearly represent the mapping relationships, the description information aside from the name in the VSM model is not displayed in all model elements.

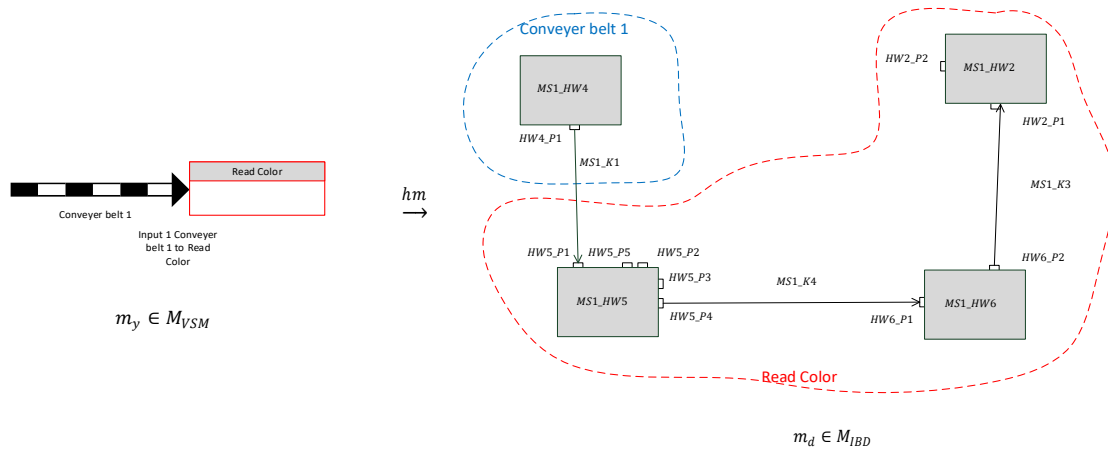


Figure 88: Example of mapping relationship  $hm$

In Table 11, every model element and relation element in  $m_y$  is mapped with functions  $hmm$  and  $hma$  to the model elements and relation elements in  $m_d$ .

$m_y$	$hmm(m_y)$
Conveyer belt 1	$\{MS1\_HW4, HW4\_P1, MS1\_K1\}$
Read color process	$\left\{ \begin{array}{l} MS1\_HW5, HW5\_P1, HW5\_P2 \\ HW5\_P3, HW5\_P4, HW5\_P5, \\ MS1\_K4, MS1\_HW6, HW6\_P1, HW6\_P2, \\ MS1\_K3, MS1\_HW2, HW2\_P1, HW2\_P2 \end{array} \right\}$
$a_y$	$hma(a_y)$
Input 1 Conveyer belt 1 to read color	$\{(MS1\_K1, HW5\_P1)\}$

Table 11. Mapping relationship of every model and relation element

## 4.4 Formal managed evolution of LL-CPSs

In this section, the formal descriptions for the managed evolution processes of LL-CPSs are introduced on the formal semantical foundation layer and model-based description layer. An ongoing LL-CPS is defined as the existing status of this system. The managed evolved status of this LL-CPS is defined as its targeted status. The managed evolutions of a set of LL-CPSs from existing statuses to targeted statuses are represented with a set of unidirectional evolution functions  $\{fm_i\}$  on the model-based description layer. On the formal semantical foundation layer, the managed evolutions are represented with a set of unidirectional functions  $\{f_i\}$  (See Figure 89).

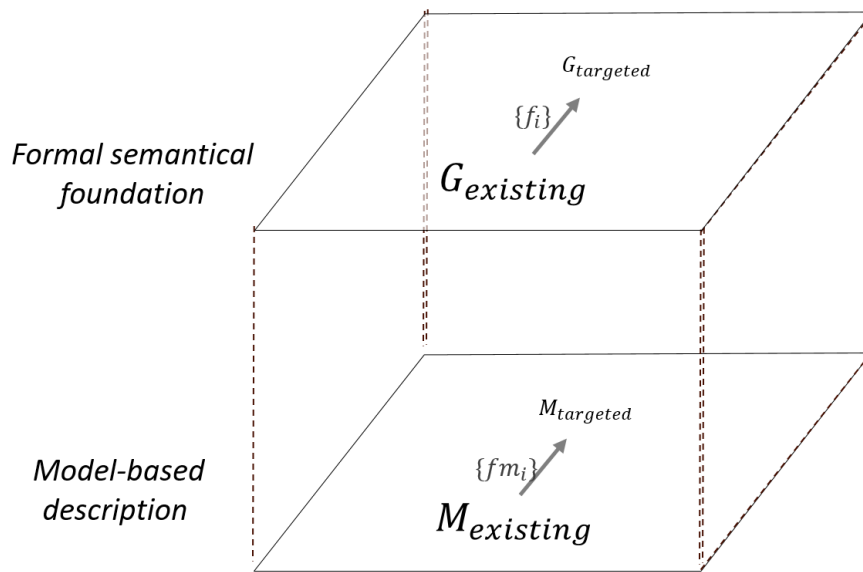


Figure 89: The managed evolution functions

### 4.4.1 Formal semantical foundation

On the formal semantical foundation layer, a set of evolution functions  $\{f_i\}$  maps a set of graphs  $G_{existing}$ , which represents a set of ongoing LL-CPSs, to a set of graphs  $G_{targeted}$  representing a set of targeted status of the ongoing LL-CPSs.

#### 4.4.1.1 Mathematical foundation

The evolution function  $f_i$  is a unidirectional function from a set of graphs  $G_{existing}$  to a set of graphs  $G_{targeted}$ . A set of these functions is represented by  $\{f_i\}$ .

Definition 50

$$f_i := G_{existing} \rightarrow G_{targeted}$$

$$G_{existing} \subset G_{in}$$

$$G_{targeted} \subset G_{in}$$

Every evolution function  $f_i$  can be represented with a system of linear equations, as below.

$$g_c \in G_{existing} \quad \text{and} \quad g_d \in G_{targeted}$$

$$f_i(g_c) = g_d$$

there is a system of linear equations:

$$g_d = g_c + G_c^\Delta$$

$$G_c^\Delta = G_c^+ - G_c^-$$

$$G_c^+ = \{g_c^{+i} \mid g_d - g_c\}$$

$$G_c^- = \{g_c^{-j} \mid g_c - g_d\}$$

$$g_c^{-j} \subseteq g_c \quad g_c^{+i} \subseteq g_d$$

$$\forall i, j = 1, \dots, n \quad n \in \mathbb{N}$$

**Operators Explanatory Notes:**

- $g_c + G_c^\Delta$  means the union graph of graph  $g_c$  and a set of graphs  $G_c^\Delta$ .
- $G_c^+ = \{g_c^{+i} \mid g_d - g_c\}$  is a set of subgraphs in  $g_d$ , which are different compared with  $g_c$ . Every subgraph must be a connected graph. For this reason, these subgraphs can include some vertices in  $g_c$ , which are directly connected with the vertices not in  $g_c$ . Accordingly, each subgraph  $g_c^{+i}$  comprises a set of the new added vertices and edges with the direct connected existing vertices and edges.
- $G_c^- = \{g_c^{-j} \mid g_c - g_d\}$  is a set of subgraphs in  $g_c$ , which are different compared with  $g_d$ . Every subgraph must be a connected graph, although it cannot include any vertex or edge in  $g_d$ . Therefore, each subgraph  $g_c^{-j}$  is composed of a set of the deleted vertices and edges.

**4.4.1.2 Example**

Figure 90 illustrates managed evolutions of an ongoing LL-CPS. A set of functions  $\{f_1, f_2\}$  represents the managed evolutions from this ongoing LL-CPS representing with graph  $g_c$  to two graphs  $g_{d,1}$  and  $g_{d,2}$ , where the targeted status of this ongoing LL-CPS is expressed with two different graphs  $g_{d,1}$  and  $g_{d,2}$  on the formal semantical foundation layer. The attributes of every vertex and edges are unseen to clearly express the managed evolution.

In this example, the graph  $g_{d,1}$  shows one possible reconstruction for the targeted status of the ongoing LL-CPS, where vertices 4 and 5 are new added. According to the operation rules,

the remaining vertex 3 has to be added to  $g_c^{+1}$ , in order to obtain the new added edge (3,4) and build into the graph structure.

For  $f_1$  there are:

$$G_c^+ = \{g_c^{+i} \mid g_{d,1} - g_c\} = \{g_c^{+1}(3,4,5)\}$$

$$G_c^- = \{g_c^{-j} \mid g_c - g_{d,1}\} = \emptyset$$

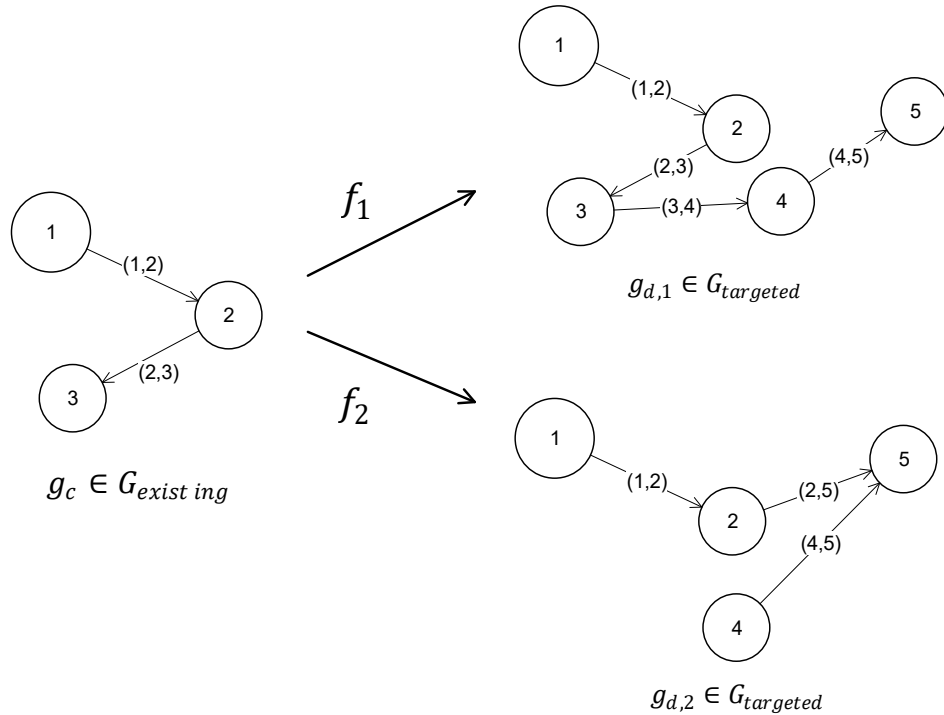


Figure 90: Example of two managed evolution functions on the formal semantical foundation layer

Graph  $g_{d,2}$  shows another evolved possibility for the targeted status of the ongoing LL-CPS, where vertices 4 and 5 are new added and vertex 3 is formed as a deleted vertex.

For  $f_2$  there are:

$$G_c^+ = \{g_c^{+i} \mid (g_{d,2} - g_c)\} = \{g_c^{+1}(2,4,5)\}$$

$$G_c^- = \{g_c^{-j} \mid g_c - g_{d,2}\} = \{g_c^{-1}(3)\}$$

## 4.4.2 Model-based description

On the model-based description layer, the existing LL-CPSs and their targeted statuses are formed with a set of models  $M_{existing}$  and a set of models  $M_{targeted}$ . The managed evolution relationships from  $M_{existing}$  to  $M_{targeted}$  are represented with a set of functions  $\{fm_i\}$ .

### 4.4.2.1 Mathematical foundation

A function  $fm_i$  is a unidirectional function from  $M_{existing}$  to  $M_{targeted}$ .

Definition 51

$$fm_i := M_{existing} \rightarrow M_{targeted}$$

$$M_{existing} \subset M$$

$$M_{targeted} \subset M$$

The linear equations below can be used to represent this evolution function  $fm_i$ .

$$m_c \in M_{existing} \quad \text{and} \quad m_d \in M_{targeted}$$

$$fm_i(m_c) = m_d$$

there is a system of linear equations:

$$m_d = m_c \oplus M_c^\Delta$$

$$M_c^\Delta = M_c^+ \ominus M_c^-$$

$$M_c^+ = \{m_c^{+i} \mid m_d \ominus m_c\}$$

$$M_c^- = \{m_c^{-j} \mid m_c \ominus m_d\}$$

$$m_c^{-j} \leq m_c \quad m_c^{+i} \leq m_d$$

$$\forall i, j = 1, \dots, n \quad n \in \mathbb{N}$$

**Operators Explanatory Notes:**

- $m_c \oplus M_c^\Delta$  means an integrated system of the model  $m_c$  with a set of models:  $M_c^\Delta$ , which represents all of changes during the managed evolution of  $m_c$ .
- $M_c^+ = \{m_c^{+i} \mid m_d \ominus m_c\}$  is a set of sub-models  $m_c^{+i}$  in model  $m_d$  compared with  $m_c$ . Every sub-model must be an integrated system and must include the elements in  $m_c$ , which are direct linked with the elements that are not in  $m_c$ . Accordingly, every sub-model is composed of a set of the new added elements and their direct connected existing elements.



- $M_c^- = \{m_c^{-i} \mid m_c \ominus m_d\}$  means a set of sub-models  $m_c^{-i}$  in model  $m_c$  compared with  $m_d$ . Every sub-model must be an integrated system, although it cannot include any element in  $m_c$ . Therefore, every sub-model is only composed with the deleted elements.

#### 4.4.2.2 Example

Figure 91 shows an example of a set of managed evolution functions  $\{fm_1, fm_2\}$ . A model  $m_c$  represents the existing status of an ongoing LL-CPS and the  $m_{d,1}$  and  $m_{d,2}$  express two different models for the targeted status of the same LL-CPS. In order to clearly represent the changes during managed evolution of LL-CPS, the other description information aside from the ID is not displayed in every model element.

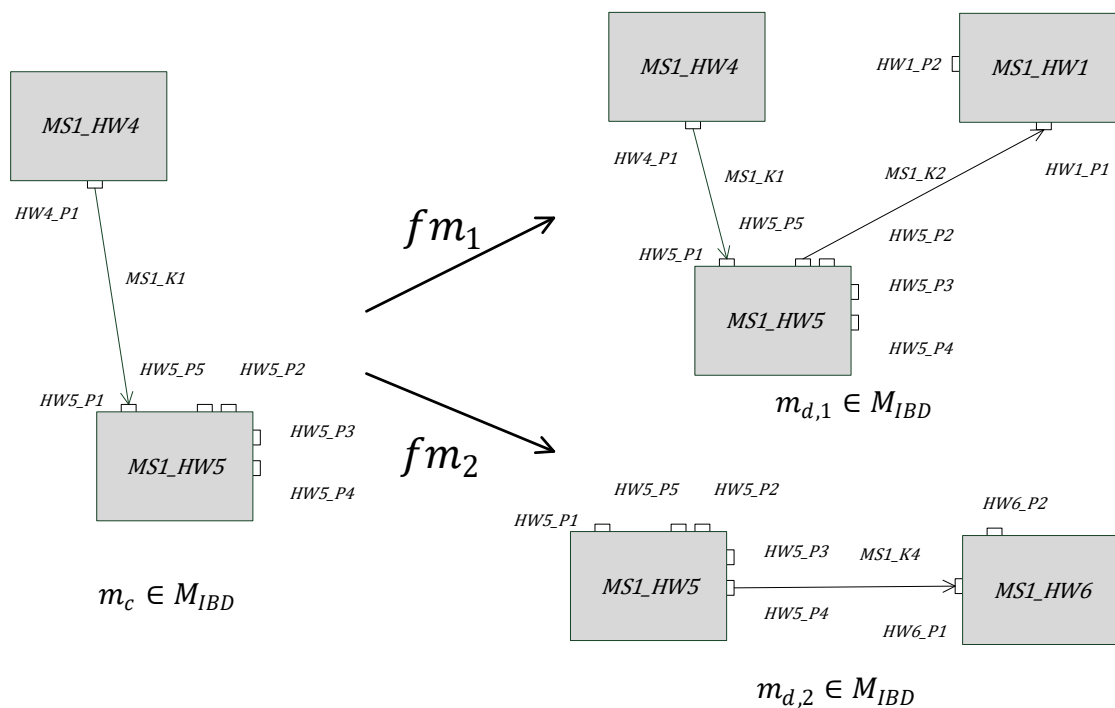


Figure 91: Example of two managed evolution functions on the model-based description layer

In this example, model  $m_{d,1}$  shows an evolved possibility for the targeted status of the ongoing LL-CPS, where the targeted status is implemented with a new system, which is integrated a new hardware MS1-HW1 on the existing model  $m_c$ .

