Otto-von-Guericke-University Magdeburg

Department of Computer Science
Institute of Technical and Business Information Systems

Masterthesis

Exchange and Integration Solutions for Heterogeneous Data in Concurrent Virtual Engineering

Author:

Yuexiao Li

May 30, 2010

Supervisor:

Prof. Dr. rer. nat. habil. Gunter Saake,
Dipl. Inf. Stephan Vornholt,
Dipl. Inf. Ingolf Geist

Otto-von-Guericke-University Magdeburg
Department of Computer Science
Postfach 4120, D–39106 Magdeburg
Germany

Surname, Firstname: Li, Yuexiao
Exchange and Integration Solutions for Heterogeneous Data in Concurrent Virtual Engineering
Masterthesis,
Otto-von-Guericke-University Magdeburg, 2010.

Acknowledgements

I want to give my best thanks to the people who have intensely helped me to finish my master thesis.

Firstly I am very thankful and appreciative to Prof. Dr. rer. nat. habil. Gunter Saake for giving me a chance to write this master thesis from which I have summarized and increased my knowledge learnt in the master stage.

Here I express my deepest appreciation and give my best thanks to my supervisor Dipl. -Inf. Stephan Vornholt for guiding me throughout the whole writing process and for the correction of this thesis. His recommendations and suggestions have been invaluable for this thesis. I thank him very much for his patience and interest in discussing with me about the topics appeared in this thesis and in answering my questions.

I give also my best thanks to Dipl. -Inf. Ingolf Geist for correcting this thesis. He gave me many good ideas to better the thesis, so he worth my best thanks here.

At last I will thank my parents Xili Li and Changhua Sun who endured this long process with me. They gave me the love, courage, strength and support I needed to achieve my goals. This thesis would not be possible without their support.

Abstract

Concurrent engineering offers a collaborative environment to the product development, while Virtual engineering offers a virtual development environment. Concurrent virtual engineering covers the features of both concurrent and virtual engineering. In concurrent virtual engineering, data exchange and integration play an important role, because collaboration between systems or engineers is of*ten required. This thesis presents different data exchange and integration solutions for heterogeneous data. Each of the solutions has benefits and weakness. To enable* the classification and comparison of existing solutions, collaboration technologies and technical approaches to handle heterogeneous data are categorized. Finally, the factors which help choosing existing collaboration systems or developing new collaboration systems will also be analyzed. [Vornholt et al., 2010]

Contents

List of Figures

List of Tables

List of Abbrevations

C		
CAD	:=	Computer Aided Design
CAE	:=	Computer Aided Engineering
CAM	:=	Computer Aided Manufacture
CCI	:=	Concurrent Component Interface
CE	:=	Concurrent Engineering
CORBA	:=	Common Object Request Broker Architecture
D		
DB	:=	Database
DBMS	:=	Database Management System
DLA	:=	Defence Logistics Agency
DMU	:=	Digital Mock Ups
DW	:=	Data Warehouse
E		
EDM	:=	Engineering Data Management
F		
FEM	:=	Finite Element Analysis Models
I		
IDL	:=	Interface Definition Language
IGES	:=	Initial Graphics Exchange Specification
ISO	:=	International Organization for Standardization

J

JECPO	:=	Joint Electronic Commerce Program Office

K

KSE	:=	Knowledge Sharing Effort
KQML	:=	Knowledge Query and Manipulation Language
KIF	:=	Knowledge Interchange Format

M

MCAD	:=	Mechanical Computer Aided Design

O

OBR	:=	Object Broker Request
OKBC	:=	Open Knowledge Base Conectivity

P

PLC	:=	Product Data Interoperability
PDM	:=	Product Data Management
PDML	:=	Product Data Markup Language
PLC	:=	Product Life Cycle
PLCC	:=	Product Life Cycle Collaboration
PLM	:=	Product Life Cycle Management
PTC	:=	Parametric Technology Corporation

S

STEP	:=	STandard for Exchange of Product model data

U

UML	:=	Unified Modelling Language

V

VE	:=	Virtual Engineering
VP	:=	Virtual Prototyping
VPDM	:=	Virtual Product Development Management
VR	:=	Virtual Reality

X

XML	:=	eXtensible Markup Language

Chapter 1

Introduction

1.1 Motivation

What is important for current product development? Every company has been thinking deeply about this question. The following aspects have important effects on the whole product development:

1. Product costs
2. Product quality
3. The time needed to bring the product to market

The increased industrial globalization is causing acceleration in the pace of product change. This trend is forcing the engineers to complete the product development process with lower product costs, better product quality, and shorter development time. If a product owns these characters, then we can say that the product development process at this time is successful. All the companies make every effort to achieve this goal.

In current engineering there are two types of modes for the product development process, including linear engineering and concurrent engineering (shown in Figure 1.1).

- Linear engineering: The complete development process is executed step by step according to the order of time. The tasks in each step are developed consecutively.
- Concurrent engineering: Different tasks in a step are executed concurrently. While executing the first task the second task can be carried on at the same time. Dependencies exist between these two tasks. If the first task needs the data from the second task, it stops carrying on and gets the needed data from the second task. A dependent work process can also be stopped when necessary information is not available.

In present times linear engineering has been nearly replaced by Concurrent Engineering, because Concurrent Engineering improves the product development process and satisfies the requirements of producing the successful products. Chapter 2 will explain these reasons in detail .

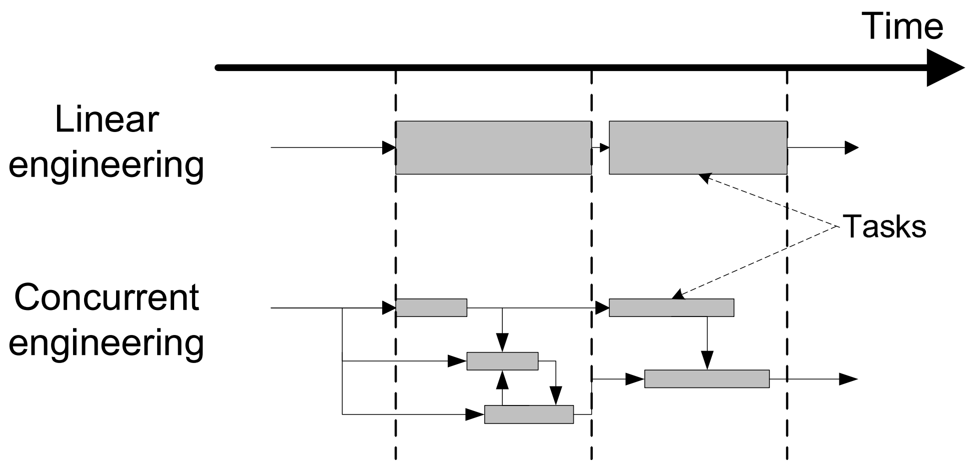


Figure 1.1: Two types of engineering

Besides Concurrent Engineering, Virtual Engineering is also a very meaningful term to satisfy the goal of producing successful products. Before computers are applied in industrial manufacturing, manual manipulation caused a lot of trouble. It prolongs the production time. Since without the accurate design and analysis of the computer for the product, product costs are hard to cut down and the product quality could not be ensured. As the computer technologies have been rapidly developed, the tools for such as dynamic visualization, 3-D designs or other simulations are developed. These tools in the virtual computerized environment are more and more embedded in the industrial manufacturing. This application is making a big progress in the industrial manufacturing. We could also say that Virtual Engineering facilitates the product development processes. Based on Virtual Engineering companies find more ways to get closer to the successful product.

Figure 1.2 illustrates an example for production in Virtual Engineering. It is a virtual factory, in which the whole production process is simulated. The virtual design of a product model is constructed with the aid of computer aided design or other tools.

Concurrent Virtual Engineering is generated by introducing the virtual design, virtual simulation, or other virtual manufacturing elements to Concurrent Engineering.

Since in Concurrent Virtual Engineering many different design tools are applied to heterogeneous systems model designs, the heterogeneity of communication and collaboration of different systems becomes avoidable. The communication and collaboration include data exchange and integration between different product data models. There are two situations in which data exchange and integration are necessary.

- Different design tools are used to design models in one domain. These models are represented by different formats through these design tools. Solutions are needed to transfer the data from one format to others or to translate all the formats of models into one. For example, when two computer aided design (CAD) systems are used to design a geometry model, the transformation between these two formats is necessary.

- Different models are used to design one product in different domains. These models can be designed at the same time, but they are not isolated. Dependencies exist

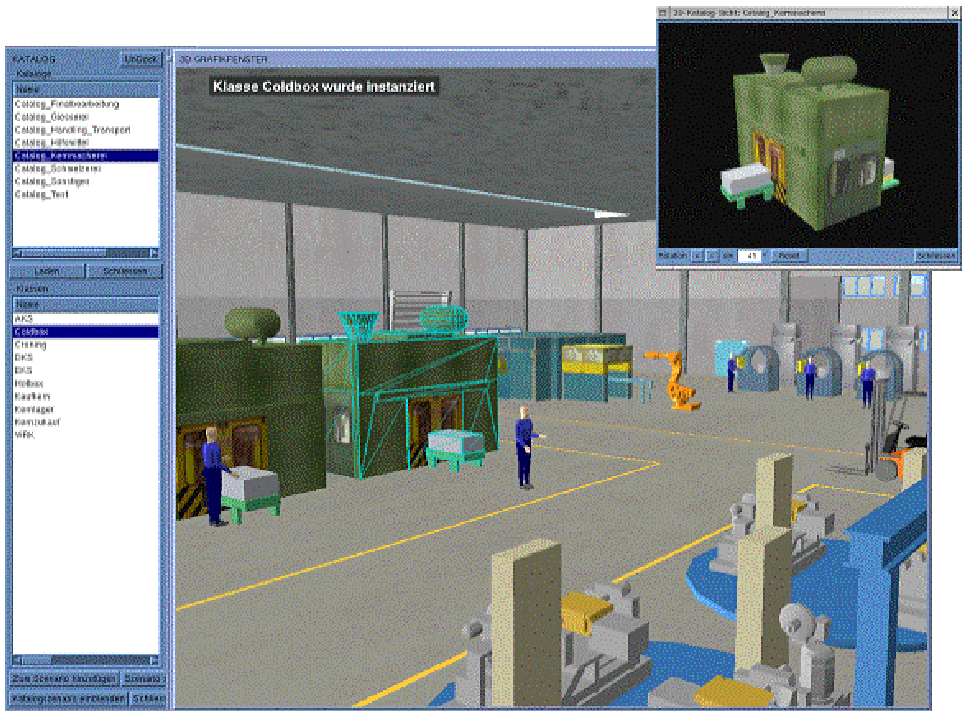


Figure 1.2: An example of virtual engineering

between these models, therefore, it is necessary to exchange and integrate the data between these models.

Good communication between heterogeneous systems plays an important role in Concurrent Virtual Engineering. Good solutions for data exchange and integration improve tight relationships between different systems and make the collaborative work more efficient. Collaboration brings a huge progress in cutting cost, improving quality and reducing the time to market. Many engineers have been concentrating on finding more practical and easily manipulated solutions for data exchange and integration. For example, Initial Graphics Exchange Specification is used to exchange data formats.

Therefore, the research of finding good solutions for heterogeneous data exchange and integration in Concurrent Virtual Engineering is very important.

1.2 Goals of the Thesis

The goals of the thesis are divided into three parts:

- Goal 1: In this thesis concepts and the impact of Concurrent and Virtual Engineering should be described. According to the representations of different modelling methods the simple product models can be designed.

- Goal 2: Solutions for heterogeneous data exchange and integration in state of the art in the research are surveyed and analyzed. After the conclusion of these solutions, they will be classified and the representation of the solutions using a specific model as the example will be described.

- Goal 3: The factors which have an effect on the classification of different solutions of data exchange and integration will be illustrated. As a result, the practical decisions will be figured out according to the defined factors.

1.3 Structure of the Thesis

After the introduction the structure of remaining chapters is organized in the following:

- Chapter ***Fundamentals***: In this chapter the fundamentals for this thesis are introduced so that readers could understand the following chapters more easily. That is, we present the basic concepts of Concurrent Engineering, including its backgrounds, definitions, implementing methods and benefits, as well as the data management in Concurrent Engineering. Another important concept is Virtual Engineering. So, in the following we introduce the definition and process of Virtual Engineering as well as its branches: virtual reality and virtual prototyping. Concurrent Virtual Engineering is illustrated at the end of this chapter along with the relationships between Concurrent Engineering and Virtual Engineering.

- Chapter ***Product Data Modelling***: This chapter presents different methods of constructing models, geometry modelling for example, which is most commonly described by CAD systems. At the end, an example based on the concept "Light Bulb" is presented to describe the process of modelling in detail, where several models like the geometry model, the material model, the thermal model and the light model are built. Besides, the relationship (or their dependencies on each other) between them are defined during the construction of the models. This helps explaining the data exchange and integration introduced in the next two chapters.