For  $fm_1$  there are:

$$M_c^+ = \{m_c^{+i} \mid m_{d,1} \ominus m_c\} = \left\{ m_c^{+1} \left( \begin{array}{l} HW5\_P5, MS1\_K2, MS1\_HW1, \\ HW1\_P1, HW1\_P2 \end{array} \right) \right\}$$

$$M_c^- = \{m_c^{-j} \mid m_c \ominus m_{d,1}\} = \emptyset$$

The  $m_{d,2}$  shows another evolved possibility for the targeted status during the managed evolution of LL-CPS.

For  $fm_2$  there are:

$$M_c^+ = \{m_c^{+i} \mid m_{d,2} \ominus m_c\} = \left\{ m_c^{+1} \left( \begin{array}{c} HW5\_P4, MS1\_K4, MS1\_HW6, \\ HW6\_P1, HW6\_P2 \end{array} \right) \right\}$$

$$M_c^- = \{m_c^{-j} \mid m_c \ominus m_{d,2}\} = \left\{ m_c^{-1} \left( \begin{array}{c} MS1\_K1, MS1\_HW4, \\ HW4\_P1 \end{array} \right) \right\}$$

In  $m_{d,2}$ , the hardware MS1-HW5 with its ports is retained and connects with a new hardware MS1\_HW6 by using the ports HW5\_P4 and HW6\_P1. A connector MS1\_K4 links these two ports together. The hardware MS1\_HW4 with its port HW4\_P1 is deleted in the targeted status.

## 5 Solution Approach Overview

### Content

---

5.1 Problems formalization

5.2 Semantical mapping

5.3 Generating graph solutions

5.3.1 Reforming the mapping domain

5.3.2 Creating the mapping codomain

5.3.3 Path morphism

5.3.4 Graphs combination

5.4 Concrete modeling

5.5 Optimizing the model solutions

---

After the introduction of the formal descriptions and transformations of the managed evolution of LL-CPSs, the problems introduced in chapter 3 will be formalized in this chapter by using the formal descriptions and transformations. On this basis, an approach is introduced to solve the problems during the managed evolution of LL-CPS.

Section 5.1 introduces the formalization of the start position for the managed evolution of a LL-CPS by using the cube model introduced in chapter 4. In section 5.2, the semantical mapping functions will be used for the equivalent transformations of the models at the start position from the model-based description layer to the formal semantical foundation layer. Section 5.3 introduces how to generate a set of graph solutions on the formal semantical foundation layer, which are generated from the changes during the managed evolution of the LL-CPS. Every graph solution in this set represents one possible solution for the targeted status of the LL-CPS. In section 5.4, these graph solutions will be equivalently transformed from the formal semantical foundation layer to a set of models on the model-based description layer, whereon every model must satisfy the combination rules in modeling method and the execution sequences of the development requirements. The models, which cannot satisfy the combination rules and execution sequences will be deleted from this set.

In section 5.5, a model will be determined that represents the local optimal costs of system reconstruction during the managed evolution of this LL-CPS. Accordingly, the problems during managed evolution of LL-CPS are solved by using this approach.

Figure 92 shows the processes of the approach to solve the problems during the managed evolution of LL-CPS and the arrangement of sections in this chapter.

# Chapter 5 - Solution Approach Overview

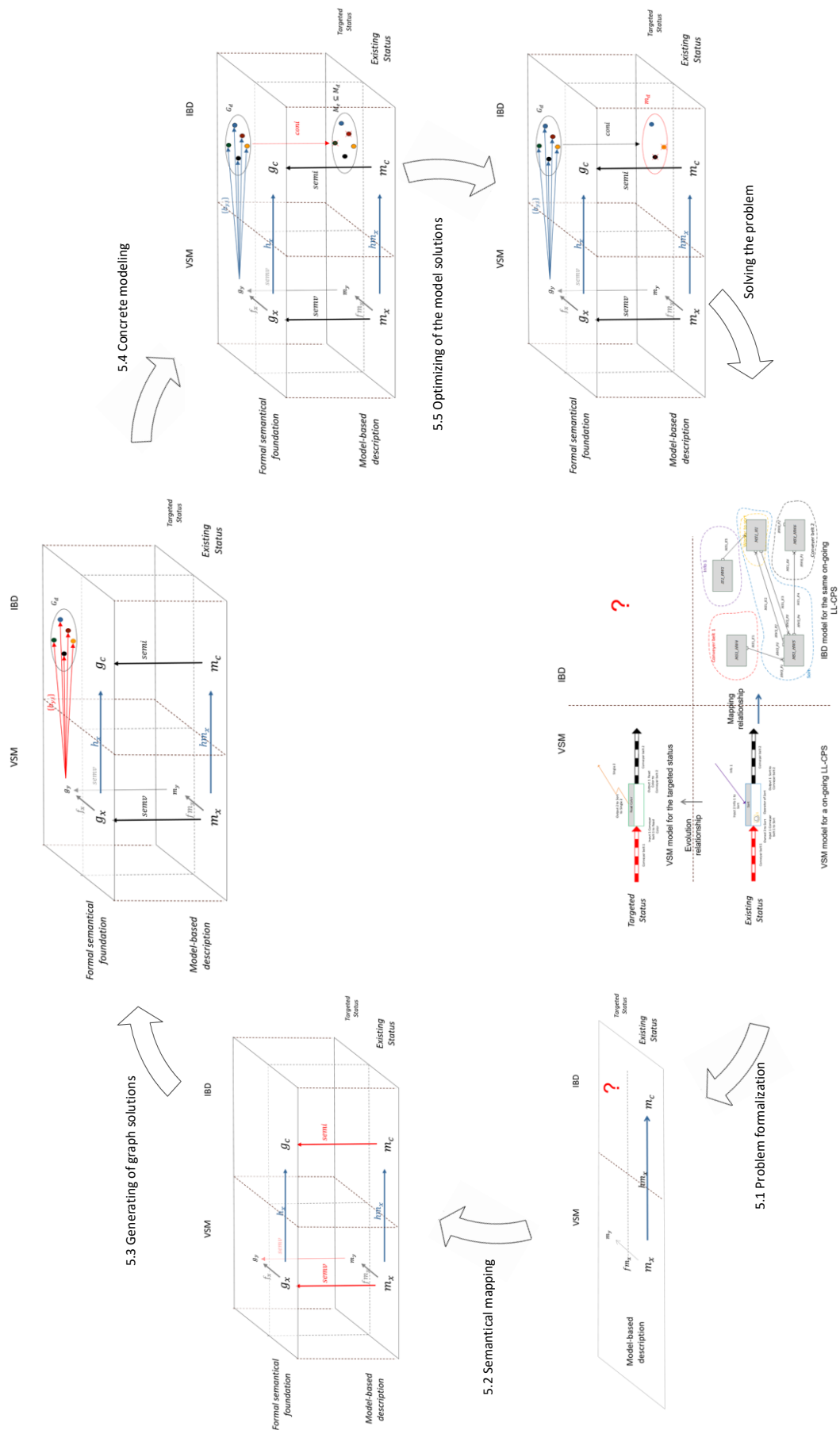


Figure 92: The processes of the approach

## 5.1 Problems formalization

Figure 93 shows the start position of the managed evolution of a LL-CPS, which is illustrated using the conveyor system with ASRS in chapter 3. In this example, the existing status of this LL-CPS named also as the ongoing LL-CPS is described concurrently with two modeling methods: VSM and IBD. There is an inherent mapping relationship from the elements in the VSM model to the elements in the IBD model, when both models describe the same LL-CPS. For instance, a conveyor belt with a RFID read/write sensor is modeled with a process element in the VSM model and modeled as a set of model components in the IBD model. Therefore, this process element is mapped to this set of model components from the VSM model to the IBD model.

The targeted status of this LL-CPS is clearly described with a VSM model. The managed evolution from the existing status to the targeted status of this LL-CPS is defined with an evolution relationship from one VSM model to another. However, in this start position the IBD model, which describes the targeted status of this LL-CPS, is unknown. The mapping relationships for the new added model elements and relation elements in the VSM model describing the targeted status are unknown. For instance, the new added process element “read color” in the VSM model for the targeted status does not have any mapping relationship to the model elements in the IBD model (see Figure 93). The evolution relationship for the existing IBD model is unknown too.

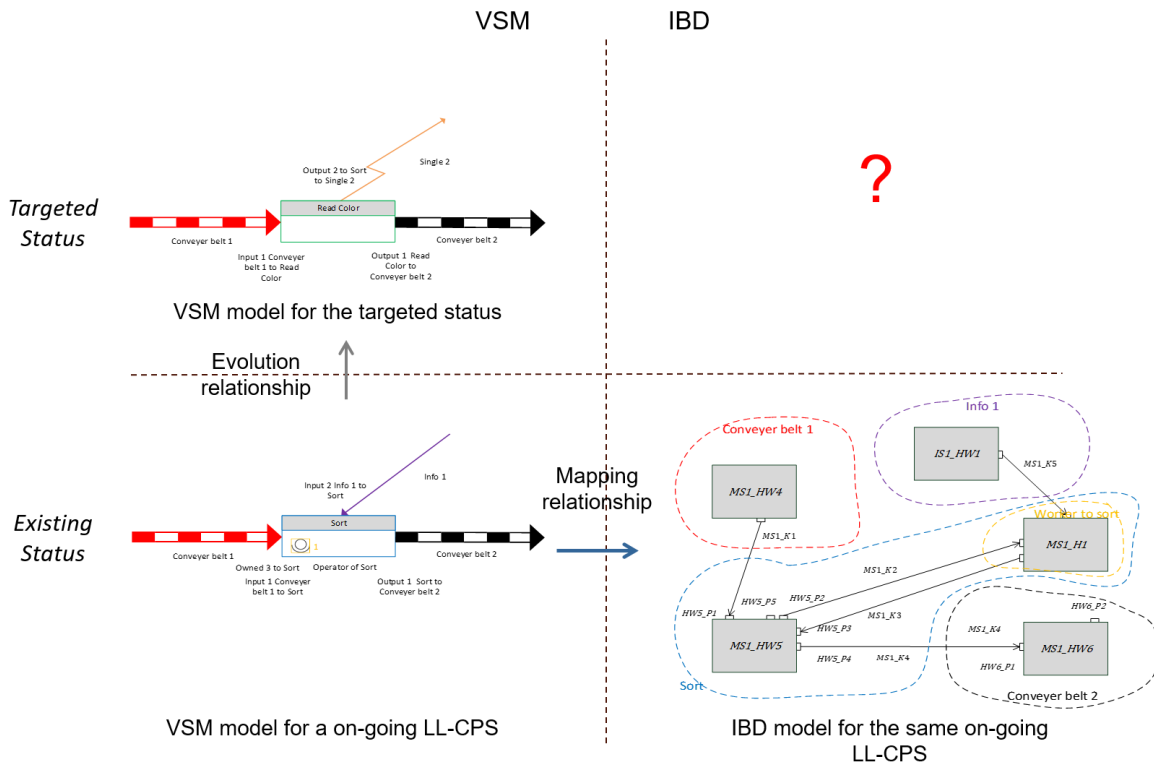


Figure 93: Example of the start position of the managed evolution of a LL-CPS

## Chapter 5 - Solution Approach Overview

The objective of this approach is to generate a new IBD model that represents the targeted status of this LL-CPS as same as the VSM model. At the same time, the managed evolution from the existing IBD model to the new IBD model must offer a local optimal costs of system reconstruction and controlled risks for system development.

This start position is formalized with the cube model introduced in chapter 4. The three models, one mapping relationship and one evolution relationship in Figure 93 are all laid flat on the model-based description layer in the cube model. The VSM model for the ongoing LL-CPS is formalized with a model  $m_x \in M_{VSM,existed}$  and the IBD model for the same LL-CPS is formalized with a model  $m_c \in M_{IBD,existed}$ . These two models are known.

The mapping relationship from  $m_x$  to  $m_c$  is defined with a function  $hm_x$ . The mapping relationships for the model elements and relation elements from model  $m_x$  to model  $m_c$  are represented with functions  $hmm_x$  and  $hma_x$ . These functions are also known.

The targeted status of this LL-CPS is described with a VSM model  $m_y \in M_{VSM,targeted}$ . The evolution relationship from  $m_x$  to  $m_y$  is formalized with a function  $fm_x$ . The model  $m_y$  and the function  $fm_x$  are known (see Figure 94).

There are

$$m_x \in M_{VSM,existing}$$

$$m_y \in M_{VSM,targeted}$$

$$m_c \in M_{IBD,existing}$$

and

$$fm_x(m_x) = m_y$$

$$hm_x(m_x) = m_c$$

for any

$$\forall me_x \in ME_x \quad \forall a_x \in A_x$$

there are

$$hmm_x(me_x) = \{me_{c,i}\}$$

$$hma_x(a_x) = \{a_{c,i}\}$$

$$me_{c,i} \in ME_c \quad a_{c,i} \in A_c$$

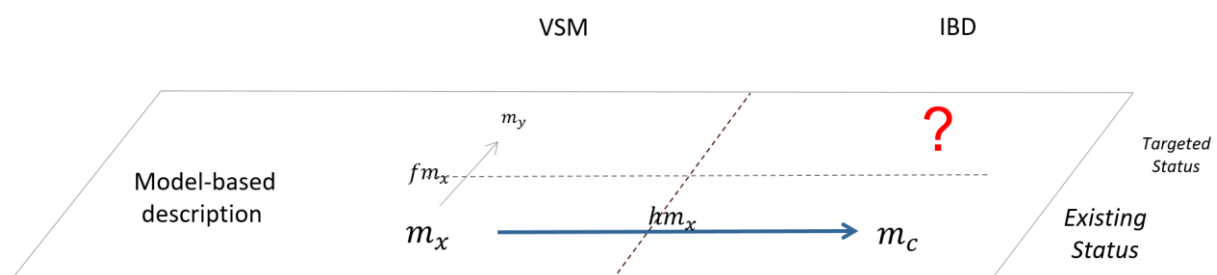
$$i = 1, \dots, n \quad n \in \mathbb{N}$$


Figure 94: Formalization of the start position with the cube model

## 5.2 Semantical mapping

All of the known VSM and IBD models will be equivalently transformed to graphs on the formal semantical foundation layer in this cube model. Such graphs represent different models with a universal form (see Figure 95). The equivalent transformation function  $semv$  transforms VSM model  $m_x$  to graph  $g_x$  and model  $m_y$  to graph  $g_y$ . The equivalent transformation function  $semi$  transforms IBD model  $m_c$  to graph  $g_c$  (see Definition 33 and Definition 40).

$$semv(m_x) = g_x \qquad semv(m_y) = g_y \qquad semi(m_c) = g_c$$

The transformations of the model and relation elements to their corresponding vertices and edges have been introduced in sections 4.1.3 and 4.2.3. The description information of the model/relation elements is reformed and ordered into the attributes of the corresponding vertices and edges.