- Chapter ***Product Data Exchange***: Firstly, this chapter motivates data exchange in Concurrent Virtual Engineering as one solution. Then it introduces a few of strategies for data exchange. As fundamentals, several existing standards of data exchange like Initial Graphics Exchange Specification (IGES) and STandard for Exchange of Product model data (STEP) are presented in the later. After that another key point of this chapter, how to perform the data exchange between models, is illustrated with the help of the "Light Bulb" models. At the end of this chapter, the problems during data exchange are discussed.

- Chapter ***Product Data Integration***: Like the previous chapter, firstly this chapter explains, what data integration is and why it is needed in Concurrent Virtual Engineering. In the following, the basic architectures of data integration are interpreted to explain its principle. Based on the architectures, this chapter presents the solutions to perform data integration. After that, we talk about the product data management (PDM) systems which are used to manage data in the whole product development process. The Light Bulb models are also taken as an example to analyze the data integration.

- Chapter ***Classification of Data Exchange and Integration Solutions***: This chapter firstly summarizes and lists several categories to classify the data exchange and data integration solutions. After that, the solutions for data exchange and data integration are discussed and then classified according to the defined categories. In addition, solutions provided by STEP and PDM systems are taken as examples and classified according to the previous defined categories. This chapter ends with a short discussion about the advantages of the exemplary classified applications.

- Chapter ***Analysis of Data Exchange and Integration Solutions***: This chapter begins with an introduction to decision trees, where the definition, the usage and the advantages as well as the weaknesses of decision trees are interpreted. In the following, this chapter analyzes the decisions of data exchange and integration solutions according to practical requirements in the product development by using decision trees.

- Chapter ***Conclusion and Further Works***: This chapter gives the conclusion of this thesis and the outlook to further works.

Chapter 2

Fundamentals

This chapter introduces the fundamentals of Concurrent and Virtual Engineering. In Section 2.1 relevant information about Concurrent Engineering (CE) is presented, including its background, analysis of different definitions, the implementing methods, the benefits of implementing Concurrent Engineering and the data management solutions. Section 2.2 explains Virtual Engineering (VE) and describes the VE process and two components of VE (Virtual Reality and Virtual Prototyping). Section 2.3 presents the relationship between Concurrent and Virtual Engineering.

2.1 Concurrent Engineering

2.1.1 Product Life Cycle

In CE the Product Life Cycle (PLC) is a very important term. It presents a complete product development process. The advantages of the product life cycle concept is that it provides a basic structure that allows you to see where you are, and what lies ahead. Based on minds of [Dyla, 2002], [Eastman et al., 1991a] and [Eastman et al., 1991b], six phases are formulated in PLC (see Figure 2.1):

- Planning: After determining the purpose of the product a plan for the whole development process is drawn up.
- Requirements: The engineers get the requirements from the market and the customers' feedback.
- Design: The design phase includes conceptual and detailed design. The conceptual design specifies the intended functions and properties. The detailed design complements the details to conceptual design. After the detailed design we can get relative complete structure and function models. At last these models need to be analyzed whether they conform the design standards.
- Manufacture: Based on the design phase the actual products are manufactured.
- Test: The products must be constantly tested to make sure they correspond to the standards and requirements. If the products are accepted, they will be brought into the market. Otherwise the work should go back to the previous phase.

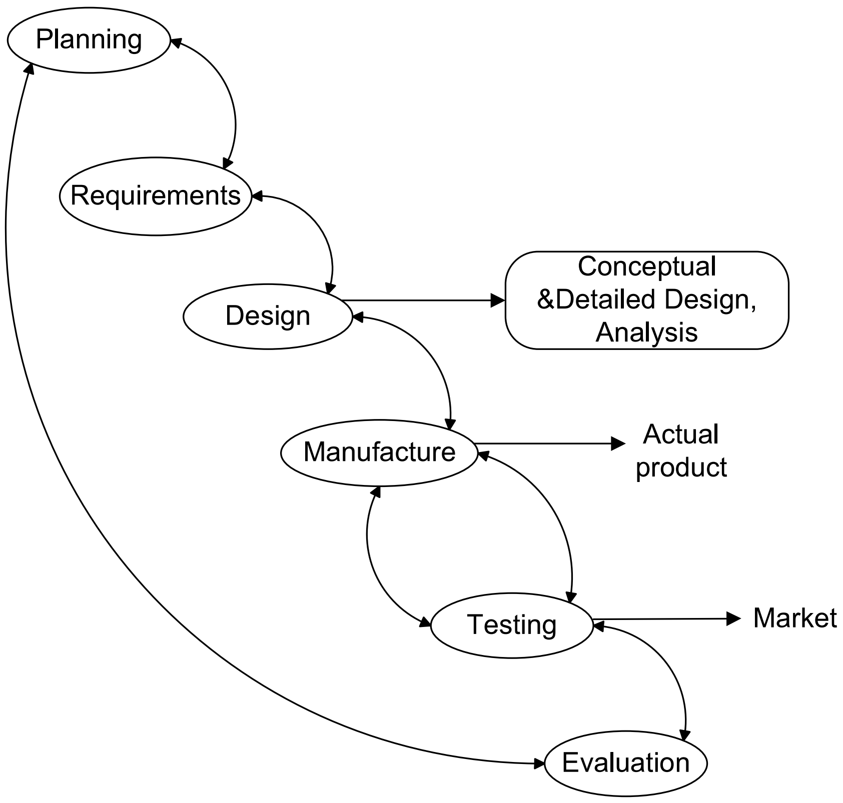


Figure 2.1: Product life cycle

- Evaluation: The evaluation work should be done after the final production. The evaluation result is offered as the experience knowledge to the first phase.

2.1.2 Concurrent Engineering Background

Before Concurrent Engineering emerged, linear engineering was used which means that different phases of product development are carried out in series. Why is linear engineering not enough in current engineering ? There are three reasons to explain this question.

- Rapid development of new technologies: Technologies have been always developing in an increasing speed. Some companies which have used new technologies make the time from the initial idea of the product to the market shorter and gain more market share. This competition causes big pressure to other companies. They are not content to fall behind, so they are pursuing new ways of development in order to shorten the time-to-market. The long time product development is eliminated.

- Isolation between the product research and development and product design areas: This causes a lack of integration with the general strategy of the business. This is increased by the different languages and understandings of design problems used by different design areas ([de Andrade and Forcellinib, 2007]).

- Cost of drawbacks: In linear engineering, the whole developing process is executed one by one according to the order of time. So, shortcomings of the product are not found until at the end of the developing process. Modifying the mistakes at this moment costs much. Sometimes a product needs to be redesigned and this kind of loss is always not small. Using CE could cut down the cost of drawbacks. The engineers that come from different departments work together. They can find and solve the problems as early as possible.

2.1.3 Definitions of Concurrent Engineering

Many different definitions of CE exist. They illustrate CE from different angles and show many facets. Simultaneously this multiplicity of the definitions reflects both strength and weakness. The strength is that CE is a type of general engineering which combines many different approaches. The weakness is that there is not a uniform understanding about what actually constitutes CE and what are the principles of CE. In this sense Canty ([Canty, 1987]) points that CE is both a philosophy and an environment. This understanding of CE as philosophy is also described in [Jo et al., 1993].

The Concurrent Design Facility [1] defines CE as following:

"*Concurrent Engineering (CE) is a systematic approach to integrated product development that emphasizes the response to customer expectations. It embodies team values of co-operation, trust and sharing in such a manner that decision making is by consensus, involving all perspectives in parallel, from the beginning of the Product Life Cycle.*"

It is defined from the customer expectations. It emphasizes the co-operation work among the designers and the experts in the product life cycle. The final decision must be arrived to consensus no matter how different the insights from different departments are.

The CE definition from Pennell and Winner ([Pennel and Winner, 1989]) is considered as a standard description:

"*Concurrent Engineering is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life cycle from conception through disposal, including quality, cost, schedule, and user requirements.*" This definition states a key aspect of CE - the parallel work of multidisciplinary teams.

Features of Concurrent Engineering

'Concurrent' is explained as 'occurring side by side' and 'acting in cooperation' in [Webster, 1989]. These two aspects can be called **parallelity** and **collaboration**. Figure 2.2 presents the comparison between linear engineering and CE. This thesis focuses on the design phase, so the figure shows the design phase (including conceptual design, detailed design, and analysis) in detail. Compared to linear engineering CE saves the developing time depending on its features parallelity and collaboration.

- **Parallelity**: The different phases in the whole product development process are accomplished in parallel, reducing the total time.

[1]The Concurrent Design Facility is the European Space Agency main assessment center for future space missions and industrial review.

- **Collaboration**: This term is always illustrated as the participants of the collaborative work. Different engineerers in different departments need to communicate to each other and exchange the information.

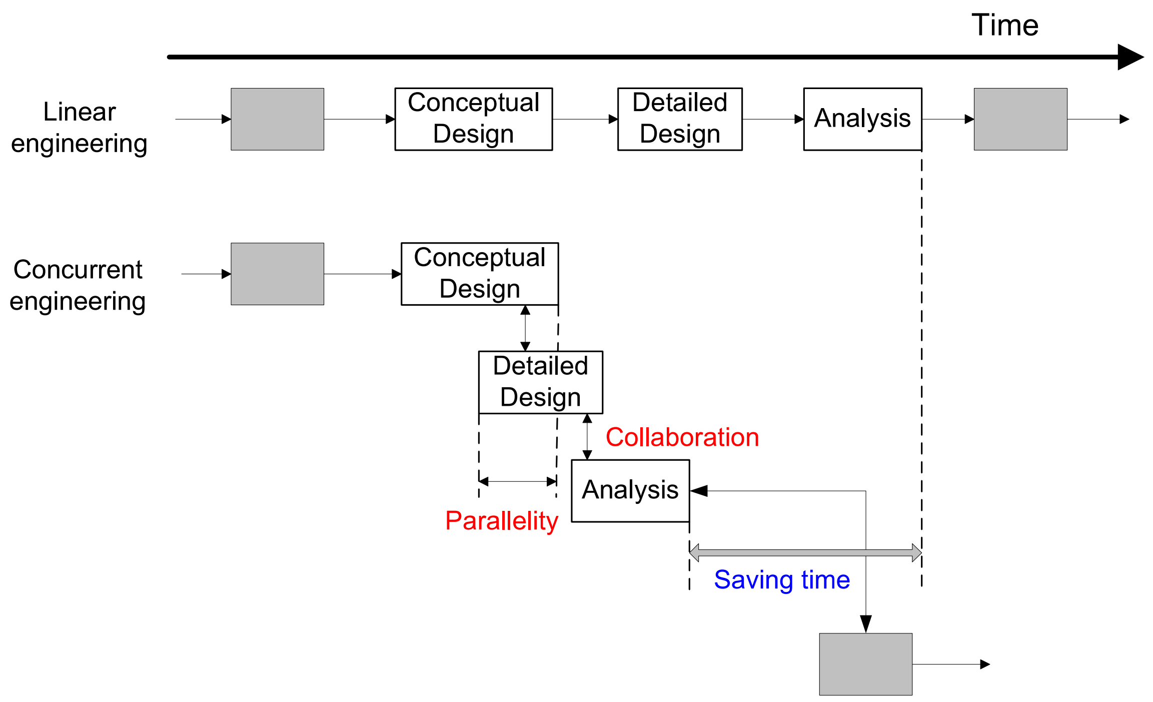


Figure 2.2: Comparison of linear engineering and concurrent engineering

2.1.4 Implementation Methods of Concurrent Engineering

Implementation methods of CE are based on the features of CE. Parallelity and collaboration should be totally materialized in these methods. The factors which may appear in the PLC should also be considered in implementation methods ([Maletic et al., 2001]). We should make the effort from the following aspects to implement CE:

- Teamwork: At the beginning of the design the personnel involved in the whole PLC should be summoned together to determine the product design plans, to evaluate the design programs and to get an optimal result. This approach collects different experts even including potential users together in a special group, in order to design a product which is easy to assemble, easy to maintain, easy to recovery, easy to use. This way of working from a large extent overcomes the shortcomings of the original linear production mode. In the past it was hard to take the requirements in the different phases of production into account because of the limitations in the aspects of designers' knowledge and experiences. Besides, the complexity and variety of the equipment, technology and material causes the difficulty of evaluation and chose to many design programs. In tight situations, the designers choose the most convenient program but not the most appropriate program. So the phenomenon of rework is unavoidable. ([Newman et al., 2003])

- Concurrent management of the design process: Technology platform and the experts from different departments are the foundation of CE. CE is the concurrent development based on the collaboration of the experts. However, producing benefits depends not only on technology platform and the experts but also on the effective management of the process. Since each expert is constrained by the professional knowledge, he always considers some factor more and ignores the overall index of the product. Therefore establishing a comprehensive design plan requires repeated communication, information exchange, and consultation. In the design process, the team leader should organize the discussion at regular or irregular intervals. The team members speak their minds and make the inspection on the product which is full or half designed in order to strive for a product which satisfies requirements as many as possible. The integrated indicators should achieve a satisfactory value. ([Estublier and Vega, 2007])

- Emphasis on the systematic design process: CE considers design, manufacture and management as an integrated system. The design process includes not only the drawings and other design information but also the quality control, cost accounting and produced progress plans ([F-L et al., 1994]). For example, in the design process other processes including assembly process and testing process can be simultaneously carried on.

2.1.5 Benefits of Implementing Concurrent Engineering

Implementing CE brings the companies many obvious benefits.

- Reducing the time-to-market: The customers are demanding shorter delivery time. Nowadays, the products are served in short supply. Customers mainly consider the function of the products. They require the perfection and practicality of the function. With the development of manufacturing technology, there are abundant goods. At this time the customers pay attention to the product price. When the manufacturers reduce the cost into a certain extent through lean production, the customers pay attention to the product quality. The market development indicates that shortening the delivery time will become the main feature of the next phase. The role of CE technologies is to reduce the time of product development and preparation.

- Reducing costs: CE can reduce costs in three aspects. Firstly, it can limit the mistakes in the design process. The later in the PLC the mistakes are found, the more the production costs. Secondly, CE is different from the traditional methods such as "repeated trial". It emphasizes that we must achieve the purpose at one time. This purpose is achieved according to the software simulation and Virtual Prototyping. It eliminates the expensive prototype trial. Thirdly, since the processing, assembly, testing, maintenance, and other factors are taken into account in the design process and the overall optimization of the product cost is highlighted, the product cost in the whole PLC is lower. Not only the customers but also the manufacturers benefit from the lower costs.

- Improving the quality: Using CE technologies can eliminate the quality problems in the design process, so that the designed products are easy to manufacture and maintain. Before offered to the market, the testing time of the products manufactured in this way becomes much shorter than in linear ways. In fact according to the theory of modern quality control the quality is first designed and then manufactured. Testing cannot improve the quality. It can only eliminate waste products. CE technologies fundamentally guarantee the quality.

- Enhancing usability of functions: In CE the salesmen or the customers can also participate in the design process. So this design method is close to the market trends and reflects the users' needs.It can also remove redundant functions not needed by the customers, reduce the complexity of equipments, and improve the reliability and practicality of products. CE enhances the enterprises' competitiveness. CE can relatively quickly introduce the products and put them on the market. At the same time the reasonable design process cuts down the costs and ensures the product quality. Therefore, the competitiveness will be strengthened.

2.2 Virtual Engineering

2.2.1 What is Virtual Engineering?

Virtual Engineering (VE) is a powerful concept, defined as *a technology that integrates geometric models and related engineering tools such as analysis and simulation, optimization and decision-making tools, etc. within a computer generated environment that facilitates multidisciplinary and collaborative product realization.*

This definition emphasizes that VE is an emerging technology. Many tools (e.g., simulation, Virtual Reality tools) are applied in VE to make the whole product development process more efficiently and more accurately.

Jian describes Virtual Engineering in [Jian et al., 2006] as *user-centered process that provides a collaborative framework to integrate all of the design models, simulation results, test data, and other decision-support tools in a readily accessible environment.*

'Integration' is the core idea of this definition. Different data models, including CAD models and other computational models are needed to be integrated in the virtual environment. VE provides a share virtual environment, enhancing communication at all levels in the PLC.

2.2.2 Virtual Engineering Process

In last subsection we have talked about what is Virtual Engineering. Now another question comes: how is it processed? Which steps should be followed in the Virtual Engineering? To this question, Bullinger [Bullinger, 2002] gave his answer as following (shown in Figure 2.3):

1. **Data creation**
   On this layer the computer aided data, Engineering Data Management (EDM) and PDM data are created and saved with the software. At the same time documents

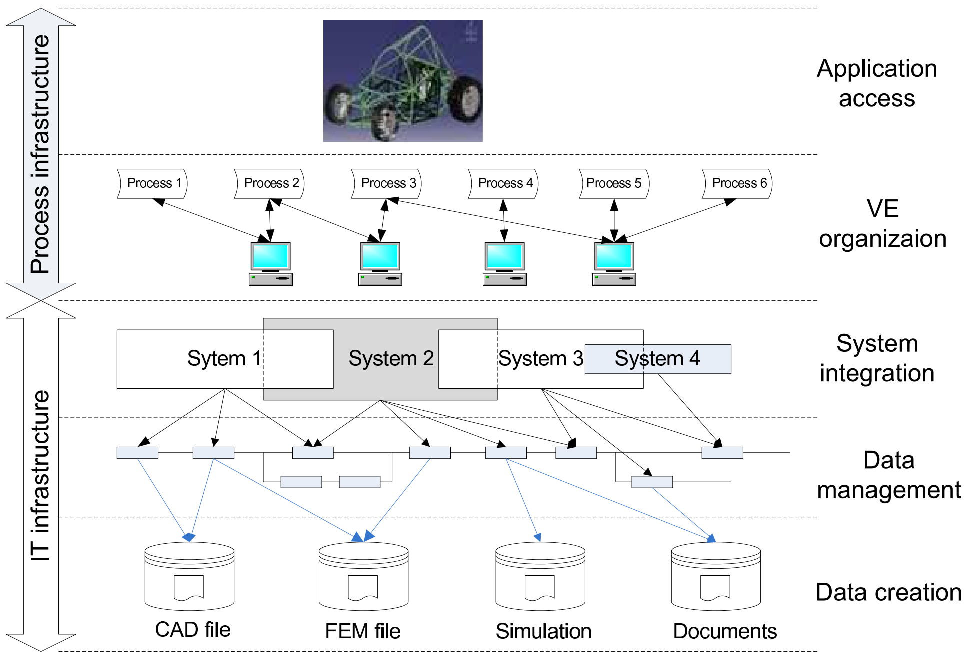
for describing and managing of the data are also created. For example, geometry data is created and will be used as the foundation of the following operations in the whole process.

2. **Data management**
   On this layer the data created on the layer "data creation" will be handled by data management systems. For example, on this layer the PDM systems store, transfer and process the information about used applications and processes.

3. **System integration**
   This layer includes data integration as well as the connection to applications. All applications, which are required by users, must be installed and integrated in the system. Data exchange in the compatible modules should be bidirectional, fast and easy, e.g. from CAD to Virtual Reality (VR) simulation as well as reverse.

4. **VE organization**
   This layer comprises integrated process management. This includes controlling the communication, managing the finite (qualitative and quantitative) resources, handling projects and processing risks and difficulties as well as tracking project advancement.

5. **Application access**
   This layer comprises communication of costumers, cooperating companies and external engineering partners. However, integration of new domains and management of complexity are also affected by status control, portals, and access control.



Adapted from: [Bullinger, 2002]

Figure 2.3: The five layers of virtual engineering

2.2.3 Virtual Reality

Virtual Reality (VR) is an application of simulated environment. It simulates the objects of the real word by using the simulation tools. VE used VR in the designs of product models. For example, VR helps engineers to develop CAD systems.

Virtual Reality is described as "a high end user interface that involves real-time simulation and interactions through multiple sensorial channels. These sensorial modalities are visual, auditory, tactile, smell, taste, etc." ([Burdea and Coiffet, 2003])

VR allows its users to intuitively interact with the virtual environment and its objects as if they were real by immersing them in this three-dimensional computer generated simulation. This method of conveying information promotes understanding of complex systems even in people with limited past experiences or knowledge about that system. ([Shukla et al., 1996])

VR provides new possibilities to reduce the time needed from the first idea of a product to its mass production. It makes new simulation methods available and reduces costs without lowering product quality. Therefore, Virtual Reality technologies also become indispensable to the manufacturing sector. ([Bernard, 2005])

2.2.4 Virtual Prototyping

Virtual Prototyping (VP) technologies are being widely used in current industry, because they facilitate VE by making the communication among various engineers easier, identifying the problems early in PLC, and provide the cross-functional evaluation at a lower product development cost. ([Cecil and Kanchanapiboon, 2007])

In engineering a prototype is a full-scale model of a structure or piece of equipment, used in evaluating form, design, fit, and performance. It represents important features of a product, which are to be investigated, evaluated, and improved. It is used to prove design alternatives, to do engineering analysis manufacturing planning, support management decisions, and often just to show a product to the customers.

"Virtual prototype is a computer simulation of a physical product that can be presented, analyzed, and tested from concerned PLC aspects such as design/engineering, manufacturing, service, and recycling as if on a real physical model. " [Wang, 2002]

VP is a modelling approach which uses Virtual Reality technologies to realize the creation, presentation, test and evaluation of a virtual prototype. It is an effective method for collaborative simulation and verification of product design information in virtual environment.

The advantages of using VP:

- Low costs and fast speed: VP saves the expensive costs spent on physical prototype. It takes shorter time using VP in computer than constructing physical prototypes.
- Easy to design optimization: It is easy to modify virtual prototype. We could compare different design schemata by using VP and then choose a optimal design schema.
- Good for product manufacture: According to VP we could get the accurate and fine product designs, so the data from virtual prototype can be directly put into

product manufacture.

2.3 Concurrent Virtual Engineering

In this thesis Virtual Engineering is focused on the data creation process. The data creation process belongs to the design phase of Concurrent Engineering. Using virtual tools to finish the design work of the product identifies the mixture of Concurrent and Virtual Engineering. **Concurrent Virtual Engineering** is used as a term in this thesis.

The geometry model is the base for other models. It includes graphical representation and dependencies of subparts as well as constrains and domain descriptions. This model is converted into a CAD model. Other domains (e.g., light and thermal models) are successively added. The connection between different elements is defined in each domain. The resulting virtual prototype can be converted into different simulation based formats (e.g., FEM simulation and mechatronic simulation). In each of these phases, standards, and libraries with existing products are used to increase correctness ([Whyte et al., 2000]). Changes on the intended product, like variants or versions require a re-design of the product.

2.3.1 Design Types in Concurrent Virtual Engineering

As introduced in Section 2.1.3 one feature for CE is parallelity. This feature is also appropriate in Concurrent Virtual Engineering. Not every process is executed in parallel and not every parallelity is executed in the same degree, because the information flow within the processes and the relationships of the processes with each other are different. A process can be executed in sequential or in parallel ([Pop et al., 2004]). Which choice is appropriate depends how the steps of a process are related to each other.

The detailed design process is classified into three categories (linear, concurrent and simultaneous design) in Concurrent Virtual Engineering ([Hamri and Léon, 2004]). As shown in Figure 2.4, there are four different models to be designed in different ways, including model 1, model 2, model 3 and model 4. **Bidirectional arrows** present the information exchange. Three types of design can be chosen to accomplish the design phase.

- Linear design process: The design process is executed in sequence. In linear design, model 1 and model 2 have no dependencies with each other. Model 1 is designed first and then model 2 is designed. After the designs of model 1 and model 2, these two models may exchange information, or they need a composition. This way of execution hat two probability:
  - One model design starts after it receives the entry signal from another model design.
  - All entry signals of model design processes come from the only one of the other processes.
- Concurrent design process: It means that the design process is executed in parallel. It indicates no dependency between the parallel design processes or a dependency

which is unconsidered. Figure 2.4 shows that in concurrent design model 1 and model 2 are designed in parallel. During the design processes for both there is no need to exchange data with each other. In some situations these models need a final composition. Concurrent design is a special form of simultaneous design ([Kawabata et al., 2002]).

- Simultaneous design process:
  - Partial parallel: It means that the design process is executed partial in parallel. In this situation the dependency of the design process for the parallelity is determined. The degree of the parallelity is calculated according to the dependency ([Prasad et al., 1997]). For example, when 70 percent of one model depends on another, the degree of the parallelity is only 30 percent.
  - Close parallel: For two or more models they have so close relationships with each other that they have to exchange information each time.

  Figure 2.4 gives an example for simultaneous design. Model 1 and model 4 have a close relationship. They need to exchange data each time during the design process. Model 1 has dependencies to model 2 and model 3. After the design process of model 1 is executed partly, this process should be stopped. Model 2 and Model 3 designs start. So, we can say, that the design processes of model 1, model 2, and model 3 are executed partly parallel. Model 1 receives the needed data from model 2 and model 3. Simultaneous design is more complex than linear and concurrent design.

After the detailed design these design models will be simulated with the help of virtual methods which will be described in Section 2.3.2. The analysis for detailed design and simulation is carried out through the whole design process.