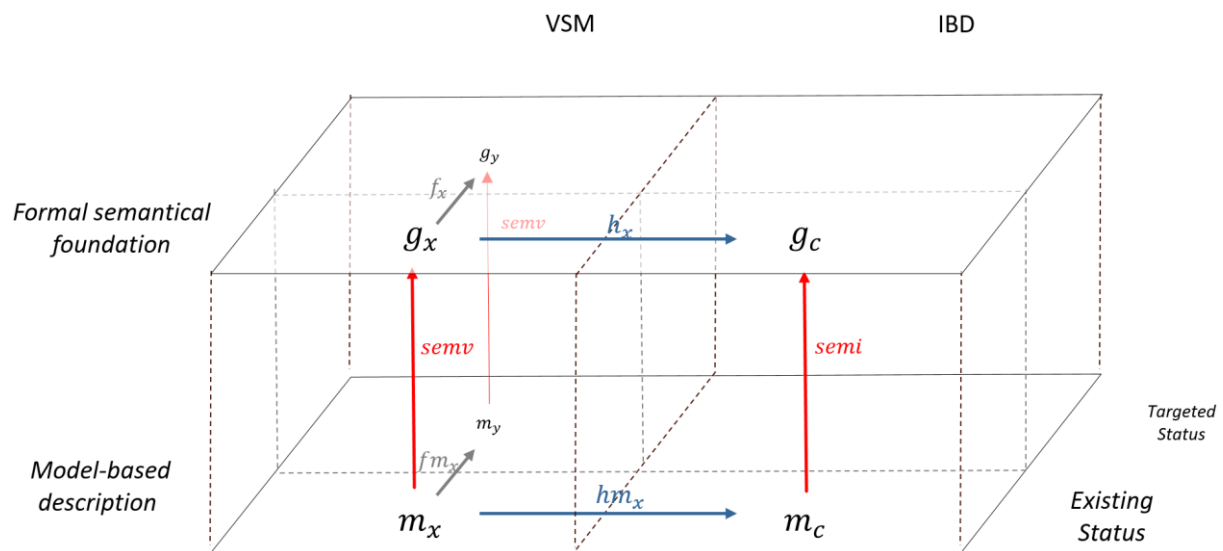


Figure 95: Semantical mapping with concrete example

Because the  $semv$  and  $semi$  are used for equivalent transformations, the graphs  $g_x$ ,  $g_y$  and  $g_c$  on the formal semantical foundation layer can be understood as the projections of the models  $m_x$ ,  $m_y$  and  $m_c$ . The mapping relationship  $h_x$  from  $g_x$  to  $g_c$  and the evolution function  $f_x$  from  $g_x$  to  $g_y$  are projected from the functions  $hm_x$  and  $fm_x$ . The mapping functions  $hmm_x$  and  $hma_x$  are projected to the mapping functions  $hv_x$  and  $he_x$  for vertices and edges from  $g_x$  to  $g_c$ . The models and functions are known, therefore the projected graphs and functions are known.

There are

$$\begin{aligned} g_x &\in G_{VSM,existing} \\ g_y &\in G_{VSM,targeted} \\ g_c &\in G_{IBD,existing} \end{aligned}$$

and

$$f_x(g_x) = g_y$$

$$h_x(g_x) = g_c$$

for any

$$\forall v_x \in V_x \quad \forall e_x \in E_x$$

there are

$$hv_x(v_x) = \{v_{c,i}\}$$

$$he_x(e_x) = \{e_{c,i}\}$$

$$v_{c,i} \in V_c \quad e_{c,i} \in E_c$$

$$i = 1, \dots, n \quad n \in \mathbb{N}$$

### 5.3 Generating graph solutions

All of the known models and functions at the start position of the managed evolution of a LL-CPS have been equivalently transformed to the graphs and functions on the formal semantical foundation layer in the previous section. In this section, they will be used to generate a set of graphs  $G_d = \{g_{d,i}\}$  from the VSM domain to the IBD domain on the formal semantical foundation layer by using algorithms in graph theory. Each graph in this set represents an IBD graph for the targeted status of this LL-CPS named the graph solution. A set of functions  $\{h_{y,i}\}$  maps the graph  $g_y$  to this set of graphs  $G_d = \{g_{d,i}\}$  (see Figure 96).

$$h_{y,i}(g_y) = \{g_{d,i}\} = G_d$$

$$i = 1, \dots, n \text{ and } n \in \mathbb{N}$$

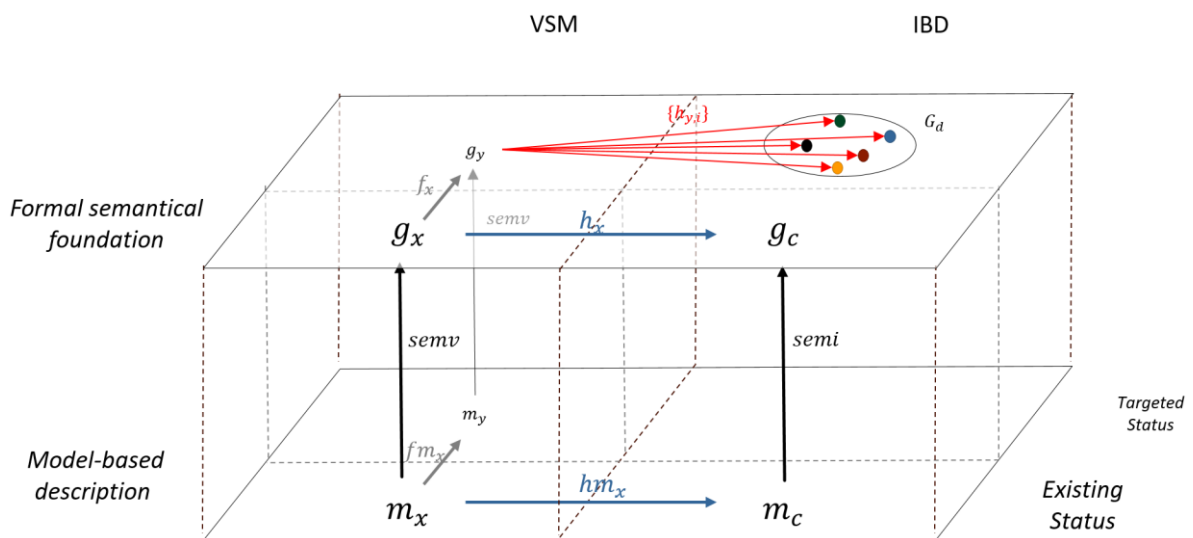


Figure 96: Generating a set of graph solutions



Graph  $g_y$  comprises two parts: the remaining vertices and edges and the new added vertices and edges during the managed evolution from  $g_x$  to  $g_y$ . By using the linear equations of the managed evolution function  $f_x$  (see section 4.4.1), these two parts can be reformed with a set of subgraphs  $G_x^+ = \{g_x^{+j}\}$  for all new added vertices and edges and a set of subgraphs  $g_y - G_x^-$  for all remaining vertices and edges.

$$\begin{aligned}
 G_x^+ &= \{g_x^{+i} \mid g_y - g_x\} \\
 G_x^- &= \{g_x^{-j} \mid g_x - g_y\} \\
 g_x^{+i} &\subseteq g_y \\
 g_x^{-j} &\subseteq g_x \\
 i, j &= 1, \dots, n \quad n \in \mathbb{N}
 \end{aligned}$$

Moreover, the mapping relationship from graph to graph is essentially the mapping for every vertex to vertices and edge to edges. For the remaining vertices and edges, the mapping functions  $hv_x$  and  $he_x$  are used to map them to the vertices and edges in graph every  $g_{d,i}$  (see Definition 45 and Definition 46). For the new added vertices and edges, it is necessary to define two new mapping functions.

$$\begin{aligned}
 \forall v \in V_y \exists hv_{y,i}(v) &= \begin{cases} hv_x(v), & \text{if } v \in V_x \\ \text{function 1}, & \text{if } v \notin V_x \end{cases} \\
 \forall e \in E_y \exists he_{y,i}(e) &= \begin{cases} he_x(e), & \text{if } e \in E_x \\ \text{function 2}, & \text{if } e \notin E_x \end{cases} \\
 i &= 1, \dots, n \text{ and } n \in \mathbb{N}
 \end{aligned}$$

### 5.3.1 Reforming the mapping domain

The path morphism mapping function  $pm$  (see Definition 13 and section 4.3.1.1) is used to define these two new mapping functions: *function 1* and *function 2*, for the new added vertices and edges in a VSM graph into vertices and edges in an IBD graph. The first step is reforming of the new added vertices and edges to paths, which are defined as the domain elements of the path morphism mapping function  $pm$ .

The set of subgraphs  $G_x^+$  includes all new added vertices and edges, which will be reformed into a set of paths  $\{p_l \mid l = 1, \dots, n \quad n \in \mathbb{N}\}$ , and they must satisfy the following requirements.

- The set of paths  $\{p_l\}$  is a finite set of paths and must include all of the new added vertices and edges in  $g_y$  compared with  $g_x$ .
- Any path has at least one remaining vertex or at most two remaining vertices.
- Any path must have at least one new added vertex.
- Any two remaining vertices cannot be adjacent in any path.
- Any remaining vertex cannot lie between two new added vertices in any path.
- Find out the paths including two remaining vertices as much as possible.

It should be emphasized that if the remaining vertices in any path are deleted, the remaining part after the deleting must be a series of connected new added vertices or one single new added vertex. This method can be used to evaluate the reforming results.

### 5.3.2 Creating the mapping codomain

The elements in the codomain of the mapping function  $pm$  should also be paths in a graph. A foundation graph  $g_c^*$  is created to determine the codomain of  $pm$ . The following steps show how to create this foundation graph  $g_c^*$ .

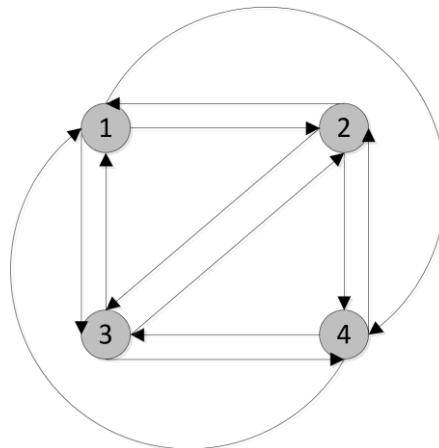


Figure 97: Complete directed graph K4

- First of all, get all of the vertices in graph  $g_c$ , which represents the existing status of the LL-CPS.
- Link those vertices together to build a complete directed graph  $Kn$ . The variate  $n$  is the number of all vertices in  $g_c^*$ . Figure 97 shows a complete directed graph  $K4$ .
- Subsequently, insert a virtual vertex between every two existing vertices in  $Kn$  and link the virtual vertex to the existing vertices to substitute the edges in  $Kn$ . This new graph is named  $Kn^*$ . Each virtual vertex is an IBD vertex and it must have an identifier, which comprises the ID of its front vertex and the ID of its following vertex. For instance, the

identifier of virtual vertex  $v_{x(a,b)}$  comprises the identifier “a” of its front vertex  $v_a$  and the identifier “b” of its the following vertex  $v_b$ . The index “x” shows, the vertex is a virtual vertex. The attributes in a virtual vertex can be defined as an empty value or an infinite value. The empty virtual vertex represents a directed edge. An infinite value can be understood as a black box contains any number of vertices to represent any change during the managed evolution of a LL-CPS. A new added virtual edge is an IBD edge. Figure 98 shows an example of a graph  $K4^*$ .

- Combine the graphs  $h_x(g_x - G_x^-)$  and  $Kn^*$  together to obtain a new directed graph  $g_c^*$ .

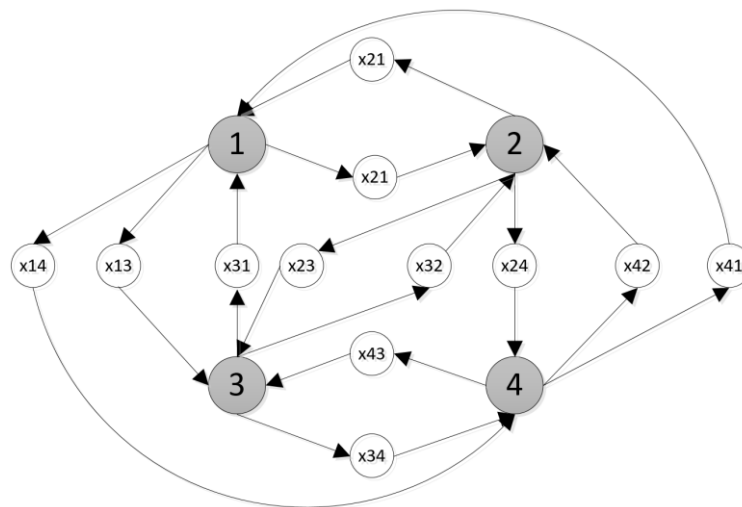


Figure 98: Directed graph  $K4^*$

Because the set of graphs  $G_x^-$  includes all deleted vertices and edges, all remaining vertices and edges in VSM graph during the managed evolution of LL-CPS can be formed with  $g_x - G_x^-$ . By using the mapping function  $h_x$ , all remaining vertices and edges in  $g_y$  during the managed evolution of the LL-CPS are mapped to  $h_x(g_x - G_x^-)$  that represents the remaining vertices and edges in  $g_c$  during the managed evolution of this LL-CPS.

$$h_x(g_x - G_x^-) = h_x(g_x) - h_x(G_x^-) = g_c - h_x(G_x^-)$$

The virtual vertices can contain any number of attributes to represent any change during the managed evolution of a LL-CPS, thus the graph  $g_c^*$  is understood as a graph represents any changes during the managed evolution of this LL-CPS. Therefore, a set of paths  $\{p_{l,i}\}$  in the foundation graph  $g_c^*$  is defined as the codomain of mapping  $pm$ .

### 5.3.3 Path morphism

The path morphism function  $pm$  (see Definition 13 and section 4.3.1.1) maps every path in a set of paths  $\{p_l\}$  in graph  $g_y$  to a set of paths  $\{p_{l,i}\}$  in graph  $g_x^*$ .

$$\begin{aligned}
 pm(p_l) &= \{p_{l,i}\} \\
 p_l &= (v_1, \dots, v_n) \\
 p_{l,i} &:= (v_1', \dots, v_n') \\
 pm(v_1) &:= \begin{cases} hv_x(v_1), & \text{if } v_1 \in V_x \\ \text{any connected virtual vertex with } hv_x(v_1), & \text{if } v_1 \notin V_x \end{cases} \\
 pm(v_n) &:= \begin{cases} hv_x(v_n), & \text{if } v_n \in V_x \\ \text{any connected virtual vertex with } hv_x(v_n), & \text{if } v_n \notin V_x \end{cases}
 \end{aligned}$$

Every path in  $\{p_{l,i}\}$  is a result path and it must satisfy the following requirements.

- Every result path must be a finite path.
- Every result path must include at least one remaining vertex.
- Each vertex in result path can only appear a finite number of times.

In order to obtain the mapping relationships for the new added vertices and edges, the result paths have to be reformed to vertices and edges.

If a remaining vertex or edge in a path  $p_l$  is deleted, the mapped vertices or edges in the corresponding path  $p_{l,i}$  will be also deleted. The other vertices and edges in path  $p_l$  hold the mapping relationships with the remaining parts in the corresponding paths  $p_{l,i}$ . This necessary condition for reforming is used to determine the two new mappings in  $hv_{y,i}$  and  $he_{y,i}$  for the new added vertices and edges. The set  $\{p_l\}$  includes all of the new added vertices and edges, therefore the mapping functions  $hv_{y,i}$  and  $he_{y,i}$  can be determined as below.

$$\begin{aligned}
 pm(p_l) &= \{p_{l,i}\} \\
 V_j &= \{v | v \in p_l \wedge v \in V_x\} \\
 \forall v \in p_l \exists hv_{y,i}(v) &= \begin{cases} hv_x(v), & \text{if } v \in V_x \\ V_{l,i}/hv_x(V_j), & \text{if } v \notin V_x \end{cases} \\
 \forall e \in p_l \exists he_{y,i}(e) &= \begin{cases} he_x(e), & \text{if } e \in E_x \\ \mathcal{P}(E_{l,i}), & \text{if } e \notin E_x \end{cases} \\
 i, j, l &= 1, \dots, n \quad n \in \mathbb{N}
 \end{aligned}$$

### 5.3.4 Graphs combination

Every result path in have to be combined with all vertices and edges in  $h_x(g_x - G_x^-)$  together to build a new graph  $g_{d,i}$ , which represents the targeted status of the LL-CPS.

$$g_{d,i} = \sum_{l=1}^n p_{l,i} + h_x(g_x - G_x^-)$$

Each graph  $g_{d,i}$  in the set of graph solutions  $G_d = \{g_{d,i}\}$  represents one reconstruction possibility for the targeted status of the ongoing LL-CPS.