2.3.2 Virtual Prototyping and Concurrent Engineering

Virtual Prototyping is an important method which realizes CE. Because concurrent/simultaneous design support both multiple processes integration and multiple product development teams, Virtual Prototyping offers support to CE from these two aspects:

- Virtual Prototyping and concurrent/simultaneous design for multiple processes integration: During the structural and functional design, the physical plan or the virtual physical design is carried on. The factors which affect the product quality, the costs, and life cycle must be overall and concurrently identified ([Dai et al., 1996]). The reasonable constrains which are based on the verification of the structural and functional design and the concurrent/simultaneous design are generated to regulate the virtual physical design. Figure 2.5 is a design flow diagram using VP. The virtual prototype of the product is generated according to the historical data after finishing the structural and functional design and verification. Virtual prototype includes the behaviors of system structures and physical design information. Based on virtual prototype, concurrent/simultaneous design considers different aspects to make simulation and tests on virtual prototype, to analyze physical parameter information, to determine whether the performances are met

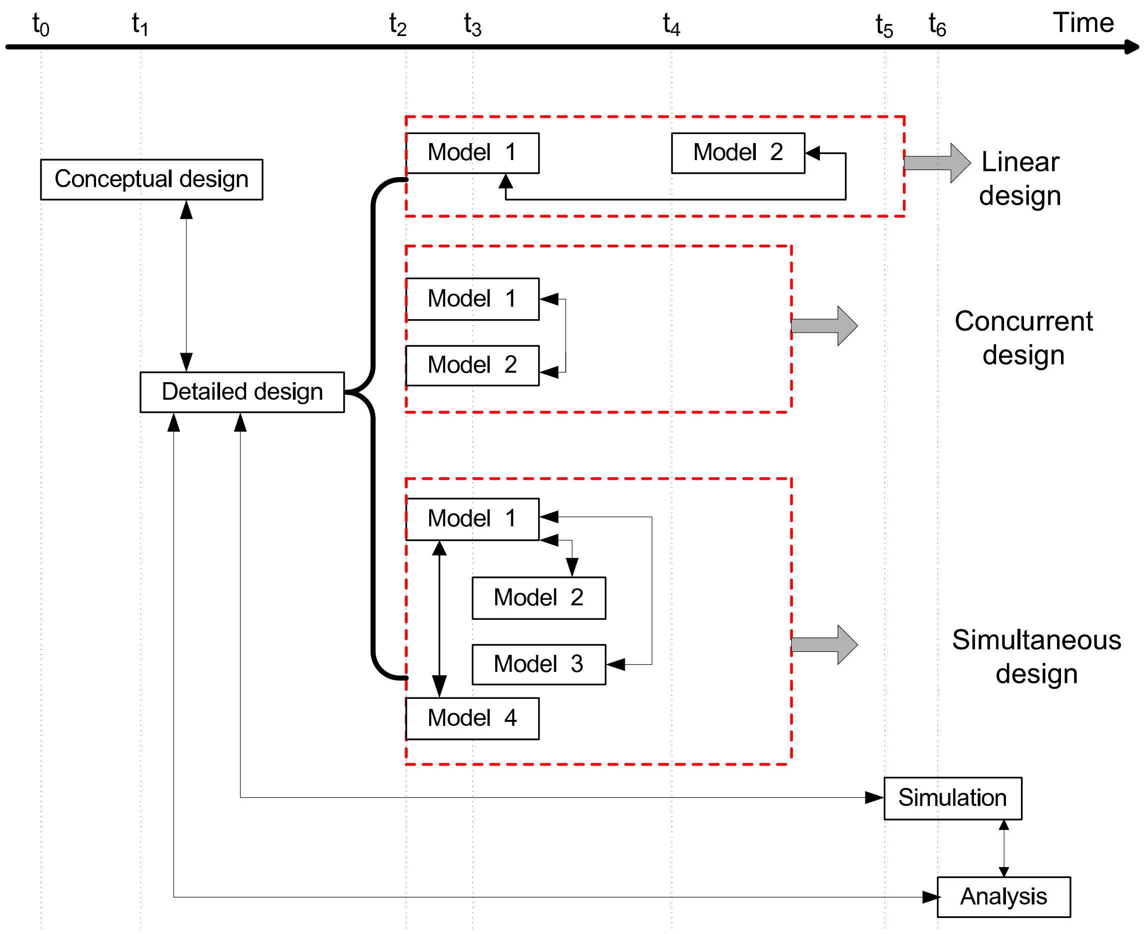


Figure 2.4: Different design types in concurrent virtual engineering

and design is reasonable. Finally, the evaluation of the product realization should be given. If the performances or the variety of constraints are found to be missed, then the corresponding modifications are proposed in order to get a new virtual prototype design plan. If not, a optimal physical plan will be established according to previous concurrent/simultaneous design.

- Virtual Prototyping and collaborative concurrent/simultaneous design for multiple product development teams: The product design space can be seen as a multidimensional space. Each dimension points to a constraint factor which may affect the PLC, e.g., the performance, the costs, manufacturability, and maintainability. These constraint factors affect each other ([Hislop et al., 2004]). To some dimension, some product design plan may be good, but due to other constraint factors this product design plan may become an unfeasible plan. In the whole design process, because of the constraint on knowledge and object wish the experts in each field always consider some aspect of the product too much and ignore the global performance of the product. To complex products, the collaborative design from various experts is unavoidable. The evaluation of the product design is mostly uncertainty, so using VP is an effective method to solve the collaborative design from various experts. In the collaborative design environment of different product development teams the experts who come from different locations and departments test, simulate, evaluate, modify the virtual prototype from different angles and requirements. Through VP they can reach purposes to communicate and share information to each other.

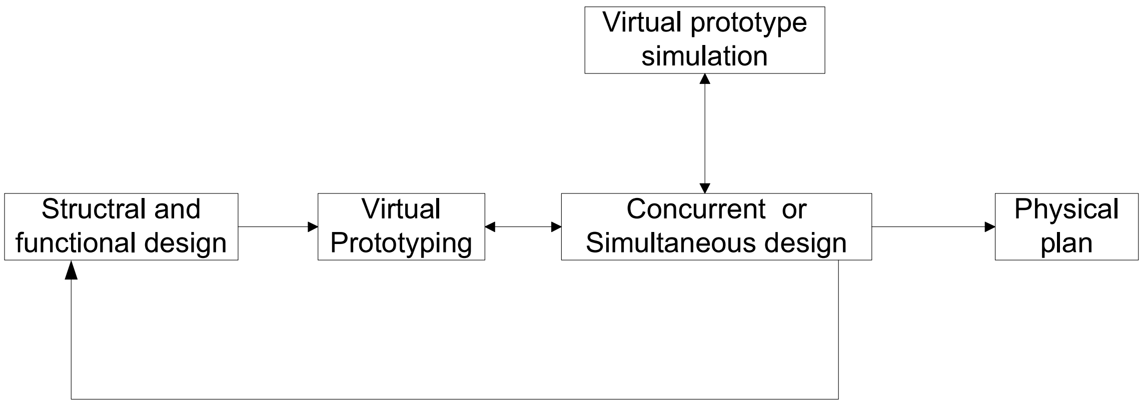

Virtual prototype simulation
Structral and functional design
Virtual Prototyping
Concurrent or Simultaneous design
Physical plan

Figure 2.5: Design flow diagram using virtual prototyping

Chapter 3

Product Data Modeling

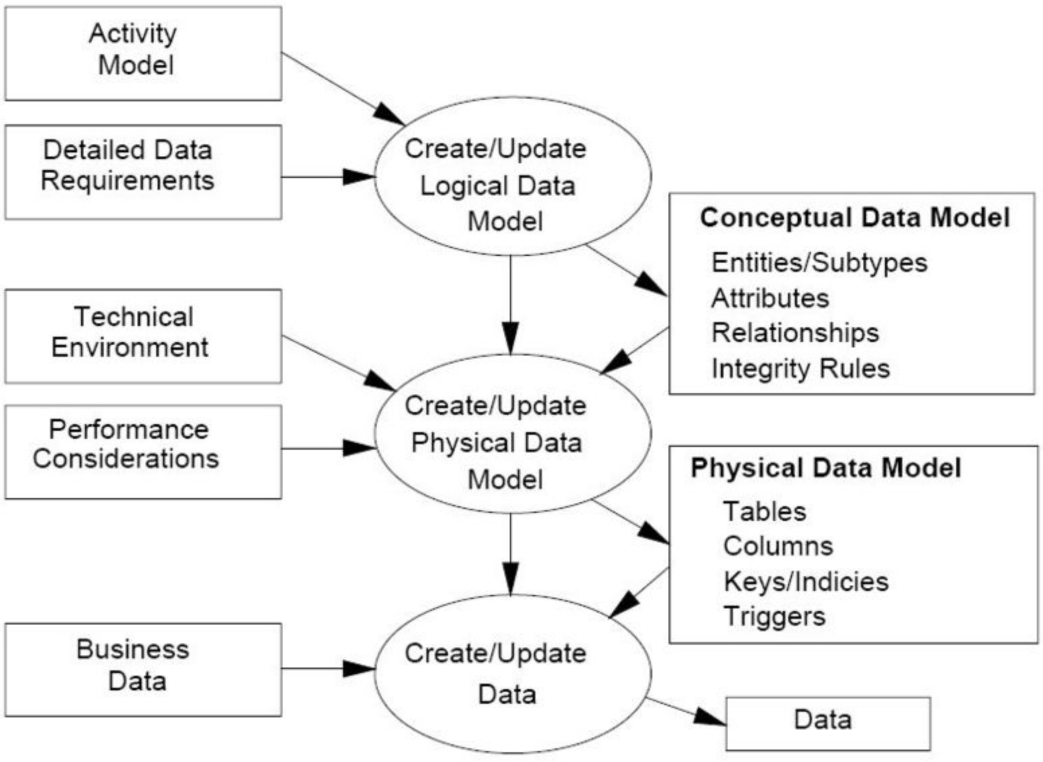
What is data modeling? Data modeling is a method used to define and analyze data requirements needed to support the business processes of an organization ([Whitten, 2004]). In software engineering it is defined as the process of creating a data model by applying formal data model descriptions using data modeling techniques ([Whitten, 2004]). Generally speaking, the goal of data modeling is to create data models.

Figure 3.1 illustrates the way data models are developed and used today. A conceptual data model is developed based on the data requirements for the application that is being developed, perhaps in the context of an activity model. The data model will normally consist of entity types, attributes, relationships, integrity rules, and the definitions of those objects. This is then used as the start point for interface or database design. ([West and Fowler, 1999])

What is product data modeling? Product data modeling is a technique for defining business requirements needed in a production, in other words, for creating product data models. Product data models are semantic models of a product. They can be found in computer-aided engineering applications. They cover the product geometry as well as functional, technological and other features. They are initially used as integration models, collecting and processing the formal, factory specific procedures of computer aided product development and manufacturing. Product data models contain both aspects of the product (application layer) and aspects of the implementation (physical layer). As an intermediate layer, a logical specification (logical layer) is introduced to secure an independent, formal representation. Due to existing, company-specific structures and the resulting complexity and size of data, a product data model comprises a set of partial models. ([Gabbert and Wehner, 1998])

3.1 Types of Data Models

To store the information of product data models, the types of data models presented in this chapter are used. A data model is an abstract model that describes how data are represented and accessed. Data models formally define data elements and relationships among data elements in a domain. According to [Hoberman, 2009], "a data model is a wayfinding tool for both business and IT professionals, which uses a set of symbols and



Source: Developing High Quality Data Models [West and Fowler, 1999]

Figure 3.1: Data modelling process

text to precisely explain a subset of real information to improve communication within the organization and thereby lead to a more flexible and stable application environment."

According to [Hoberman, 2009], there are different types of data models. However, only two types of models, which are meaningful for this thesis, are introduced in the following subsections.

3.1.1 Database Model

A database model is a theory or specification describing how a database is structured and used. Several such models have been suggested. According to [Beech and Feldman, 1983], common models include the following ones, which are shown in the figure 3.2:

- **Hierarchical model**
  The hierarchical model organizes data as a rooted tree of records. Each record has one parent record and many children. No cycles exist in this tree.
- **Network model**
  The network model is more flexible than the hierarchical model. It allows each record to have multiple parent and child records, forming an arbitrary graph.
- **Relational model**
  The relational model is based on a perception of the world which consists of a

collection of basic objects (entities) and relationships among these objects. Each entity has associated with it a set of attributes describing it.

- **Object-oriented model**
  The object-oriented model is based on a collection of objects, like the E-R model. An object contains values stored in instance variables within the object. An object also contains methods that operate on the object. UML is a kind of object-oriented modeling language.