After undertaking all of the steps above, the objective mapping functions  $h_{y,i}$ ,  $h_{v_{y,i}}$  and  $h_{e_{y,i}}$  are determined.

### 5.4 Concrete modeling

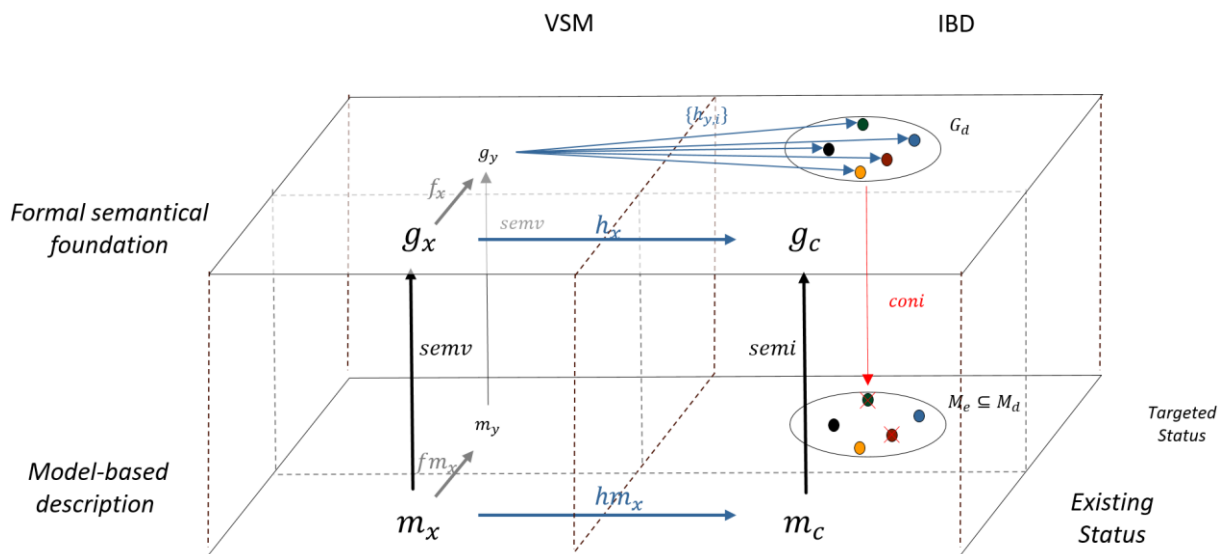


Figure 99: Concrete modeling of graph solutions

The set of graph solutions  $G_d$  have to be equivalently transformed from the formal semantical foundation layer to a set of models on the model-based layer in the cube model to obtain the IBD models. The equivalent transformation function  $coni$  transforms the set of graphs  $G_d = \{g_{d,i}\}$  to a set of IBD models  $M_d = \{m_{d,i}\}$  (see Definition 41).

$$coni(g_{d,i}) = m_{d,i}$$

If the IBD vertices and edges in  $g_{d,i}$  are remaining vertices and edges during the managed evolution of the LL-CPS, they have to be transformed to the original model elements and relation elements (see section 4.2.4). If an IBD vertex or edge is a virtual vertex or edge, it has to be transformed to a virtual model element  $vme_{d,i}$  or a virtual relation element  $va_{d,i}$  in an IBD model. The virtual model element or virtual relation element is defined as an IBD model element or IBD relation element with the undefined description information.

$$vme_{d,i} \in ME_{IBD} \quad va_{d,i} \in A_{IBD}$$

Before to ascertaining the costs local optimal solution in the set of models  $M_d$ , the description information of the virtual model elements and the virtual relation elements in any model  $m_{d,i}$  should be filled. For instance, the virtual model elements and virtual relation elements can be filled with different type information. The filled virtual model element can be an IS-HW block or a HMI port or an integrated sub-model. The filled virtual relation element can be an input, output or connection relation element. Any virtual model and relation element has to be filled to satisfy the combination rules as much as possible, which have been introduced in section 4.2.2.

Every model  $m_{d,i}$  after the filling must satisfy two requirements. The IBD models, which can meet these two filter conditions, make up a set of models  $M_e$ . This set is a subset of  $M_d$  (see Figure 99).

- The first requirement is the combination rules of this IBD modeling. If any filled model element or relation element is still unable to meet the combination rules, this model will be deleted from the set of models  $M_d$ .
- Because the model elements and relation elements in the VSM model  $m_y$  can be mapped to model elements and relation elements in the models in  $M_d$  by using the cube model. Therefore, the model elements and relation elements in any IBD model  $m_{d,i} \in M_d$  must satisfy execution sequence of the mapped the model elements and relation elements in VSM model  $m_y$ . If any IBD model cannot satisfy the execution sequence, it will be filtered out from the set  $M_d$ .

## 5.5 Optimizing the model solutions

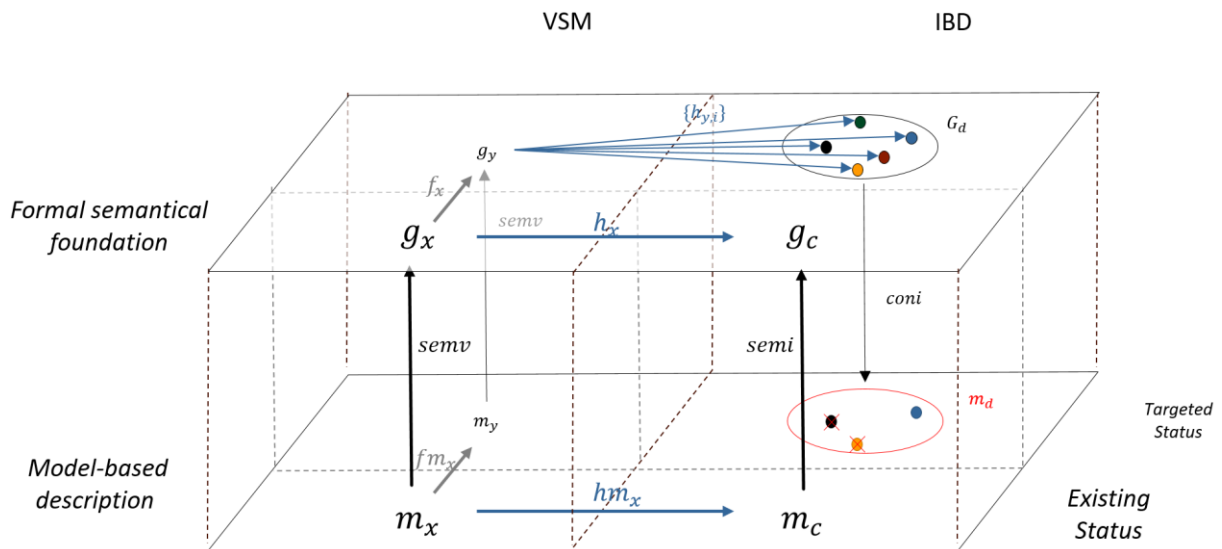


Figure 100: Optimizing the model solutions

Each IBD model in the set  $M_e$  represents a possible reconstruction for the managed evolution of the LL-CPS from the existing status to its targeted status. In order to ascertain the cost-optimal one, every model and relation element in each IBD model will be labelled with “costs”. For every model solution  $m_{d,i}$  in  $M_e$ , there is a costs function  $C$  to label the costs of the reconstruction for every model element in the IBD model. The total costs of the reconstruction in one model solution is defined as the sum of these costs of its all reconstructed model elements.

$$C := C |_{ME_{IBD} \rightarrow \mathbb{N}}$$

$$C(m_{d,i}) = \sum C(ME_{d,i})$$

The local optimal solution  $m_d$  (see Definition 18) is defined as the final solution for the local minimal cost of the reconstruction during the managed evolution of LL-CPS (see Figure 100).

$$m_d = \min(C(m_{d,i})) \quad i = 1, \dots, n \quad n \in \mathbb{N}$$

This approach is introduced for the managed evolution of a LL-CPS with local optimal costs and controlled risks in ongoing operations. In chapter 6, this approach will be implemented. The evaluation of this approach and the implementation will be introduced in chapter 7.





## 6 Implementation

### Content

---

- 6.1 System requirements
    - 6.1.1 Use case diagram
    - 6.1.2 Input models restructuring
  - 6.2 System architecture
    - 6.2.1 System structure
    - 6.2.2 System behavior
  - 6.3 Functions realization
- 

In this chapter, a Java application is developed to implement the approach that has been introduced in the previous chapter. In section 6.1, the system requirements of this application are illustrated by using of a use case diagram. Subsequently, the Input models restructuring for this application is introduced to serve a foundation for the implementation of the main tasks of this approach. In section 6.2, the system architecture and behavior in this application are introduced by using a class diagram and sequence diagram. The realization of the important algorithms and functions is introduced in the last section by using the formal and informal programming description.

### 6.1 System requirements

The approach should be implemented within the scope of a suitable environment and a clear implementation process. All of the requirements for this application are illustrated by using the use case diagram in this section.

#### 6.1.1 Use case diagram

This application is named the LL-CPS managed evolution solutions generation system and its system environment comprises LL-CPS planner and LL-CPS designer. The LL-CPS planner defines two VSM models as input models for the managed evolution of LL-CPS. The LL-CPS designer places the IBD model of the ongoing LL-CPS as an input model and he defines the mapping relationships form the VSM model to this IBD model for the ongoing LL-CPS. By using this application, the LL-CPS designer obtains the costs local optimal IBD model as a solution for the implementation of the managed evolution of this LL-CPS, which is visualized with a graphical user interface (GUI). The other functions and algorithms in this application are shown with a block “Generating” in Figure 101.

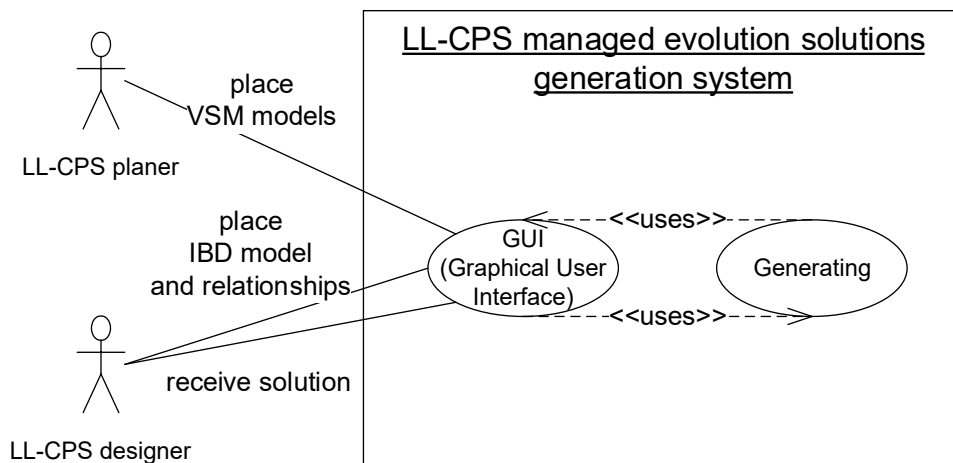


Figure 101: Use case diagram for the LL-CPS managed evolution solutions generation system

### 6.1.2 Input models restructuring

The data structure plays an important role in the system requirements, which serves the basis for the later data processing. All of the input models for this application will be reformed with a standard data structure. Every VSM model has to be restructured by the LL-CPS planner with the key-value data structure in a pair documents. One of them represents the model elements in the VSM model. The other one represents the relation elements with a two-tuple data structure, which describes the connection relationship with the IDs from one model element to another.

Key (ID)	Value
A	string
b	string

(a) Model elements in a VSM model

Relation elements
a,b
b,a

(b) Connection relationships in a VSM model

Table 12: Example of data structure in a VSM model

Table 12 shows an example of this data structure with two inputs documents for a VSM described LL-CPS.

The IBD model restructured by the LL-CPS designer describes the ongoing LL-CPS (see Table 13) with the same data structure as the VSM models.

Key (ID)	Value
1	string
2	string
3	string

(a) Model elements in an IBD model

Relation elements
1,2
2,3

(b) Connection relationships in an IBD model

Table 13: Example of data structure in an IBD model

There are another two necessary input documents to represent the mapping relations for the model and relation elements in the VSM model to model and relation elements in the IBD model, which model the same ongoing LL-CPS. Table 14 illustrates these mapping relationships with two input documents.

In this example, the model element “a” in the VSM model is mapped to a set of model elements “1” and “2” in the IBD model. The relation element “a, b” in this VSM model is mapped to the relation element “2,3” in the IBD model.

Model element in VSM	Model elements in IBD
a	1,2

(a) Input document for the mapping relation between model elements

Relation element in VSM	Model elements in IBD
a,b	2,3

(b) Input document for the mapping relation between relation elements

Table 14: Example of input document for the mapping relationships

## 6.2 System architecture

### 6.2.1 System structure

The system structure describes the organization and arrangement of interrelated components in a system. In software engineering, it can be represented in diagrams such as a class diagram, in which the structural classes reflect the functional requirements of the application.

The class diagram in Figure 102 shows the system structure of this application implemented by Java in the software programming tool Eclipse. The class “home” provides a graphical user interface for the generic organizing and structuring of this application and therein the program

## Chapter 6 - Implementation

starts with the main function. The class “Algorithmilib” is an algorithms library and it comprises the different algorithms; for instance, the algorithms in the graph theory, the optimization problem, etc. They will be used by the class “graph” to implement the functional requirements. The class “SystemRules” provides the combination rules, which is used by the class “Algorithmilib” to obtain the combinable solutions.

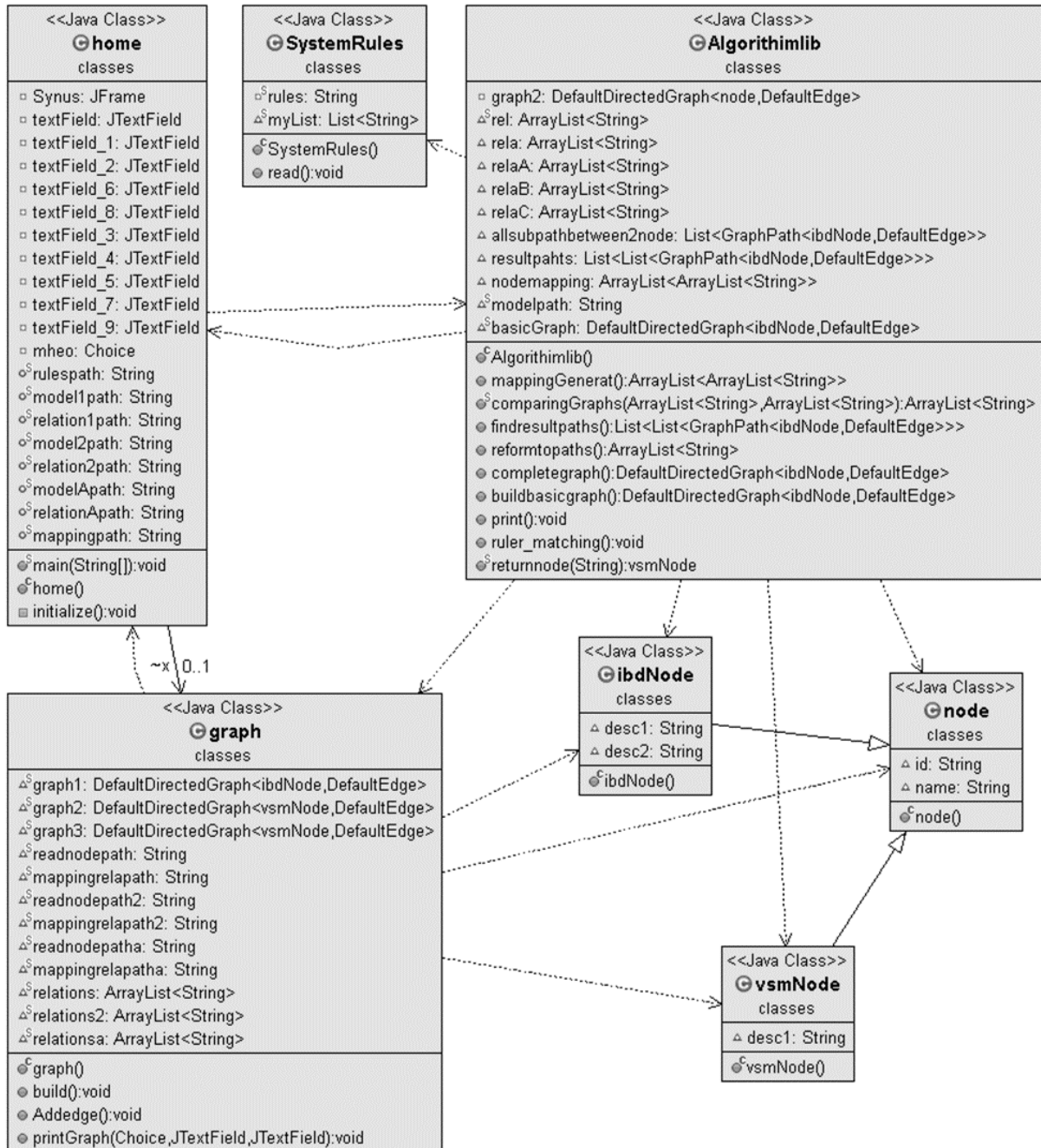


Figure 102: Class diagram of the application

### 6.2.2 System behavior

The system behavior is a specification of events that occur dynamically over situations or time. This specification is determined specifically by what events occur in that situations or time. The sequence diagram in Figure 103 represents the interactions between the classes in this application, arranged in time sequence.