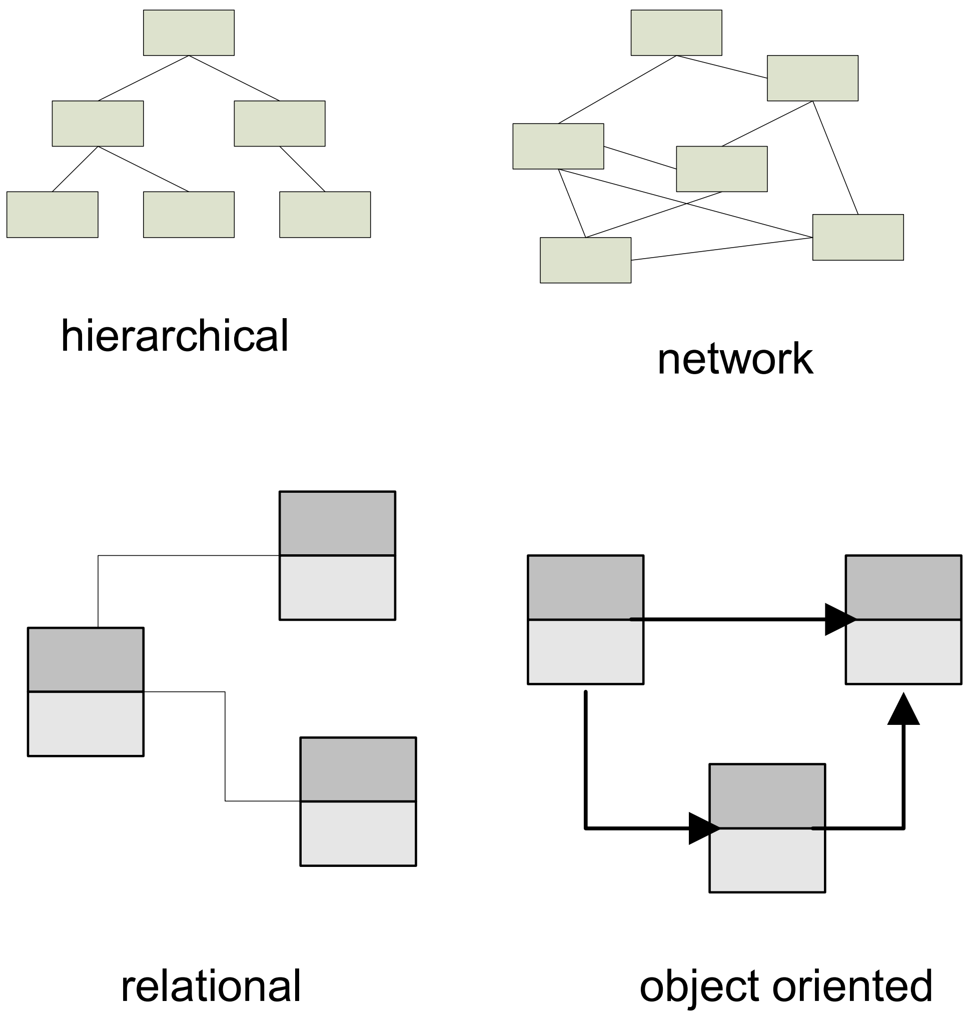
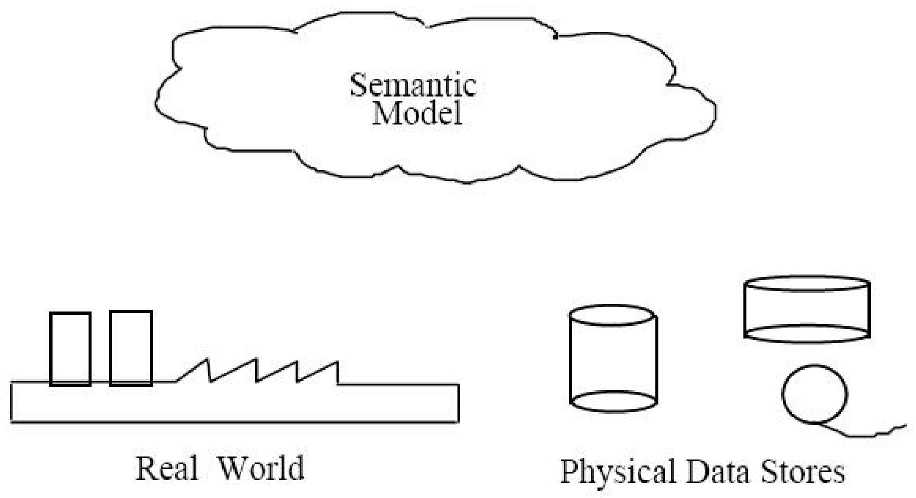


Figure 3.2: Different structures of database models

3.1.2 Semantic Data Model

A semantic data model is a technique to define the meaning of data within the context of its interrelationships with other data. The logical data structure of a database management system (DBMS), whether hierarchical, network, or relational, cannot totally satisfy the requirements for a conceptual definition of data because it is limited in scope and biased toward the implementation strategy employed by the DBMS. Therefore, the need to define data from a conceptual view has led to the development of semantic data modeling techniques. That is, techniques to define the meaning of data within the context of its interrelationships with other data. As illustrated in the figure 3.3, the real world, in terms of resources, ideas, events, etc., are symbolically defined within physical data stores. A semantic data model is an abstraction which defines how the stored symbols relate to the real world. Thus, the model must be a true representation of the real world. ([NIST, 1993])



Source: FIPS Publication 184 [NIST, 1993]

Figure 3.3: Semantic data models

3.2 Geometry Modeling

Computer-aided technologies are a broad term describing the use of computer technology to aid in the design, analysis, and manufacture of products. General CAD system architectures and strategies for internal model representation are described, for instance by Spur/Krause in [Spur and Frank-Lothar, 1984].

Today the extensive development of computer aided engineering becomes the big advantage for implementing CE. By using computer aided engineering all facets of the product development process can access a common design and can innovate and improve the design from the different disciplines by using a common computer aided engineering database ([Bettaieb et al., 2004]). At first there must be a convenient platform on which different divisions for a new product development can work and communicate simultaneously in the CE.

In a PLC there are a serial of phases. For each phase we use one or more tools to implement it, such as CAD, CAE, CAM ([Wiebe, 1998]). For example, CAD was used in the phase "Design" in the figure 2.1.

- Computer Aided Design (CAD): a set of tools for automating certain of the steps in the design process

- Computer Aided Engineering (CAE): the concept of automating all steps involved in creating a product and enabling all engineering disciplines to share the data created by others
- Computer Aided Manufacture (CAM): the application of computers to the manufacture process

As shown in the figure 3.4, CAD is a part of CAE and used in the design phase. Different geometry models are established in this phase. It receives Requirements from CAE and process the design in three steps. A concept design will be started to define a raw concept using the input requirements from CAE. Using the created raw concept in the first phase, a detailed design is carried out. As a result, a fast maturely designed model is sent to analysis and finally it is submitted to CAM.

Mostly we get the geometrical information about a product from CAD. The geometrical design of the product plays an important role in the whole PLC, because it will be strictly adopted in the phased after the design ([Summers et al., 2001]). So it should be well-formed defined and described.

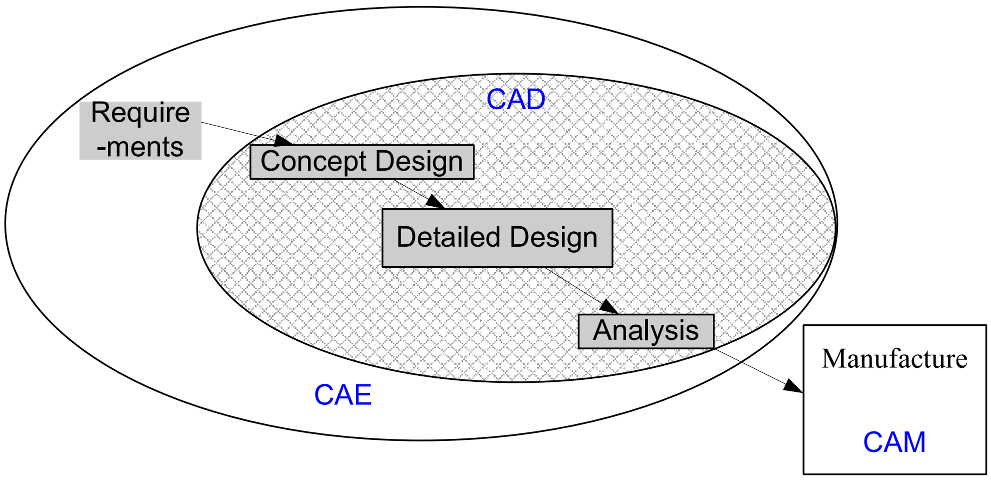


Figure 3.4: Interaction of the various computer aided technologies

3.2.1 Feature-based Modeling

In today's common understanding, features are the main modeling objects of CAD systems. Regarding to their semantics and shape content, features in modern CAD systems can be considered as being complex design objects, characterized by a set of attributes or properties that determine the resulting behavior of the feature within the CAD model and, specifically, the shape model. The semantic content varies with the scope of domain and application. Features for design, assembly, manufacturing, or quality assurance contain different application specific information. ([Spur and Frank-Lothar, 1984])

3.2.2 Parameter-based Modeling

CAD systems provide parametric design both in shape and feature modeling. The user may assign values to dimensional and feature variables. From the user point of view, geometric constraints, e.g., shape elements being parallel, intersecting, co-planar, rectangular have to be distinguished. Also constraints on parameters owned by features, part or assembly objects need their own description. Additionally, CAD systems provide means for defining shape or feature independent variables. ([Spur and Frank-Lothar, 1984])

3.2.3 Shape Model Entity Identification - Persistent Naming Problem

Out of a large number of research findings somewhat related to data exchange, the so-called persistent naming problem may be named as an input to later conceptual discussion.

As a means for internal data management, CAD systems need to be able to identify those elements of their shape model that for further modeling operations have been selected, i.e. picked, by the user. As an example, an edge of a body is selected by the user as an input to a blend or chamfer operation. Even if the original entity has lost validity due to later model alterations, the originally selected entity must remain identifiable for the system. For that purpose, the application of persistent identifiers has been proposed. The strategy for assigning persistent identifiers to shape entities is not trivial. In literature, it is referred to as the persistent naming problem [Chen and Hoffmann, 1995].

For system internal application, some heuristic solutions to persistently naming of model entities have been proposed for shape modeling (see for example [Capoyleas et al., 1996], [Chen and Hoffmann, 1995], [Raghothama and Shapiro, 1998], [Wu et al., 2001]). Kripak describes an application for history-based parametric solid modeling systems in [Kripac, 1997].

All these proposals do not provide a closed solution to the persistent naming problem. The authors limit their investigations to the scope of system internal model management. An adaptation to the area of product model exchange is not known.

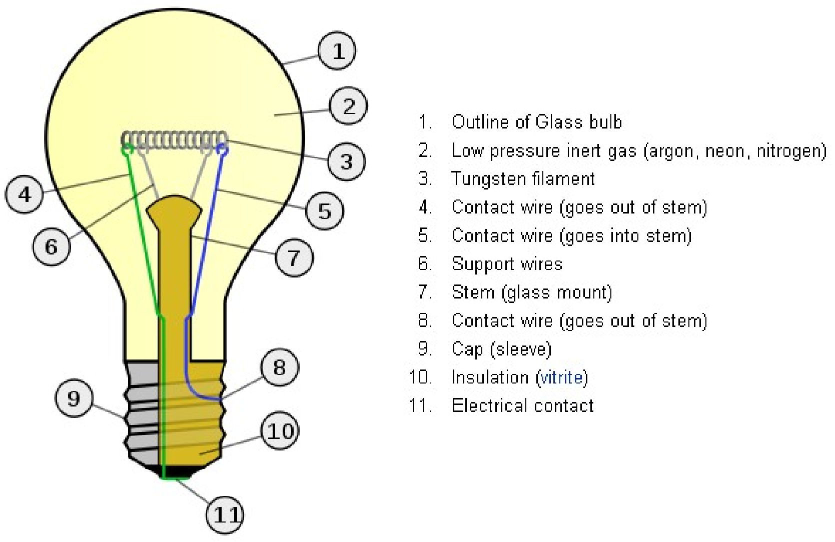
3.3 Light Bulb Model

The model of a light bulb, illustrated in Figure 3.5, is investigated as an exemplary virtual product designed using VE. There are two reasons to choose the light bulb as the descriptive object throughout the thesis:

- The light bulb is a common product and the structure of the light bulb is simple. It is easy to describe it.

- Although the light bulb is simple, it comprises many aspects such as geometric, thermal, light aspects and so on. It is obvious that different aspects of design are more advantageous for different representations.

This section will describe the light bulb models designed in different aspects by different modeling methods. The following chapters will use these models to illustrate the data exchange and integration of these models.



Adapted from: Incandescent Lamps [2]

Figure 3.5: Structure of a light bulb

3.3.1 Knowledge Model of a Light Bulb

Every person stores the knowledge of the world in his mind. Each designer has his own knowledge and understanding when he wants to design a product. In the knowledge model of a light bulb some knowledge existing in the mind which is relevant to the light bulb should be extracted. This information is described as descriptive features and constraints ([Roucoules and Tichkiewitch, 2000]).

Descriptive features: Descriptive features are basic objects which are already stored in our minds and easy to describe. They are the first image when we design the product model. Descriptive features are described according to different characteristics (expressed in rectangles) and behaviors (expressed in ellipses). As shown in Figure 3.6, descriptive features are ball, gas, and metal wire. The characteristic of the ball is the radius. One behavior of the ball is to calculate its volume according to its radius. The characteristics of the gas are the density and the sort. One behavior of the gas is to calculate its weight. The characteristics of the metal wire are the length, the cross-sectional radius, and the density. The behaviors of the metal wire are to calculate its weight and anchorage force according to the its characteristics. The ball is the abstract object of the outline of the light bulb. The gas is the abstract object of the gas in the light bulb. The metal wire is the abstract object of the filament and support wire.

Constraints: Constraints for different characteristics of descriptive features are described here. These constrains are just simple mathematic comparisons. For example, in Figure 3.6 the constrains are shown as equality ([Roucoules and Tichkiewitch, 2000]) and greater. Variable 1 and 2 could represent any two values of the characteristics of descriptive features.