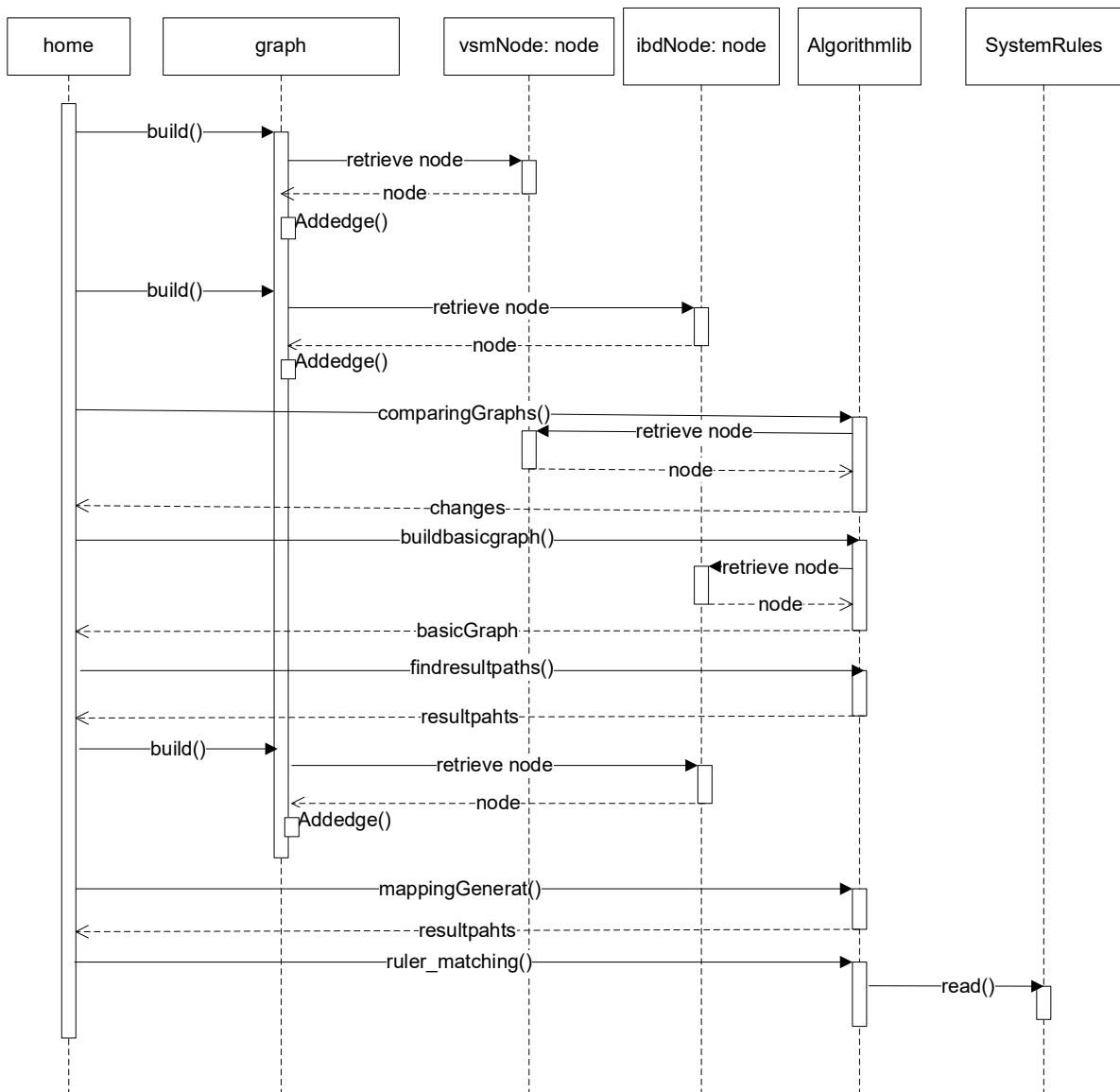


Figure 103: Sequence diagram of the application

## 6.3 Functions realization

Two important functions for this application are introduced by using the informal and formal descriptions in this section.

The reforming the mapping domain of the path morphism function  $pm$  introduced in section 5.3.1 is first introduced with the pseudocodes below. They describe the operating principle of the algorithm, which reforms the new added vertices and edges during the managed evolution of a LL-CPS into a set of paths.

---

**Algorithm** findChangedPaths

---

**Input:** verxGp1: all of the vertexes in Gp1,  
 verxGp2: all of the vertexes in Gp2,  
 edgesGp2: all of the edges in Gp2

**Output:** candidates[ ]

**Parameters and variables:**

(Vx ,Vy) is any edge from Vx to Vy in Gp2. (Vx,Vy)  $\in$  edgesGp2

candidates[ ] is a set of paths

```

1: initialize: set candidates[ ] = null
2: set all of the elements in verxGp2 as unvisited vertices:
   Vx.visited = false
3: set all of the elements in edgesGp2 as unvisited edges:
   (Vx,Vy).visited = false
4: while any edge has not been visited do
5:   if Vx and Vy are in verxGp1 then
6:     set edge (Vx,Vy).visited = true
7:   else if Vx  $\notin$  verxGp1 and Vy  $\notin$  verxGp1 then
8:     set edge (Vx,Vy).visited = true
9:   else if Vx  $\in$  verxGp1 and Vy  $\notin$  verxGp1 then
10:    set edge (Vx,Vy).visited = true
11:    function arrowSearch(Vx,Vy )
12:   else if Vx  $\notin$  verxGp1 and Vy  $\in$  verxGp1 then
10:    set edge (Vx,Vy).visited = true
14:    function oppositeSearch(Vx,Vy )
15:   End if
16: End while

```

---

The functions arrowSearch and oppositeSearch are used to chain all of the new added vertices and edges into paths. The pseudocodes below describe the function arrowSearch. The function oppositeSearch is the same as the function arrowSearch, albeit in a different search direction (see section 13.1).

---

**function** arrowSearch( )

---

```

1: initialize: set list tempPath[ ] = null
2: initialize: set a Stack S := null
3: Set Vx.visited = true, Vy.visited = true
4: if Vy has any linked following vertex in Gp2
5:   push all of the linked following vertices of Vy into Stack S
6:   while S is not empty do
7:     if S.peek.visited = false then
8:       if S.peek  $\notin$  verxGp1 then
9:         if S.peek has not linked following vertex then
10:          add S.peek into tempPath[ ]
11:          set S.peek.visited = ture
12:          save Path(Vx,Vy,tempPath[ ]) in candidates[ ]
13:          set all paths in Path(Vx,Vy,tempPath[ ]) as visited paths
14:          delete S.peek
15:          reset tempPath[ ] =null
16:        else
17:          add S.peek into tempPath[ ]
18:          set S.peek.visited = ture
19:          replace S.peek with all of its linked following vertices in S
20:        else
21:          add S.peek into tempPath[ ]
22:          set S.peek.visited = ture
23:          save Path(Vx,Vy,tempPath[ ]) in candidates[ ]
24:          set all paths in Path(Vx,Vy,tempPath[ ]) as visited paths
25:          delete S.peek
26:          reset tempPath[ ] =null
27:        else
28:          delete S.peek
29:        End if
30:      End while
31:    else
32:      save Path(Vx,Vy) in candidates[ ]

```

---

The implementation of another important function is introduced with the Java codes in Figure 104, which is used to compose the result paths to a set of solution graphs.

```

for(int m = 0; m < temp1.size(); m++){
    if(vertexExisted.contains(temp1.get(m))){
        continue;
    }else{
        bw1.write(temp1.get(m)+ ", " + temp1.get(m));
        bw1.flush();
        bw1.newLine();
        bw2.write(temp1.get(m-1)+ ", " + temp1.get(m));
        bw2.flush();
        bw2.newLine();
        bw2.write(temp1.get(m)+ ", " + temp1.get(m+1));
        bw2.flush();
        bw2.newLine();
    }
}

```

Figure 104: Generating to graphs in Java code





## 7 Evaluation

### Content

---

- 7.1 Case study 1: Conveyor System with ASRS
    - 7.1.1 LL-CPS managed evolution by using the approach
    - 7.1.2 Comparison of solutions
    - 7.1.3 Development risk evaluation
    - 7.1.4 Economic evaluation
  
  - 7.2 Case study 2: Project “Synus”
- 

This chapter evaluates the efficiency of approach introduced in chapter 5 to solve two problems during the managed evolution of LL-CPSs in this thesis: one problem is the local minimization of the reconstruction costs of implementation (economic requirements), the other one is the controlled risks in ongoing operations (risk management). Therefore, the evaluation is divided into two parts: the development risk evaluation and economic evaluation. Figure 105 shows the evaluation concept. In this chapter, two cases are used to evaluate this approach. One is the laboratory model of the conveyor system with ASRS introduced in chapter 3, which will be here continually applied for the evaluation. The other one is an industry project named in short Synus: Methods and tools for the synergetic conception and evaluation of Industry 4.0 solutions, which is funded by the European Regional Development Fund (EFRE-ZW 6-85012454) and managed by the Project Management Agency NBank [58].

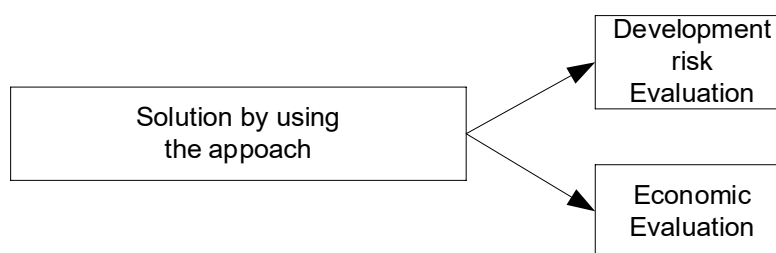


Figure 105: The evaluation concept

### 7.1 Case study 1: Conveyor System with ASRS

Continuing the laboratory model of a conveyor system with ASRS in chapter 3, the ongoing system is defined as the existing status of this LL-CPS and described with a VSM model (section 3.1.1) and an IBD model (section 3.1.2). The targeted status of this LL-CPS is formed with another VSM model (section 3.2.2), which also describes the new functional requirements

during the managed evolution. By using the approach, a set of IBD models is generated, whereby a final solution is selected and implemented that represents the local optimal costs of the reconstruction. In order to evaluate this solution, another IBD model is chosen for evaluating the economic requirements and development risk. In this case, the direct costs are defined as the reconstruction costs during the managed evolution of a LL-CPS, whereby it is not necessary to consider the time-dependent costs.

### 7.1.1 Managed evolution of LL-CPS by using the approach

The existing status of LL-CPS is represented with a IBD model  $m_c$ . The final solution is formed with IBD model  $m_{d,2}$ . IBD model  $m_{d,1}$  represents another evolution solution. They will be implemented and compared with each other and model  $m_c$ . This comparison is from four areas in LL-CPS: the relevant environment factors, mechanical system, control system and information system. The reconstructions during the managed evolution of LL-CPS are specified into three actions: the addition, modification and deletion.

#### The implemented model $m_{d,1}$

Model  $m_{d,1}$  is implemented as one solution for the managed evolution of the ongoing LL-CPS. A new RFID real sensor is integrated in  $m_{d,1}$  to read the color information of the wares. The software codes are developed to control the gripper robot and many other hardware components to reach the new functional requirements in the targeted status of this LL-CPS. Compared with the existing status  $m_c$ , worker 1 is deleted in  $m_{d,1}$ . The connections between worker 1 and the conveyor system are also deleted (see Figure 106).

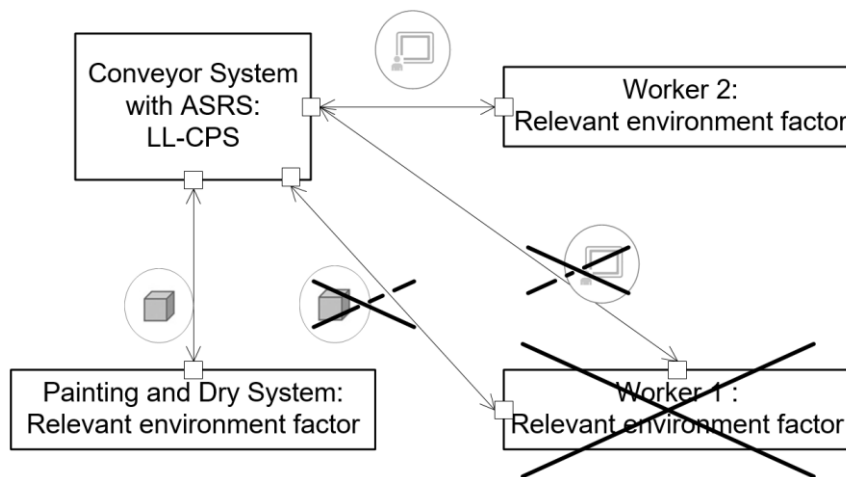


Figure 106: Changes of the relevant environment factors in  $m_{d,1}$

In the mechanical system in  $m_{d,1}$ , a new RFID read sensor is installed. Figure 107 shows the changes of hardware components in the mechanical system in  $m_{d,1}$ .

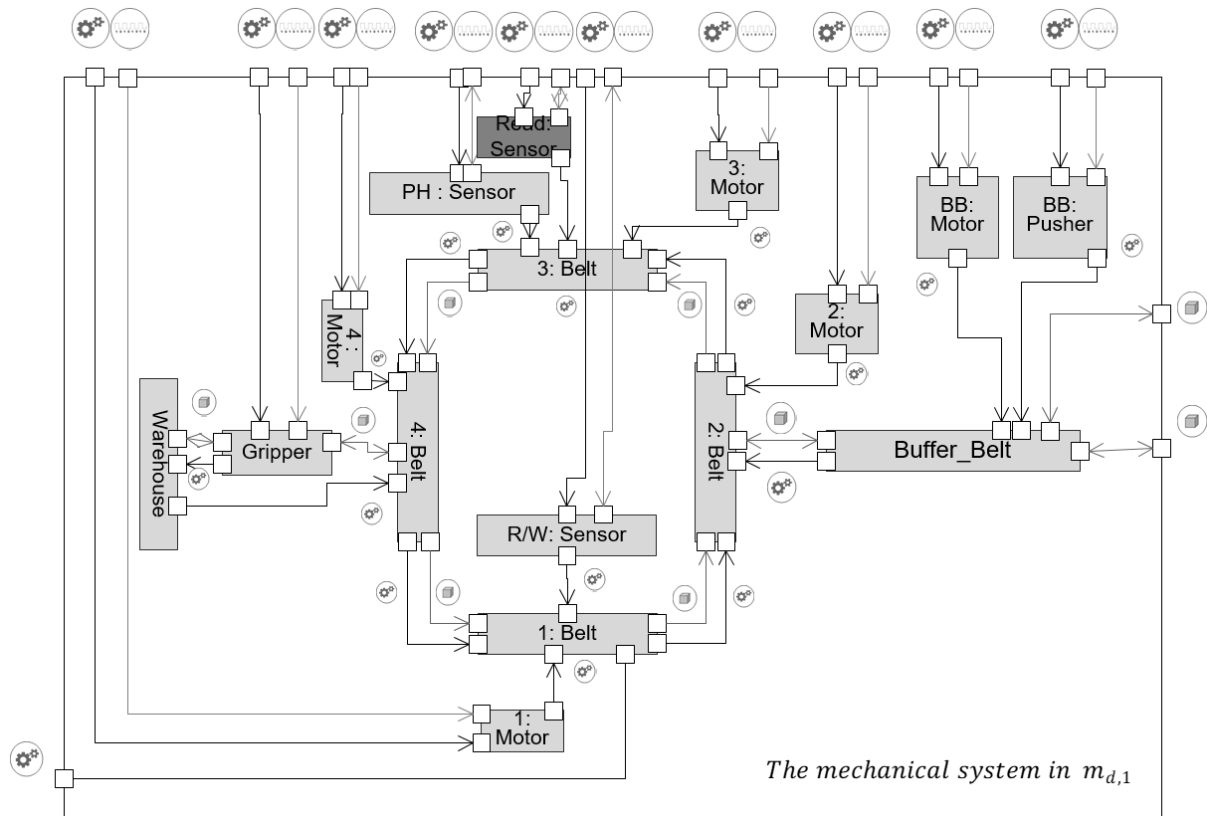


Figure 107: Changes of hardware components in mechanical system in  $m_{d,1}$

In  $m_{d,1}$ , the software codes in the control system and information system have to be changed to adapt to the reconstruction in the mechanical system. Figure 108 shows the changes of the software codes in the control system in  $m_{d,1}$  compared with the ongoing LL-CPS modeled with model  $m_c$ . In this figure, the red color marks the deleted codes during the managed evolution from  $m_c$  to  $m_{d,1}$ . The green color marks the new added codes and the blue one labels the codes that changed the executing place.

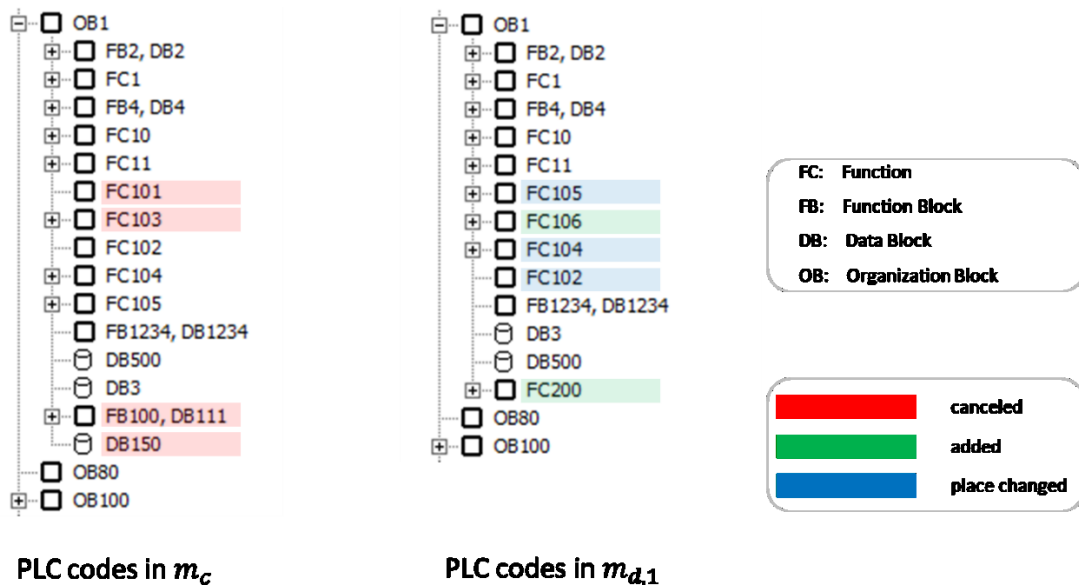
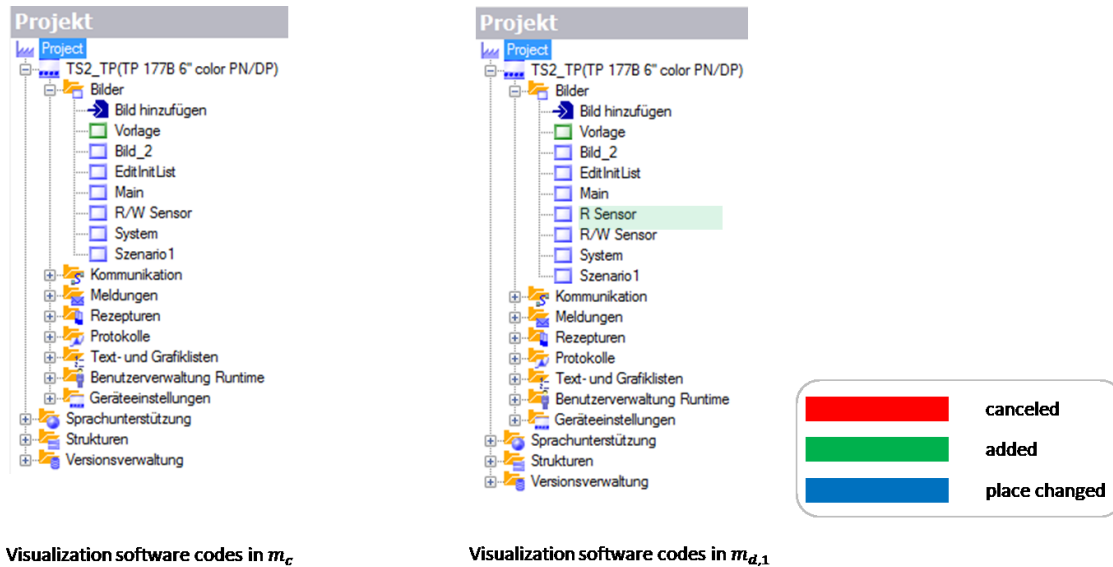


Figure 108: Comparing software codes in control system between  $m_c$  and  $m_{d,1}$

Figure 109 shows the changes of the codes in the information system during the managed evolution from  $m_c$  to  $m_{d,1}$ . The new information system has the new codes to visualise the information collected from the new RFID read sensor.



### The implemented model $m_{d,2}$

In  $m_{d,2}$ , an existing RFID read/write sensor is used to write the color information into the ware, when the ware reaches it for the first time, and it is reused to read the color information from the ware, when the ware reaches it again. In this situation, the other hardware components have to be adapted to reach this requirement. Accordingly, all four conveyor belts have to after the painting color process continually transport the wares in a cycle to enable the wares to reach the existing RFID read/write sensor again. Meanwhile the gripper and the photoelectric sensor are blocked to let the ware run in a cycle and activated again, when the read/write sensor obtain the full information of all wares.

Compared with the existing status  $m_c$ , worker 1 and its related connections have to be deleted to reach the requirements in targeted status of this LL-CPS (see Figure 110).

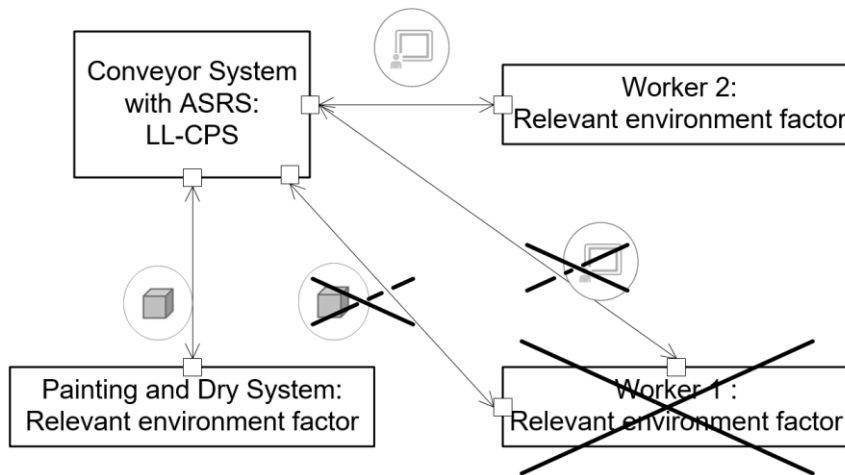


Figure 110: Changes of the relevant environment factors in  $m_{d,2}$

In the mechanical system in  $m_{d,2}$ , there are no changes during the managed evolution. However, the software codes need to be developed to reach the functional requirements. Figure 111 shows the changes of software codes in the control system in  $m_{d,2}$  compared with  $m_c$ .



Figure 111: Comparing software codes in control system between  $m_c$  and  $m_{d,2}$

The new codes in the information system in  $m_{d,2}$  visualise the information collected by using the existing RFID Read/Write sensor (see Figure 112). The other code changes in  $m_{d,2}$  comparing to the ongoing LL-CPS is introduced in section 13.2.

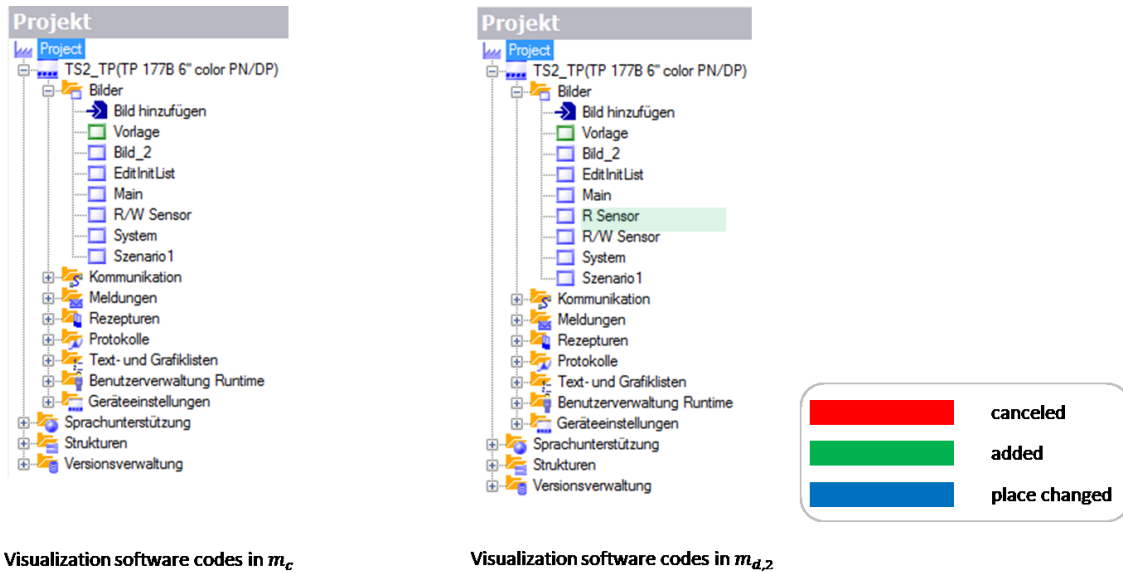


Figure 112: Comparing software codes in information system between  $m_c$  and  $m_{d,2}$

### 7.1.2 Comparison of solutions

The evolution function  $f_m$  and its linear equations system introduced in section 4.4.2 are used here to compare the changes between  $m_{d,1}$  and  $m_{d,2}$ .

There are

$$f_{m_1}(m_c) = m_{d,1}$$

$$f_{m_2}(m_c) = m_{d,2}$$

and

$$m_{d,1} = m_c \oplus M_c^{\Delta 1}$$

$$m_{d,2} = m_c \oplus M_c^{\Delta 2}$$

The set of sub-models  $M_c^{\Delta 1}$  represents the changes during the managed evolution form  $m_c$  to  $m_{d,1}$ , and  $M_c^{\Delta 2}$  represents the changes during the managed evolution form  $m_c$  to  $m_{d,2}$ .

### 7.1.3 Development risk evaluation

The development risk evaluation can be divided into three parts: risk identification, risk analysis and risk prioritization [59] as cited in [60]. The risk identification entails listing the important risks. In this case, the matching of functions is identified as the most important risk

factor for the managed evolution of LL-CPSs. The risk analysis involves the possible negative effects for each risk. The risk prioritization specifies the sequence of negative effects of risks.

The functional requirements have been defined with a VSM model in section 3.2 and are specified with the following points:

1. There is no worker standing by the buffer belt to sort the wares.
2. The color information is read by using RFID sensor.
3. The wares are retrieved through the gripper robot with the sort information in the predefined floor in the warehouse.

Table 15 shows that all of the functional requirements are satisfied in  $m_{d,1}$  and  $m_{d,2}$ .

Requirements	Solution $m_{d,1}$	Solution $m_{d,2}$
1	Yes	Yes
2	Yes	Yes
3	Yes	Yes

Table 15. The functional evaluation of two solutions

### 7.1.4 Economic evaluation

In order to evaluate the economic efficiency, the economic requirements during the managed evolution of LL-CPSs need to be specified. In this case, the total reconstruction costs only comprise the direct costs. The indirect costs, the non-construction related costs, the time dependent costs, the software codes rewriting costs e.g. are not included in the total reconstruction costs. A costs function  $C$  represents a mapping relationship from the model and relation elements to their direct costs for reconstruction.  $DC$  is a set of direct costs of the model and relation elements for reconstruction during managed evolution of LL-CPSs.

Definition 52

$$C := (ME \cup A) \rightarrow DC$$

The local optimal solution must have a local minimal reconstruction costs compared with the ongoing LL-CPS (see Definition 18).

$$P_{costs} = \min(C(M_c^{\Delta i}))$$

$$i = 1, \dots, n \quad n \in \mathbb{N}$$

In this case, the model elements are classified into a set of hardware elements and a set of software elements, because the direct reconstruction costs are related to this character of the model elements. The reconstructions of any relation element are defined as the cost free works. Table 16 shows an example of the direct reconstruction costs of the model and relations elements classified by cyber-physical characters.

Reconstruction actions \ Character	Addition	Modification	Deletion
HW element	200 unit	100 unit	20 unit
SW element	50 unit	20 unit	1 unit
Relation element	0 unit	0 unit	0 unit

Table 16: Example of the direct costs for reconstruction

The addition of a new hardware element is among the most expensive in all of the reconstruction actions. The modifications of hardware and software elements incur more costs than their deletions. The costs unit here is a unit of measurement, named as a unit.

The model  $m_{d,2}$  can be proved by practice as the local optimal solution in the set of all possible solutions by using the method named proof by exhaustion, in which the direct reconstruction costs of every model and relation element in the table above are used to check the local solutions.

### 7.2 Case study 2: Project “Synus”

Industry 4.0 is the short name for the fourth industrial evolution, which encompasses areas of intelligent, network, smart factory, etc. Since 2012, the German Federal Ministry of Education and Research has supported more than EUR 120 Million for projects in the context of industry 4.0 to improve the global competitiveness and the level of prosperity in the country [61]. In the coming years, it is expected that German industrial companies will invest up to EUR 40 Billion in industry 4.0 solutions [62]. The project “Methods and tools for the synergetic conception and evaluation of Industry 4.0 solutions” (in short Synus) began in June 2017 and will last until June 2020. The important goal of this project is to solve the problem that during the use of industry 4.0 solutions often involve high investment and unknown follow-up costs; for instance, integration an industry 4.0 solution on an ongoing production system. Figure 113 [63] shows a concept for a data-driven managed evolution of a production system, which is defined as a LL-CPS. The evaluating factors like production time, energy consumption and processing costs are extracted for the evaluation of the ongoing production system. During



the analysis of the result of the evaluation by the experts, some current industry 4.0 solutions are proposed to improve the production performance of the ongoing production system. The changes from the existing status of this production system to its targeted status are formed with requirements data, which drives on the evolution of this system. The approach introduced in this thesis is used to generate a set of the configurations of the components in the targeted status of this production system, which have to meet the requirements of the driving data. One of these configurations will be simulated and implemented as a new production system. This new system is named target 1 and it will be continually evaluated and evolved into the second iteration.