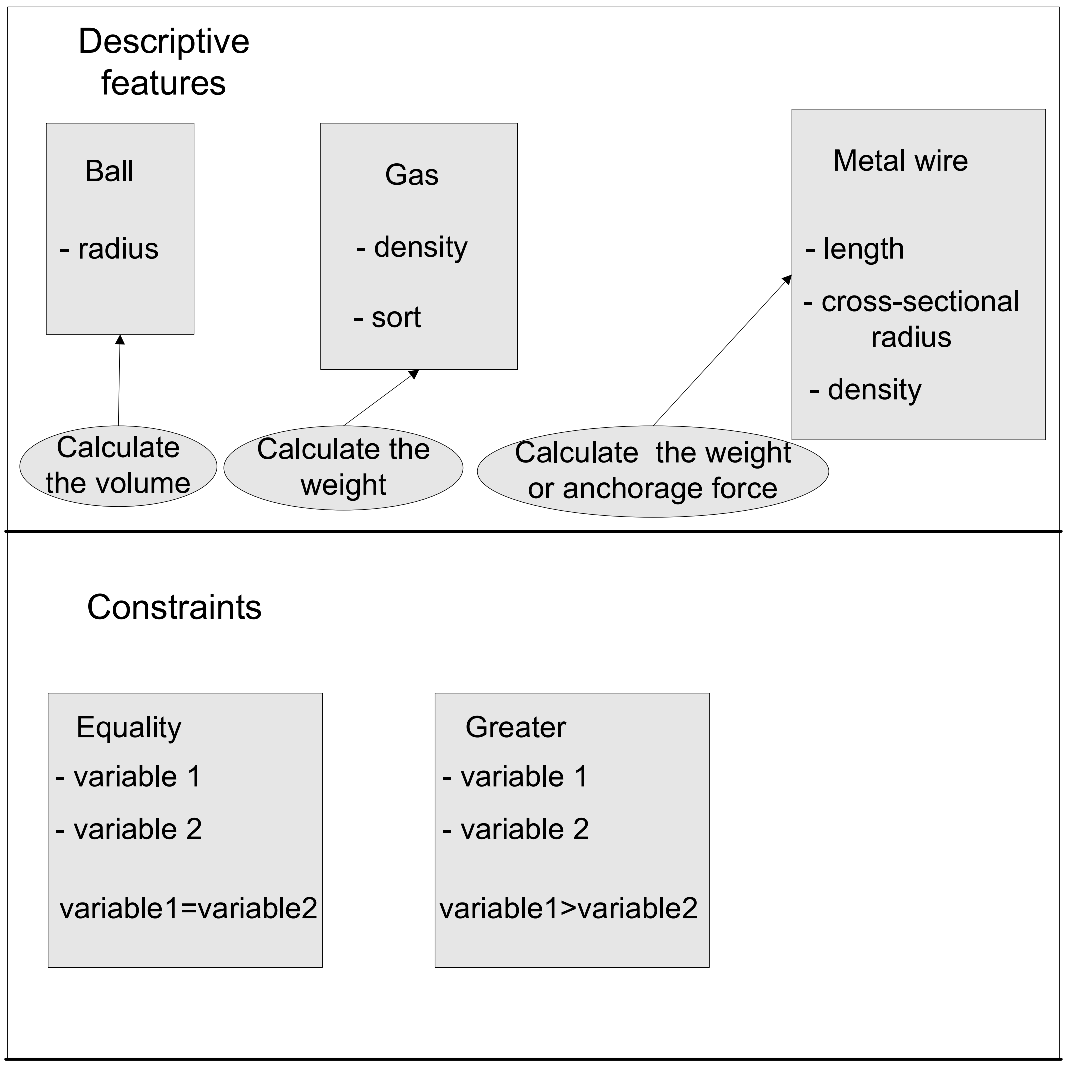


Figure 3.6: Knowledge model of a light bulb

3.3.2 Structural Model of a Light Bulb

The structural model of the light bulb is shown in Figure 3.7. In fact this model only represents part of the structure of the light bulb. We represent six components of the light bulb which are a part of the components of light bulb mentioned in Figure 3.5. The six components are outline, gas, filament, support wire, contact wire, and electrical contact which are all expressed in the form of rectangles. The ellipses express the links. If two components have a physical contact, then we draw two ellipses connected to these two components. These two ellipses are named using the names of these components. For example, in Figure 3.7 the filament has a contact with the support wire. The link *Filament/Support wire* is connected to filament and the link *Support ware/Filament* is connected to support wire. The Outline is filled with gas, so the outline and the gas have also a contact. The links to express this contact are named *Outline/Gas* and *Gas/Outline*. A relationship constraint restricts the relationship between components.

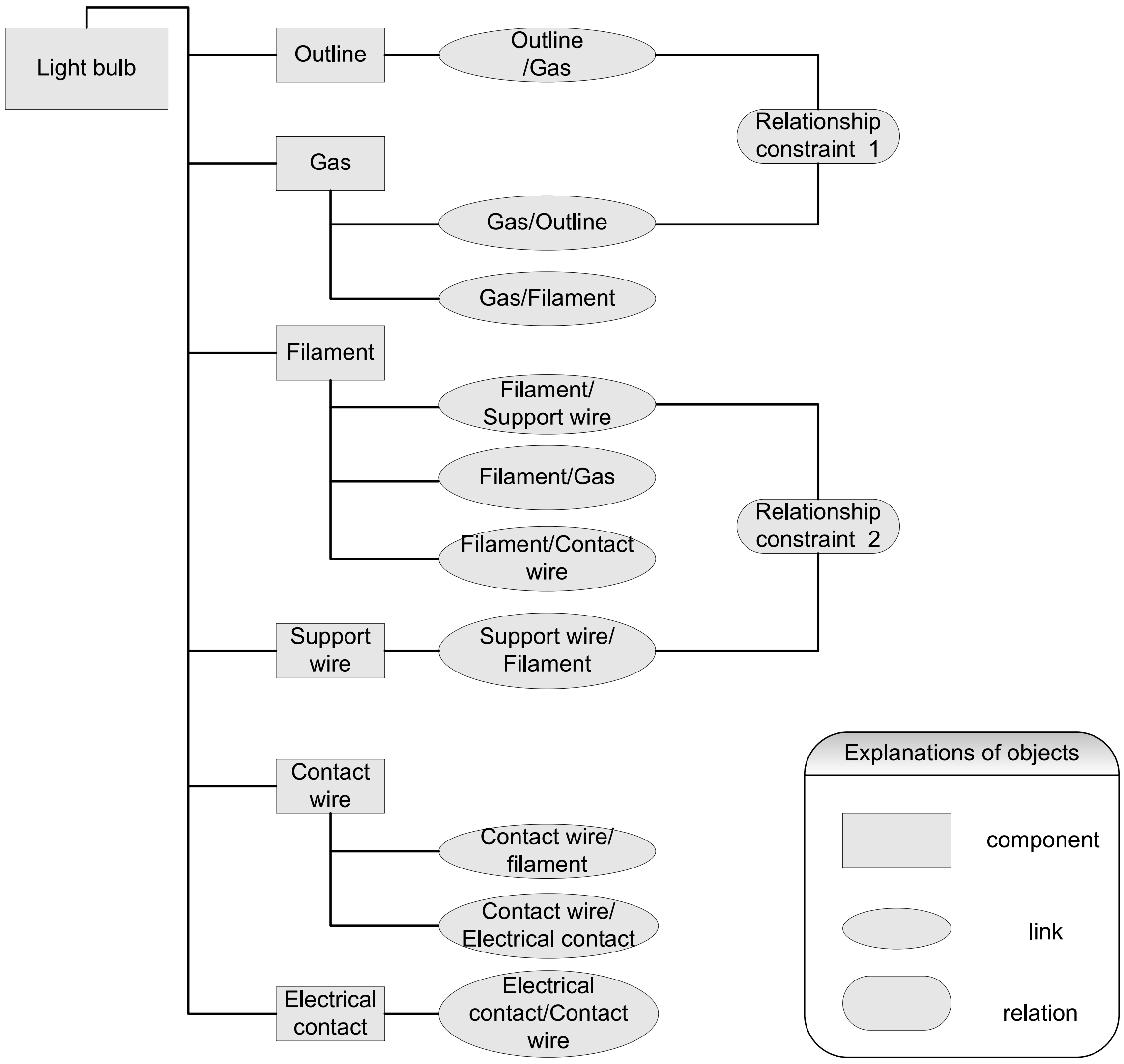


Figure 3.7: Structural model of a light bulb

As shown in Figure 3.8, the relationship constraint between filament and support wire is Anchorage force > Weight. Calculating the anchorage force and the weight is a behavior of the descriptive feature *metal wire* in the knowledge model. Filament and support wire are also metal wire. So, they have this behavior. After the calculation, we get the anchorage force of support wire and the weight of filament. Then according to the constraint *greater* in the knowledge model, we get the relationship constraint between filament and support wire. When we change some characteristics of filament which has an effect on calculation on weight of filament, then the relevant characteristics of support wire should also be changed.

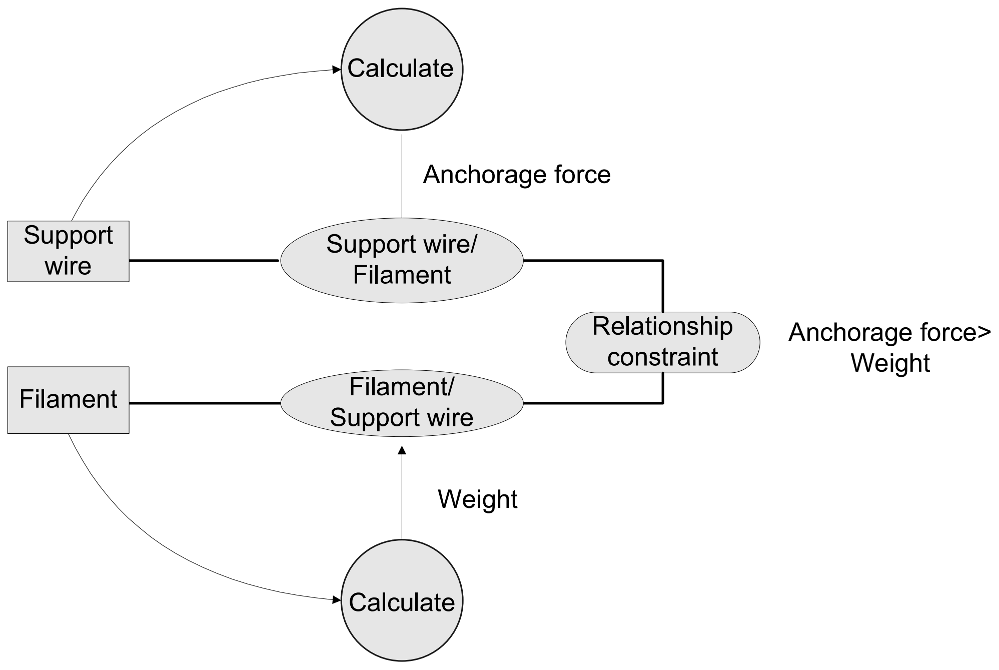


Figure 3.8: Relationship between filament and support Wire

3.3.3 Light Bulb Models in Different Applications

In this section we describe further kinds of models of light bulb, including the geometry, the material, the thermal, and the light model. These models represent different aspects of the light bulb.

In order to construct the model of a light bulb more convenient and easier to understand we use only four parts of the light bulb:

- Outline,
- Filament,
- Contact wire, and
- Electrical contact.

Geometry Model

Figure 3.9 shows the geometry model of a light bulb. It is another representation of the light bulb. The structure consists of three levels. In each level we have an element which is distinguished in a dash box. Each element has its attributes. These attributes are the geometric parameters of the elements. According to these attributes the geometric property of the light bulb is fully described. We omit several not important attributes. The element light bulb is named as bulb_geometry which represents the highest geometric level of the light bulb. The element *bulb_geometry* consists of four elements: the outline (named as *outline_geom*), the filament (named as *filament_geom*), the contact wire (named as *contact_geom*), and the electrical contact (named as *electrical_geom*). *glass_geom* (representation of geometry of the outline of glass) is generalized from *outline_geom*. *tungsten_geom* (representation of the tungsten wire's geometry) is generalized from *filament_geom*. Table 3.1 shows the descriptions for the elements and the attributes of the geometry model.

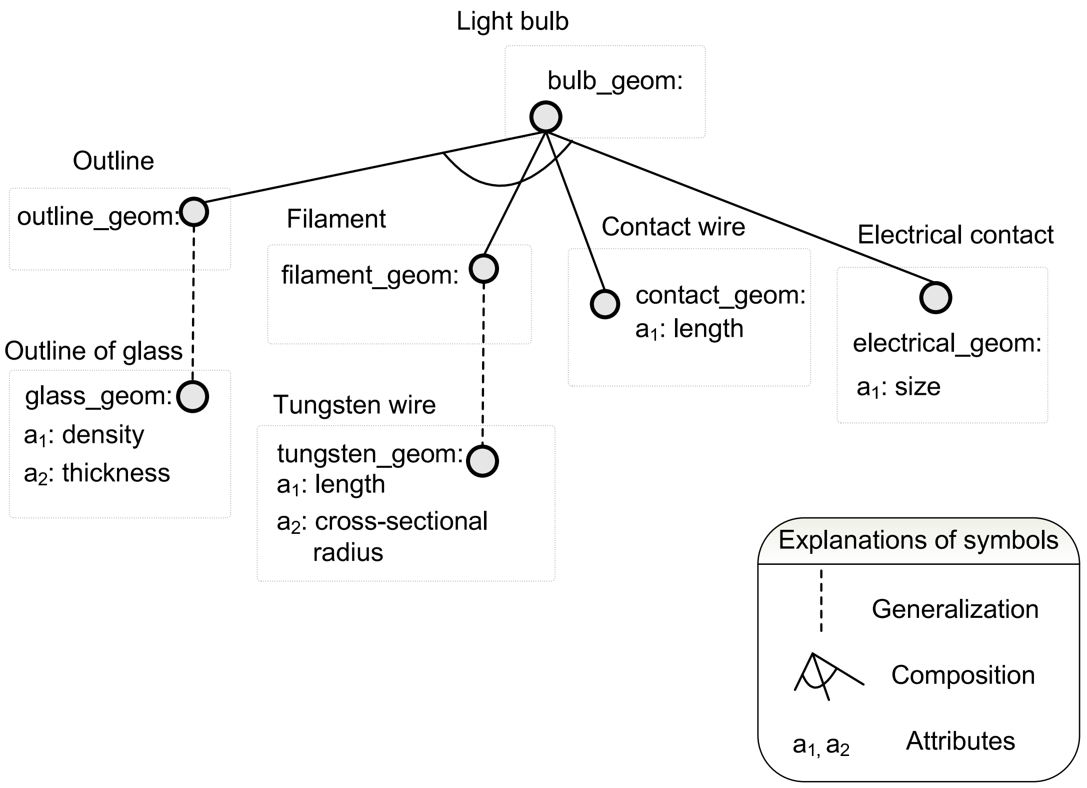


Figure 3.9: Geometry model of a light bulb

Elements names	Attributes
bulb_geom	omitted
outline_geom	omitted
filament_geom	omitted
contact_geom	a_1:length
electrical_geom	a_1:size
glass_geom	a_1:density, a_2:thickness
tungsten_geom	a_1:length, a_2:cross-sectional radius

Table 3.1: Descriptions of elements in geometry Model

Material Model

Figure 3.10 illustrates the material model of a light bulb. The material information of each component of the light bulb is included in this material model. *bulb_mater* describes the general material property. It needs four material property to describe it, including *outline_mater* (representation of the outline material), *filament_mater* (representation of the filament material), *contact_mater* (representation of the contact wire material), and *electrical_mater* (representation of the electrical contact material). *glass_mater* is generalized from *outline_mater*. *tungsten_mater* is generalized from *filament_mater*. Table 3.1 shows the descriptions for the elements and the attributes of the material model. These attributes are the material parameters.

The attributes of the elements in this model describe the material property, e.g., resistivity [2].

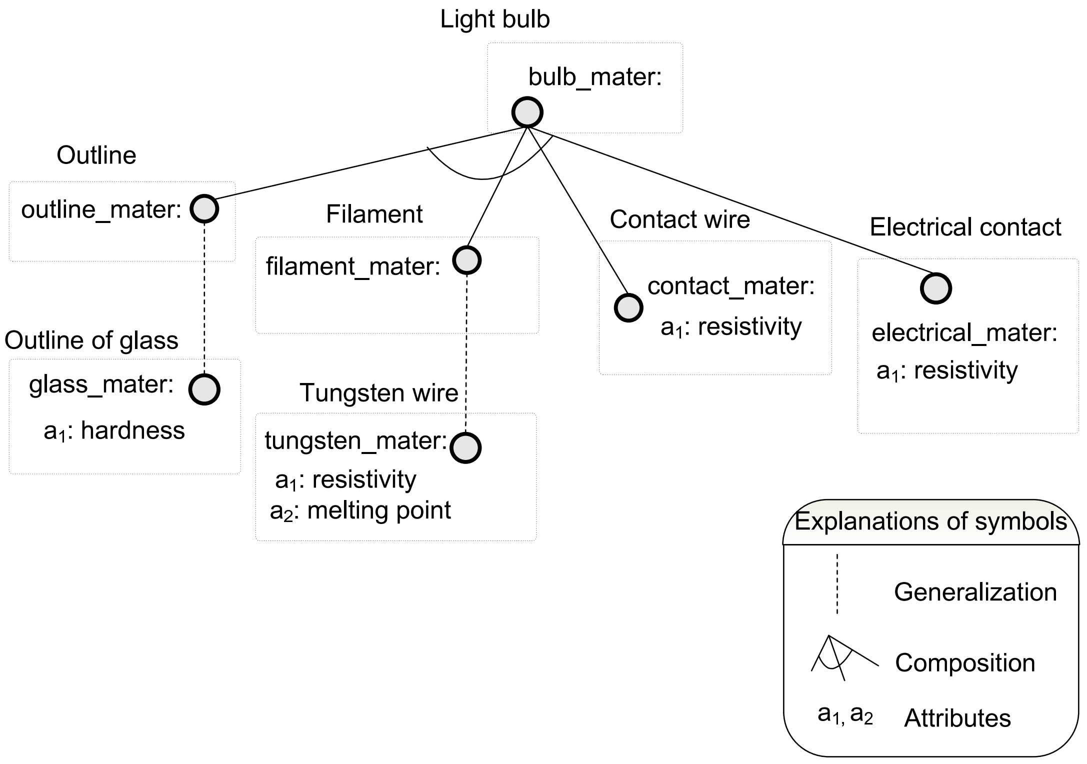


Figure 3.10: Material model of a light bulb

Thermal Model

Figure 3.11 shows the thermal model of a light bulb. *bulb_therm* represents the thermal property of the light bulb. It mainly points to *filament_therm* (representation of the thermal property of filament). *tungsten_therm* is generalized from *filament_therm*. The important parameter of the thermal model is the value of thermal energy of the tungsten wire. The thermal model allows the computation of the thermal energy consumption

[2]Resistivity is a measure of how strongly a material opposes the flow of electric current. A low resistivity indicates a material that readily allows the movement of electrical charge. http://en.wikipedia.org/wiki/Resistivity

Elements names	Attributes
bulb_mater	omitted
outline_mater	omitted
filament_filament	omitted
contact_mater	a_1:resistivity
electrical_mater	a_1:resistivity
glass_mater	a_1:hardness
tungsten_mater	a_1:resistivity, a_2:melting point

Table 3.2: Descriptions of elements in material model

of a light bulb. After knowing this the light energy will be calculated according to the thermal energy. Knowing the light energy helps engineers design different types of light bulbs. The properties of the tungsten wire are computed using following formulas. The formula components are described in Table 3.3.

$$Q = \frac{U^2}{R}t \tag{3.1}$$

$$R = \rho\frac{L}{S} \tag{3.2}$$

$$S = \pi \times r^2 \tag{3.3}$$

From formulas (3.1), (3.2), and (3.3) the following formula is derived:

$$Q = \frac{\pi U^2 r^2}{\rho L}t \tag{3.4}$$

We can see that the parameters ρ, L, and r appearing in Formula (3.5) are the attributes of *tunsten_mater* and *tungsten_geom* in the geometry and material model. In order to calculate Q, we must get these parameters from the geometry and material model. It means that there are dependencies between the thermal model and the geometry model, and dependencies between the thermal model and the material model. These dependencies are marked in Figure 3.11 as dashed arrow from *tungsten_therm* to *tungsten_mater* and from *tungsten_therm* to *tungsten_geom*.

Light Model

Figure 3.12 shows the light model of a light bulb. *bulb_light* represents the light property of the light bulb. It includes *filament_light* (representation of the light property of the filament). *tungsten_light* (representation of the light property of the tungsten) is generalized from *filament_light*. The attribute of *tungsten_light* in this model is the light energy. The light energy of each light bulb determines its type. Engineers design different types of light bulbs according to the values of the light energy. The calculation formula of the light energy is given in the following equation with W the light energy:

Letters	Descriptions
Q	thermal energy
U	voltage (here it is 220 v)
t	time of energization
R	resistance
ρ	resistivity (here it is the resistivity of tungsten)
L	length of the tungsten wire
S	cross-sectional area of the tungsten wire
r	radius

Table 3.3: Descriptions of the letters in formulas

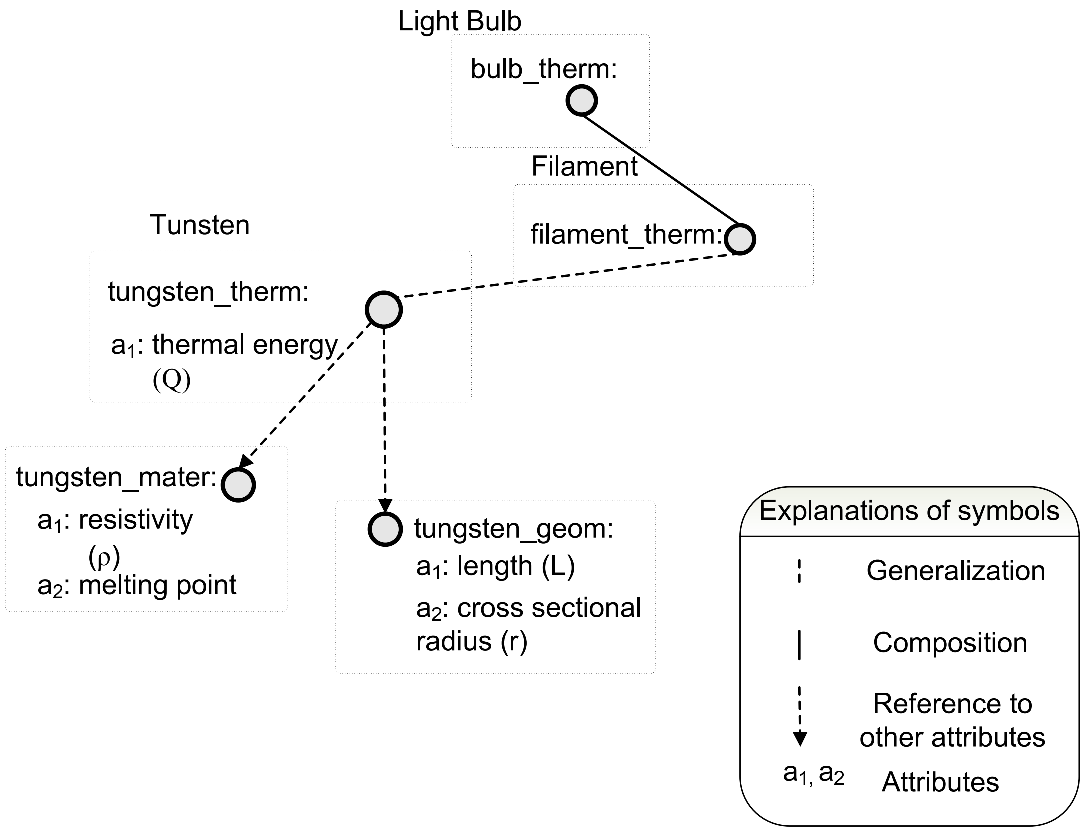


Figure 3.11: Thermal model of a light bulb

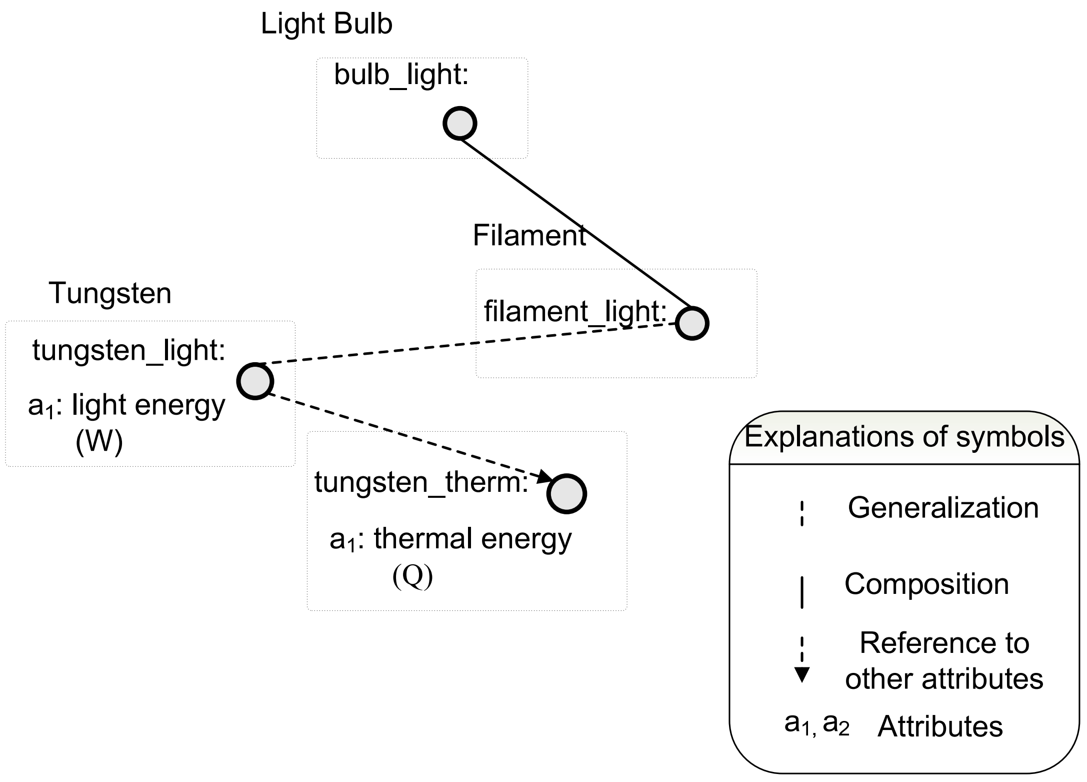


Figure 3.12: Light model of a light bulb

$$W = Q \tag{3.5}$$

The parameter Q in formula (3.5) is the attribute of *tungsten_therm* . In order to calculate W we must get the value of Q from the thermal model. It means that there is a dependency between the light model and the thermal model. This dependency is marked in Figure 3.12 as dashed arrow from the element of *tungsten_light* to *tungsten_therm*.