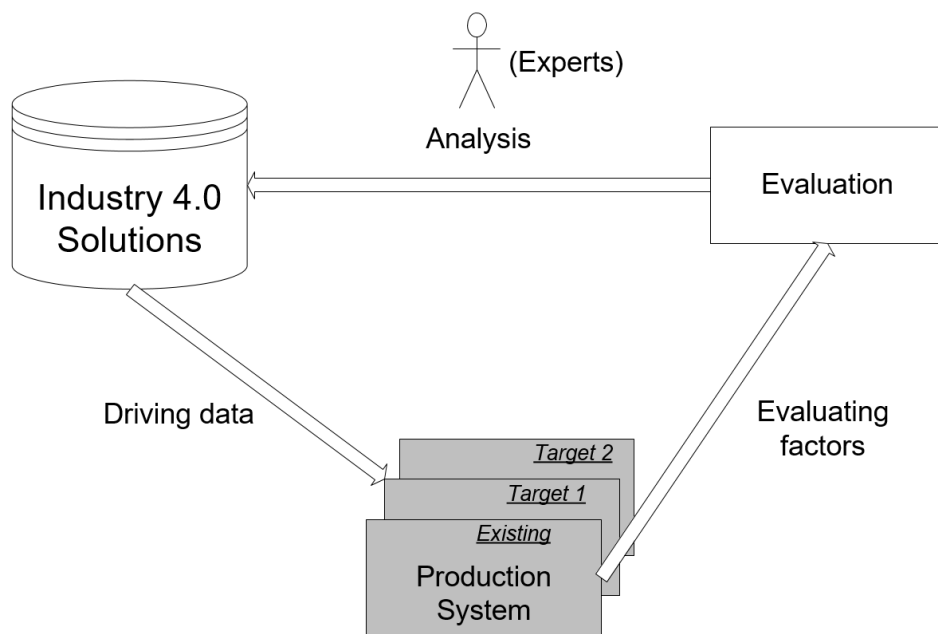


Figure 113: Data driven development of a production system

At present, there are few opportunities for small and medium-size enterprises (SMEs) to gather the information which they need to adopt Industry 4.0 solutions. This project using the approach provides a decision support for the development of LL-CPSs in the SMEs, before the implementation of new systems.

The project Synus is financed by European Regional Development Fund (ERDF) for Lower Saxony to help to reduce the regional imbalances. The goal of ERDF is to build up the economic and social cohesion in the European Union by correcting imbalances between its regions.



## 8 Conclusion and Outlook

### Content

---

8.1 Conclusion

8.2 Outlook

---

### 8.1 Conclusion

In this thesis, an approach has been developed to generate a costs local optimal solution for the managed evolution of a LL-CPS with controlled risks in system development. This solution is generated by tracing the changes of the planned requirements during the system managed evolution and it will be used for the implementation of the targeted status of this LL-CPS that helps the system designer reduce the development risks.

In this thesis, the following research achievements have been achieved:

- The modeling method VSM has been introduced in details and applied to model the existing and targeted statuses of a LL-CPS.
- The modeling processes and the interfaces specification for a LL-CPS by using the modeling method IBD have been introduced.
- A graph based uniform description of the models modeled with two different modeling methods has been developed.
- A cube model has been formed to integrate the models modeled with two different modeling methods and the system evolution together. This cube model provides a clear and formal description for the managed evolution of a LL-CPS modeled with two different modeling methods. Moreover, it provides the transformations between the models on the model-based description layer and the graphs on the formal semantical foundation layer.
- An approach has been developed that uses the cube model to automatic generate a set of models for implementation by tracing the changes of the planned requirements. This generating reduces the inconsistency between the system implementation and the planned requirements during the managed evolution of a LL-CPS. For a long-living system, this approach can be iteratively and evolutionarily used for a long-term system evolution.
- This set of models provides a candidate to evaluate the different possible solutions.
- The implementation and evaluation of this approach have been introduced.

## 8.2 Outlook

Several areas for future work are suggested regarding the approach presented in this thesis.

First of all, the approach is used on two modeling methods (VSM and IBD) and generates the evolution changes from one modeling method to another. In the future, it may be improved by using conduction to diffuse the evolution changes from one modeling method to more modeling methods. This conduction can chain different modeling methods together to achieve a synergetic system evolution.

The input models for this approach are formed with a key-value data structure, although the transformation processes from the input models to this data structure are not automatic. Therefore, an automatic transformation platform can be developed in the future and integrated with the modeling tools to supersede the manual transformation.

In this thesis, this approach provides only one final solution. Future work could expand to a range of a set of solutions; for example, the first ten minimal costs of reconstruction in a set of solutions.

The local optimal costs of reconstruction is considered as the only one evaluation factor used to ascertain the final solution in this thesis. However, today increasingly more requirements need to be factored during managed evolution of a LL-CPS; for example, the energy expenditure, the costs of maintenance and environmental protection, etc. In the future, this approach can be continually developed considering more evaluation factors and the preference of different evaluation requirements.

In summary, this work addresses an important aspect of consistency assurance between evolution requirements and architectures of implementation. For a complete automation, the consistency assurance between requirements and architectures in the field of system engineering needs to be carried out in the future.

## 9 Publication Bibliography

- [1] M. Broy, "Cyber-Physical Systems Innovation durch Software-Intensive eingebettet Systeme," *acatech Disk.*, p. 17, 2010.
- [2] H. Chen, "Applications of Cyber-Physical System: A Literature Review," *J. Ind. Integr. Manag.*, vol. 02, no. 03, pp. 1750012–2, 2017.
- [3] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," *47th ACM/IEEE Des. Autom. Conf.*, pp. 731–736, 2010.
- [4] E. A. Lee, "Cyber-Physical Systems - Are Computing Foundations Adequate?," *Position Pap. NSF Work. Cyber-Physical Syst. Res. Motiv. Tech. Roadmap*, pp. 1–9, 2006.
- [5] A. Sangiovanni-Vincentelli et al., "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems\*," *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, 2012.
- [6] H. Giese, B. Rumpe, B. Schätz, and J. Sztipanovits, "Science and Engineering of Cyber-Physical Systems," *Dagstuhl Semin.*, vol. 11445, no. 11, pp. 1–22, 2011.
- [7] I. Gerostathopoulos, "Ilias Gerostathopoulos Model-Driven Development of Software-Intensive Cyber-Physical Systems," 2015.
- [8] E. Geisberger and M. Broy, "agenda CPS - Integrierte Forschungsagenda Cyber-Physical Systems," 2012.
- [9] H. Koziol et al., "Towards Software Sustainability Guidelines for Long-living Industrial Systems," in *3rd Workshop of GI Working Group'Longliving Software Systems L2S2 Design for Future 2011*, 2011.
- [10] U. Goltz, R. H. Reussner, M. Goedicke, W. Hasselbring, L. Martin, and B. Vogel-Heuser, "Design for future: managed software evolution: The DFG priority programme for long-living software systems," *Comput. Sci. - Res. Dev.*, vol. 30, no. 3–4, pp. 321–331, 2015.
- [11] F. Li, G. Bayrak, K. Kernschmidt, and B. Vogel-Heuser, "Specification of the requirements to support information technology-cycles in the machine and plant manufacturing industry," *IFAC Proc. Vol.*, vol. 14, no. PART 1, pp. 1077–1082, 2012.
- [12] Peter Hartmann, *Mathematik für Informatik*. Landshut: Springer Vieweg, 2012.
- [13] R. R. Geiß, "Graphersetzung mit Anwendungen im Übersetzerbau," Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (TH), 2008.
- [14] K. Ruohonen, "Graph theory." Tampere University of Technology, 2013.
- [15] A. Schwartz, "Einführung in die Graphentheorie," *Institut für Mathematik, Julius-Maximilians-Universität Würzburg*. Würzburg, p. 123, 2013.

## Chapter 9 - Publication Bibliography

- [16] G. Hahn and C. Tardif, "Graph homomorphisms: structure and symmetry," *Graph symmetry*, vol. 497, pp. 107–166, 1997.
- [17] F. Gadducci and G. V. Monreale, "A decentralised graphical implementation of mobile ambients," *J. Log. Algebr. Program.*, vol. 80, no. 2, pp. 113–136, 2011.
- [18] K. Bettenhausen and S. Kowalewski, "Cyber-Physical Systems : Chancen und Nutzen aus Sicht der Automation," *VDI/VDE-Gesellschaft Mess- und Autom.*, no. April, pp. 1–12, 2013.
- [19] S. EVEN, *Graph Algorithms, 2nd Edition*. New York: CAMBRIDGE UNIVERSITY PRESS, 2012.
- [20] S. Kiranyaz, T. Ince, and M. Gabbouj, *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*, vol. 15. Springer-Verlag Berlin Heidelberg, 2014.
- [21] H. Chiang and C. Chu, "A Systematic Search Method for Obtaining Multiple Local Optimal Solutions of Nonlinear Programming Problems," vol. 43, no. July 1997, pp. 99–109, 1996.
- [22] M. F. Tasgetiren and P. N. Suganthan, "A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem," *2006 IEEE Congr. Evol. Comput.*, pp. 33–40, 2006.
- [23] G. P. Potdar and R. C. Thool, "OPTIMAL SOLUTION FOR SHORTEST PATH PROBLEM USING HEURISTIC SEARCH TECHNIQUE," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 3, no. 9, pp. 3247–3256, 2014.
- [24] L. Fu, D. Sun, and L. R. Rilett, "Heuristic shortest path algorithms for transportation applications: State of the art," *Comput. Oper. Res.*, vol. 33, no. 11, pp. 3324–3343, 2006.
- [25] W. W. Royce, "MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS: CONCEPT AND TECHNIQUES," in *In Proceeding of the 9th international conference on Software Engineering*, 1970, no. August, pp. 1–9.
- [26] S. Copley, "Implementing the New System," *IGCSE ICT*. [Online]. Available: <https://www.igcseict.info/theory/8/implem/>.
- [27] H. Grönniger, J. O. Ringert, and B. Rumpe, "System model-based definition of modeling language semantics," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5522 LNCS, pp. 152–166, 2009.
- [28] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide , The ( 2nd Edition ) ( Addison-Wesley Object Technology Series ) Unified Modeling Language User Guide , The Unified Modeling Language User Guide , The Many of the designations used by manufacturers and sellers to dist*, First Edit., vol. 2nd. Addison Wesley, 2015.
- [29] M. Lütjen, "Modellierungskonzept zur integrierten Planung und Simulation von Produktionsszenarien entwickelt am Beispiel der CFK - Serienfertigung," Bremen

- university, 2014.
- [30] M. Rother and J. Shook, *Learning to see: Value-Stream Mapping to Create Value and Eliminate Muda*. Cambridge MA. USA: The lean Enterprise Institute, Inc., 2003.
- [31] P. R. Berning, "Prozessmanagement und Nachhaltigkeit: Prozessorientierte Organisationskonzepte und Business Process Management." AKAD Bildungsgesellschaft GmbH, pp. 87–88, 2007.
- [32] S. S. Salunke and S. Hebbar, "Value Stream Mapping: A Continuous Improvement tool for Reduction in Total Lead Time," *Int. J. Curr. Eng. Technol.*, vol. 5, no. 2, pp. 931–934, 2015.
- [33] C. Atkinson, J. Bayer, and D. Muthig, "Component-Based Product Line Development: The Kobra Approach," in *Software Product Lines*, 2000, pp. 289–309.
- [34] A. Rausch, R. H. Reussner, R. Mirandola, and F. Plasil, *The Common Component Modeling Example: Comparing Software Component Models*, vol. 891, no. 1. Clausthal-Zellerfeld, Germany: Springer-Verlag Berlin Heidelberg, 2008.
- [35] OMG, "SysML-Modelica Transformation Specification," no. November. Object Management Group, Inc., Needham, MA 02494, U.S.A., pp. 87–106, 2012.
- [36] H. Jaakkola, "Modeling the requirements engineering process," *Inf. Model. Knowl. Bases V Princ. Form. Tech.*, vol. 5, no. July, p. 85, 1994.
- [37] Object Management Group (OMG), "OMG Meta Object Facility (MOF) Core Specification," *OMG Document Number: formal/2016-11-01*, no. 2.5.1. Object Management Group, Inc., Needham, MA 02494, U.S.A., 2016.
- [38] J. de S. Saraiva and A. R. da Silva, "Evaluation of MDE Tools from a Metamodeling Perspective," *J. Database Manag.*, vol. 19, no. 4, pp. 50–75, 2008.
- [39] Y. Rhazali, Y. Hadi, I. Chana, M. Lahmer, and A. Rhattoy, "A model transformation in model driven architecture from business model to web model," *IAENG Int. J. Comput. Sci.*, vol. 45, no. 1, pp. 104–117, 2018.
- [40] P. Stevens, "Bidirectional model transformations in QVT: Semantic issues and open questions," *Softw. Syst. Model.*, vol. 9, no. 1, pp. 7–20, 2010.
- [41] K. Czarnecki and S. Helsen, "Classification of Model Transformation Approaches," in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, 2003, pp. 1–17.
- [42] W. Bolton, *Programmable Logic Controllers*. 2006.
- [43] S. K. Boell and D. Cecez-Kecmanovic, "What is an Information System?," *2015 48th Hawaii Int. Conf. Syst. Sci.*, pp. 4959–4968, 2015.
- [44] H. Bradley, "Applied Mathematical Programming," *Appl. Math. Program.*, pp. 363–409,

1977.

- [45] M. Deynet, "Entwurf und Spezifikation einer Freisprechanlage nach V-Modell XT," Technische Universität Kaiserslautern, 2005.
- [46] C. Bartelt, A. Rausch, and K. Rehfeldt, "Quo Vadis Cyber-Physical Systems Research Areas of Cyber-Physical Ecosystems," in *Proceedings of the 1st International Workshop on Control Theory for Software Engineering*, 2015, pp. 22–25.
- [47] P. G. Larsen, J. Fitzgerald, J. Woodcock, R. Nilsson, C. Gamble, and S. Foster, "Towards Semantically Integrated Models and Tools for Cyber-Physical Systems Design," in *7th International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation.*, 2016, pp. 171–186.
- [48] J. Fitzgerald, C. Gamble, P. G. Larsen, K. Pierce, and J. Woodcock, "Cyber-Physical Systems Design: Formal Foundations, Methods and Integrated Tool Chains," in *Proceedings - 3rd FME Workshop on Formal Methods in Software Engineering*, 2015, pp. 40–46.
- [49] G. Simko, D. Lindecker, T. Levendovszky, S. Neema, and J. Sztipanovits, "Specification of cyber-physical components with formal semantics - Integration and composition," in *Model-Driven Engineering Languages and Systems (MODELS 2013)*, 2013, vol. 8107 LNCS, pp. 471–487.
- [50] A. Bhave, B. Krogh, D. Garlan, and B. Schmerl, "Multi-domain modeling of cyber-physical systems using architectural views," in *Analytic Virtual Integration of Cyber-Physical Systems Workshop (AVICPS 2010)*, 2010, pp. 43–50.
- [51] M. Tiller, *Introduction to Physical Modeling with Modelica*, Book 615. Springer US, 2001.
- [52] T. Johnson, A. Kerzhner, and R. Burkhart, "Integrating Models and Simulations of Continuous Dynamics Into SysML," *J. Comput. Inf. Sci. Eng.*, vol. 12, no. March 2012, pp. 1–11, 2012.
- [53] Y. Laghouaouta, A. Anwar, and M. Nassar, "A Formal Definition of Model Composition Traceability," *Int. J. Comput. Sci. Issues*, vol. 12(6), no. Dec. 2015, pp. 46–54, 2015.
- [54] F. J. A. Padilla, F. Weis, and J. Bourcier, "Towards a Model @ runtime Middleware for Cyber Physical Systems," in *Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing, Article No. 6*, 2014.
- [55] W. Orabi, S. M. Asce, K. El-rayes, M. Asce, A. B. Senouci, and H. Al-derham, "Orabi2009.Pdf," *J. Constr. Eng. Manag.* 135(10), no. October, pp. 1039–1048, 2009.
- [56] F. Fondement and T. Baar, "Concrete syntax definition for modeling languages," *Isim-Lgl*, vol. PhD thesis, p. 24, 2007.
- [57] M. Bodirsky, "Graph Homomorphisms and Universal Algebra Course Notes," TU Dresden, 2015.



## Chapter 9 - Publication Bibliography

- [58] D. Wang, S. Lawrenz, C. Knieke, and A. Rausch, "Innovationsverbund Synus: Methoden und Werkzeuge für die synergetische Konzipierung und Bewertung von Industrie 4.0-Lösungen," *Institute for Software and Systems Engineering in TU Clausthal*, 2018. [Online]. Available: <https://isse.tu-clausthal.de/de/forschung/laufende-projekte/synus/>. [Accessed: 15-Oct-2019].
- [59] E. Wallmüller, "Risk Management for IT and Software Projects," *Bus. Contin.*, pp. 165–178, 2002.
- [60] Barry W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Comput.*, pp. 61–72, 1988.
- [61] Bundesministerium für Bildung und Forschung, "Industrie 4.0: Innovationen für die Produktion von morgen," 2015.
- [62] R. Geissbauer, S. Schrauf, V. Koch, and S. Kuge, *Industrie 4.0 - Chancen und Herausforderungen der vierten industriellen Revolution*. PricewaterhouseCoopers Aktiengesellschaft Wirtschaftsprüfungsgesellschaft, 2014.
- [63] D. Wang, C. Knieke, and A. Rausch, "Data-driven Component Configuration in Production Systems," *Adapt. 2019 Elev. Int. Conf. Adapt. Self-Adaptive Syst. Appl.*, no. c, pp. 44–47, 2019.



## 10 List of Figures

Figure 1: Integration of development and operation of hardware/software systems .....	4
Figure 2: A path morphism example .....	15
Figure 3: A walk morphism example .....	17
Figure 4: An example of DFS and BFS in a directed graph .....	18
Figure 5: Waterfall model .....	20
Figure 6: An example of a VSM model .....	23
Figure 7: An example of a block definition diagram for a robot domain.....	24
Figure 8: An example of an internal block diagram for the RobotDomain block.....	25
Figure 9: An example of OMG's four-layer metamodel architecture .....	25
Figure 10: An example of bidirectional model transformation .....	27
Figure 11: Examples of unidirectional model transformation .....	28
Figure 12: An example of model mapping .....	29
Figure 13: A conveyor system with ASRS in an automobile factory .....	32
Figure 14: A laboratory model of a Conveyor system with ASRS as a LL-CPS sample .....	33
Figure 15: A LL-CPS modeled with VSM and IBD.....	33
Figure 16: Processes structure of the LL-CPS sample .....	34
Figure 17: A VSM model for the existing status of the conveyor system with ASRS.....	35
Figure 18: Interfaces specification .....	37
Figure 19: System decomposition of LL-CPS with BDD .....	38
Figure 20: System decomposition of the conveyor system with ASRS with BDD.....	38
Figure 21: Interface specification between the conveyor system and its system environment .....	39
Figure 22: System decomposition of process system with BDD .....	39
Figure 23: BDD of process system in the sample .....	40
Figure 24: Top view of the laboratory model .....	40
Figure 25: BDD of the mechanical system in the sample.....	41
Figure 26: IBD of the mechanical system in the sample .....	42
Figure 27: BDD of control system in sample .....	42
Figure 28: Siemens Simatic PCS7 300: Hardware assembly in control system.....	43
Figure 29: IEC 61131-3 standards-based programming languages of PLC .....	43
Figure 30: A ST program example .....	44
Figure 31: A FDB program example .....	44
Figure 32: IBD of control system in sample .....	45
Figure 33: System decomposition of Information system with BDD.....	46
Figure 34: BDD of information system in sample .....	46
Figure 35: SIEMENS SIMATIC PANEL TOUCH .....	47
Figure 36: System software and application software relation .....	47
Figure 37: IBD of information system in sample.....	48
Figure 38: System structure for a LL-CPS .....	48

## Chapter 10 - List of Figures

Figure 39: System decomposition of the conveyor system with ASRS .....	49
Figure 40: Managed evolution of a LL-CPS.....	50
Figure 41: Targeted status of this LL-CPS .....	51
Figure 42: The interfaces specification for a pilot system in car .....	52
Figure 43: The interfaces specification for the Hands-free car kit segment.....	53
Figure 44: The perspective specification for the Hands-free car kit segment in consideration only of mechanical, electrical analog and electromechanical interfaces .....	53
Figure 45: The CPS (decomposed) with smart interfaces .....	54
Figure 46: The generic modeling environment meta-model for the composition sub-language of CyPhyML.....	55
Figure 47: The conceptual relationship between system models, views and the base architecture of CPS.....	56
Figure 48: Triple Graph Grammar Formalism .....	57
Figure 49: The architecture of reconfiguration on each node of the CPS .....	58
Figure 50: VSM and IBD models on model-based description layer.....	62
Figure 51: VSM and IBD models on two description layers .....	62
Figure 52: The managed evolution of a LL-CPS.....	63
Figure 53: Cube model with eight areas .....	63
Figure 54: Models and graphs in cube model .....	65
Figure 55: Model descriptions transformations between two description layers .....	66
Figure 56: Mapping functions from VSM to IBD side.....	66
Figure 57: Evolution functions for managed evolution of LL-CPSs .....	67
Figure 58: Positions of sections into modeling system .....	67
Figure 59: Formal descriptions and descriptions transformation for VSM .....	68
Figure 60: A Metamodel for the formal semantical foundation of VSM model.....	69
Figure 61: An example for the formal semantical foundation of VSM model.....	70
Figure 62: A Metamodel for specification of VSM model by types .....	73
Figure 63: The Metamodel for connection relation in VSM model .....	74
Figure 64: A Metamodel for owned relation in VSM model.....	74
Figure 65: An example for the model-based description of a VSM model.....	76
Figure 66: Graphical representation for semantical mapping of VSM model .....	79
Figure 67: An example of semantical mapping for a VSM model.....	80
Figure 68: Graphical representation of concrete modeling for VSM graph .....	83
Figure 69: An example of concrete modeling for a VSM graph.....	84
Figure 70: Formal descriptions and descriptions transformation for IBD model .....	84
Figure 71: A Metamodel for formal semantical foundation of IBD model .....	86
Figure 72: Example of the formal semantical foundation of IBD model .....	87
Figure 73: A Metamodel for IBD Model .....	89
Figure 74: Combination rules between IS-HW model elements and ports.....	90
Figure 75: Combination rules between CS-HW model elements and ports.....	90
Figure 76: Combination rules between IS-SW model elements and ports.....	91

## Chapter 10 - List of Figures

Figure 77: Combination rules between CS-SW model elements and ports.....	91
Figure 78: Combination rules between MS-HW model elements and ports.....	91
Figure 79: Combination rules between RF-CPS model elements and ports.....	91
Figure 80: A Metamodel for connection relations between ports in IBD model .....	92
Figure 81: An example of IBD Model .....	92
Figure 82: Graphical representation of the semantical mapping for IBD models .....	94
Figure 83: Example of semantical mapping of IBD .....	95
Figure 84: Graphical representation of the concrete modeling for IBD graphs .....	96
Figure 85: Example of concrete modeling of an IBD graph .....	97
Figure 86: The mapping functions from VSM areas to IBD areas .....	97
Figure 87: An example of mapping relationship h .....	100
Figure 88: Example of mapping relationship hm .....	102
Figure 89: The managed evolution functions .....	103
Figure 90: Example of two managed evolution functions on the formal semantical foundation layer.....	105
Figure 91: Example of two managed evolution functions on the model-based description layer .....	107
Figure 92: The processes of the approach .....	110
Figure 93: Example of the start position of the managed evolution of a LL-CPS .....	111
Figure 94: Formalization of the start position with the cube model .....	112
Figure 95: Semantical mapping with concrete example.....	113
Figure 96: Generating a set of graph solutions.....	114
Figure 97: Complete directed graph K4 .....	116
Figure 98: Directed graph K4* .....	117
Figure 99: Concrete modeling of graph solutions.....	119
Figure 100: Optimizing the model solutions .....	121
Figure 101: Use case diagram for the LL-CPS managed evolution solutions generation system .....	124
Figure 102: Class diagram of the application .....	126
Figure 103: Sequence diagram of the application .....	127
Figure 104: Generating to graphs in Java code .....	129
Figure 105: The evaluation concept.....	131
Figure 106: Changes of the relevant environment factors in $m_{d,1}$ .....	132
Figure 107: Changes of hardware components in mechanical system in $m_{d,1}$ .....	133
Figure 108: Comparing software codes in control system between $m_c$ and $m_{d,1}$ .....	133
Figure 109: Comparing software codes in information system between $m_c$ and $m_{d,1}$ .....	134
Figure 110: Changes of the relevant environment factors in $m_{d,2}$ .....	135
Figure 111: Comparing software codes in control system between $m_c$ and $m_{d,2}$ .....	135
Figure 112: Comparing software codes in information system between $m_c$ and $m_{d,2}$ .....	136
Figure 113: Data driven development of a production system .....	139
Figure 114: SCL codes in the ongoing LL-CPS.....	158

Chapter 10 - List of Figures

Figure 115: New added SCL codes in the target status of the LL-CPS ..... 158  
Figure 116: New added FUP codes in the target status of the LL-CPS..... 158

# 11 List of Tables

Table 1. Graphical representation of the formal semantical foundation of the VSM model.. 70  
Table 2. The graphical representation for the model-based description of VSM ..... 75  
Table 3. Comparison table for semantical mapping for VSM model ..... 79  
Table 4. Transformations of model/relation elements in a VSM model ..... 80  
Table 5. Comparison table for concrete modeling for VSM graph ..... 82  
Table 6. Graphical representation of the formal semantical foundation of IBD model ..... 86  
Table 7. Graphical representation of the model elements in IBD model ..... 92  
Table 8. Comparison table for semantical mapping of elements in an IBD model ..... 94  
Table 9. Comparison table for concrete modeling of elements in an IBD graph..... 96  
Table 10. Mapping relationship of every vertex and edge ..... 100  
Table 11. Mapping relationship of every model and relation element ..... 102  
Table 12: Example of data structure in a VSM model..... 124  
Table 13: Example of data structure in an IBD model ..... 125  
Table 14: Example of input document for the mapping relationships ..... 125  
Table 15. The functional evaluation of two solutions..... 137  
Table 16: Example of the direct costs for reconstruction ..... 138





# 12 List of Definitions

Definition 1..... 10  
Definition 2..... 10  
Definition 3..... 11  
Definition 4..... 11  
Definition 5..... 11  
Definition 6..... 12  
Definition 7..... 12  
Definition 8..... 12  
Definition 9..... 13  
Definition 10..... 13  
Definition 11..... 14  
Definition 12..... 14  
Definition 13..... 14  
Definition 14..... 16  
Definition 15..... 17  
Definition 16..... 17  
Definition 17..... 19  
Definition 18..... 20  
Definition 19..... 26  
Definition 20..... 27  
Definition 21..... 28  
Definition 22..... 28  
Definition 23..... 29  
Definition 24..... 30  
Definition 25..... 30  
Definition 26..... 30  
Definition 27..... 64  
Definition 28..... 69  
Definition 29..... 69  
Definition 30..... 71  
Definition 31..... 72  
Definition 32..... 73  
Definition 33..... 77  
Definition 34..... 81  
Definition 35..... 85  
Definition 36..... 85  
Definition 37..... 87  
Definition 38..... 88  
Definition 39..... 89

Chapter 12 - List of Definitions

Definition 40..... 93  
Definition 41..... 95  
Definition 42..... 98  
Definition 43..... 98  
Definition 44..... 98  
Definition 45..... 98  
Definition 46..... 99  
Definition 47..... 101  
Definition 48..... 101  
Definition 49..... 101  
Definition 50..... 103  
Definition 51..... 106  
Definition 52..... 137

## 13 Appendixes

### 13.1 Pseudocodes for oppositeSearch

---

```

function oppositeSearch ( )
1: initialize: set list tempPath[ ] = null
2: initialize: set a Stack S := null
3: Set Vx.visited = true, Vy.visited = true
4: if Vy has any linked front vertex
5:   push all of the linked front vertices of Vx into S
6:   while S is not empty do
7:     if S.peek.visited = false then
8:       if m.(S.peek)  $\notin$  Gc1 then
9:         if S.peek has not linked front vertex then
10:           add S.peek into tempPath[ ]
11:           set S.peek.visited = ture
12:           save Path(opposite of tempPath[ ], Vx,Vy) in candidates[ ]
13:           delete S.peek
14:           reset tempPath[ ] =null
15:         else
16:           add S.peek into tempPath[ ]
17:           set S.peek.visited = ture
18:           replace S.peek with all of its linked front vertices in S
19:         else
20:           add S.peek into tempPath[ ]
21:           set S.peek.visited = ture
22:           save Path(opposite of tempPath[ ], Vx,Vy) in candidates[ ]
23:           delete S.peek
24:           reset tempPath[ ] =null
25:         else
26:           delete S.peek
27:       End if
28:     End while
29:   else
30:     save Path(Vx,Vy) in candidates[ ]

```

---

### 13.2 Comparing code changes

The following codes in Structured Control Language SCL in Siemens S7 implement the counting function by using of the photoelectric sensor. Figure 114 shows the codes in the ongoing LL-CPS.

## Chapter 13 - Appendixes

```
// Ist-Werte
IF Werkstueckerkannt=true THEN
  IF zahlerRot_soll <> zahlerRot_ist THEN
    zahlerRot_ist:= zahlerRot_ist +1;
  ELSIF zahlerGelb_soll <> zahlerGelb_ist THEN
    zahlerGelb_ist:= zahlerGelb_ist+1 ;
  ELSIF zahlerBlau_ist<> zahlerBlau_soll then
    zahlerBlau_ist:= zahlerBlau_ist +1;
  END_IF;

  IF (zahlerRot_soll+zahlerGelb_soll+zahlerBlau_soll) = (zahlerRot_ist+zahlerGelb_ist+zahlerBlau_ist) THEN
    Produktionsziel_erreicht:= true;
  END_IF;
END_IF;
```

Figure 114: SCL codes in the ongoing LL-CPS

In the targeted status of LL-CPS, these code in Figure 114 will be changed with the state variable “Werkstueckerkennt” from “true” to “false”. In order to meet the functional requirements, the new codes are added as in Figure 115 and Figure 116.

```
zwischenErgebnis := db151.HeadID2.FixData.colorType;
DB105.DBW[i] := INT_TO_WORD(zwischenErgebnis);

IF zwischenErgebnis = 1 THEN
  zahlerRot_ist:= zahlerRot_ist +1;
ELSIF zwischenErgebnis = 2 THEN
  zahlerGelb_ist:= zahlerGelb_ist+1 ;
ELSIF zwischenErgebnis = 3 then
  zahlerBlau_ist:= zahlerBlau_ist +1;
END_IF;
```

Figure 115: New added SCL codes in the target status of the LL-CPS

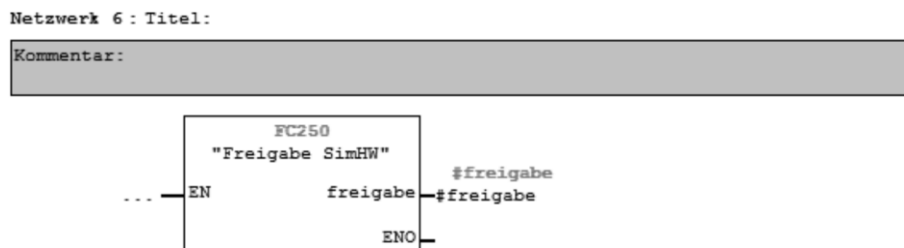


Figure 116: New added FUP codes in the target status of the LL-CPS