3.4 Conclusion

From the last three sections we know that there are dependencies among the geometry, the material, the thermal, and the light model. These models share some attributes, the value of which should be exchanged if it is changed in one model. How to find good solutions to perform the data exchange among different product models, or even different systems is an important issue in Concurrent Virtual Engineering. This will be discussed in the next chapter.

Chapter 4

Product Data Exchange

4.1 Motivation

In the phase of conceptual and detailed design, sometimes we need to construct different data models based on the virtual design. Now computer systems such as CAD and CAE have been successfully introduced into production to increase the productivity. However, the models of these computer systems are heterogeneous because they have been developed independently ([Aziz et al., 2004]). If engineers want to use these models, they should get all the relevant information of the models, analyze them and finally adapt them to their expected formats required by their tools.

Data exchange is unavoidable between different organizations, different projects, even between different modules of a large project. Date exchange exists in the following situations.

- A large project for designing a product is divided into a few small modules which are easy to finish, so that the work for designing each module can be assigned to different companies. First of all, the interfaces between the modules must be good defined for the integration later. Each module communicates with others following the well defined interfaces. By this means, the data of one module designed by one company could be used by others for their own design. But it has a big probability that different companies have different design systems relied on their own tools to design their models. So, formats transformation of different models is necessary.

- In a company this communication is also often encountered. The designers working on different data models may exchange their data information in order to get more optimized models. It is possible that these designers use the same system to construct models or different.

- A company has an existing system which it has outgrown and wishes to buy a better one. It is desirable that information should be transferred from one system to another without redesign which is time consuming and error prone.

4.2 Strategies for Data Exchange

Data exchange is defined as the process of transferring relevant/common information between different engineering parties in order to meet the projects objectives and to minimize data re-entry and duplication ([Alshawi and Ingirige, 2003]). Data transfer requires exchangeable data mediums, bus-systems, direct connections, networks, or remote transmission. It depends on at least one source and target system. Functions to transfer data into the target component or application are required. According to the format used by the systems to transfer the data, different kinds of data exchange are classified as following:

- **System specific data exchange** (see Figure 4.1(a)) includes the data transfer between two applications.
  - Direct data transfer from one system to the other and the reverse process: this data exchange means that the data exchange is directly implemented, no matter in which formats the models are described. The programs for one point-to-point transfer are written specific to these both. ([Yang et al., 2004])
  - Some systems have specific functions or subcomponents to export data into external formats. This situation is limited to some systems and not generally supported. Especially, it is recommended for transfer of data designed using a common system architecture.
- **System neutral data exchange** (see Figure 4.1(b)) translates all models into one common format, which is named as ”neutral format”. Some standards, like IGES and STEP ([Yeh and You, 2000]), have been developed to specify such a mechanism. Solutions based on the use of direct translation or neutral formats each requires a sequence of operations to be carried out to affect the exchange of data from one computer system to another.

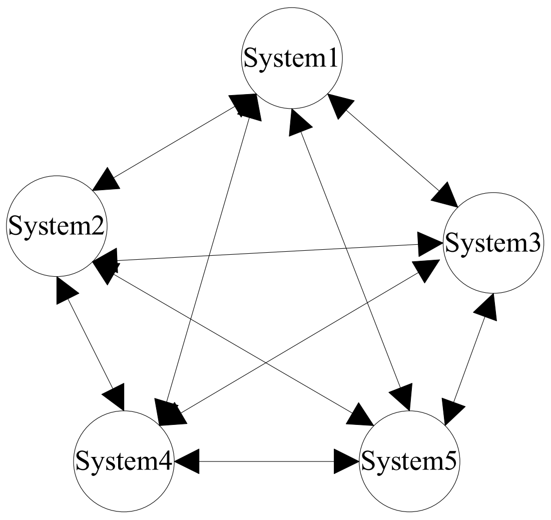
The mechanism ”direct data transfer” can achieve very high quality in results, but it suffers the following disadvantages:

- the systems for this mechanism are expensive to acquire and maintain, because the total system must be updated each time when any CAD subsystem is upgraded;
- the number of such translators increases exponentially with the number of systems involved: for n systems, the number of direct translators required is n $\times$ (n-1).

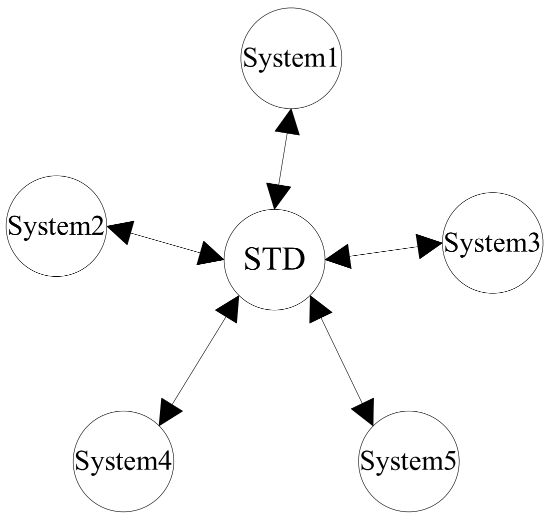
Table 4.1 gives a summary of the advantages and weaknesses of ”system specific data exchange” and ”system neutral data exchange”.

4.3 Standards of Data Exchange

In this section, some standards of data exchange, like IGES, STEP and PDML, are introduced. Besides, their advantages as well as disadvantages are listed, based on the comparison among them.



(a) System Specific



(b) System Neutral

Figure 4.1: Kinds of Exchange

	Direct Translation	**Neutral Formats**
Purpose and use for translation	Software designed for specific translation are needed	Combine two half-links from potentially different suppliers to achieve translation
Update and stability	Expensive to maintain: have to be updated every time when one system changes	Published formats are stable
Cost of translators or half-links	Require n×(n-1) translators to communicate between n systems	Require 2 × n half-links to communicate between n systems

Table 4.1: Comparison of ”direct translation” with ”neutral formats”

4.3.1 IGES

IGES (Initial Graphics Exchange Specification) defines a neutral data format that allows the digital exchange of information among CAD systems. Using IGES, a CAD user can exchange product data models in the form of circuit diagrams, wireframe, freeform surface, or solid modeling representations ([McDonald, 1997]). Applications supported by IGES include traditional engineering drawings, models for analysis, and other manufacturing functions.

Limitations of IGES

IGES aims to implement the data exchange accurately, easily, and efficiently. In practice, it does not achieve any of these aims completely.

- In practice few systems can transfer the drawings accurately. In order to avoid errors, a detailed knowledge of the sending system should be known. But if there are many systems sending information to other systems at the same time, it could be not easy to finish this work.

- IGES is applied to transfer two-dimensional models and not suitable to the three-dimensional solid ones. Besides, it is also not suitable to transfer non-geometry information.

- IGES files need much space to store. And the pre- and post-processing time is costly. If we consider the storage requirement and processing time, the use of IGES will be inefficient.

Considering these weaknesses, IGES had been further developed, but since the first version of STEP appeared in 1994, the development of IGES declined. The last version of IGES was developed in 1996. Now it has been totally replaced by STEP.

4.3.2 STEP

STEP (STandard for the Exchange of Product model data) is a standard which has been developed by the ISO (International Organization for Standardization) for the product data exchange, storage, and access. STEP has a long history. The first STEP project appeared in 1984 ([Fowler, 1995]). It has been developed based on the previous standards which are used in the data exchange among CAD/CAM systems such as IGES. STEP is an extension of the existing standards and it is fully defined. Compared to IEGS, STEP makes a separation of data definition from implementation. Therefore, the data models defined in STEP are independent to their different implementations as used in different ways ([Fowler, 1995]).

There are two important elements contained in STEP:

- **EXPRESS**
  EXPRESS is a special language used to redefine models derived from the CAX models.

- **Application Protocols**
  There are many different kinds of application protocols in STEP. They are defined to supply more constraints and requirements of the product data to the model designed by EXPRESS ([Zhang et al., 2000]).

Figure 4.2 shows the process using STEP for the data exchange between two CAD systems. Let us explain this by using the example: the data transfer from CAD system A to CAD system B. The whole process is divided into three steps which are indicated in the figure with 1, 2 and 3. The reverse process (data transfer from CAD system B to A) has the same principle. The three steps for that are also marked with the numbers 1, 2 and 3, but in another color. In the following, the process for data transfer from the system A to B is described.

- **Step 1**
  A ”pre-processor” is used to get the model designed by the CAD system A. The

result will be sent to "Inside STEP", which works as the processor of heterogeneous data models. This step is relative easy and the programs for the pre-processor are also easy to implement.

- **Step 2**
  "Inside STEP" gets the data model from the pre-processor. And then with the help of EXPRESS and AP (application protocols), a new STEP file including as much as possible data information of the CAD system A is generated. In this way the previous model in the format of CAD system A is translated into STEP.

- **Step 3**
  In this step, a "post-processor" is used to translate the model in the format of STEP into the format of system B. Now the model of system A is successfully translated into a model of system B by STEP. In this way the data exchange between system A and system B is processed. The program for this step is usually difficult to write, because it demands to be familiar with STEP and of course its own CAD system standard.

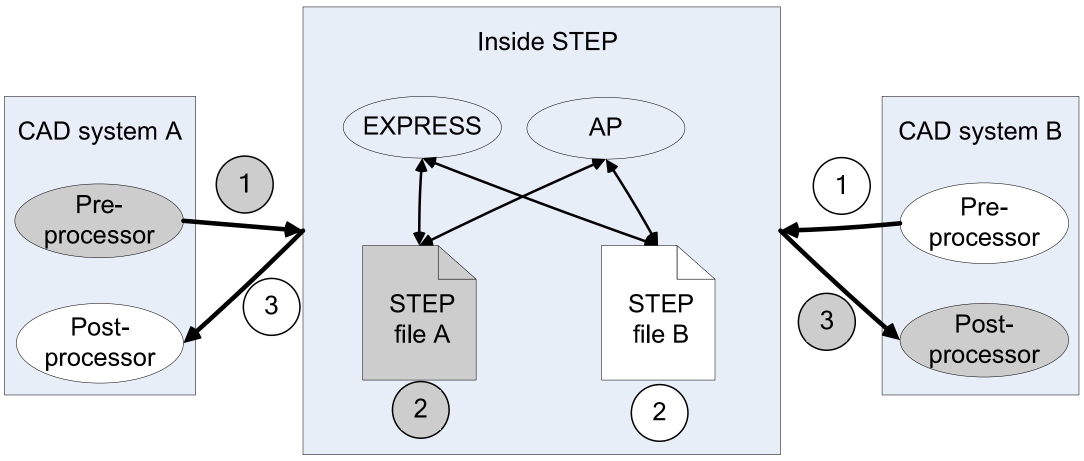


Figure 4.2: Data exchange using STEP

Advantages of STEP

The appearance of STEP has some advantages and also overcomes some problems which exist in the earlier data exchange standards (e.g., IGES):

- In STEP, each product data has its own semantics and it is easy to understand for each engineer, while an IGES file representing a CAD drawing is computer readable. This means that only the special engineers can understand the entities contained in the CAD drawings.

- STEP not only enables the creation of data models that extend beyond the description of product geometry, but also provides tools and methodologies to describe data for any conceivable product or service in any conceivable industry ([AL-Timimi and MacKrell, 1996]).

- Besides the exchange of the geometry data, STEP also allows the exchange of other types of data throughout a PLC. Any two systems can exchange data between each other, as long as they use the same STEP-based data model.

Limitations of STEP

Although STEP has gained much more success than any other earlier data exchange standards, it is still not perfect. Some limitations of STEP are concluded in the following.

- The target system can not control the received data from the source system. Maximally, it can control part of the data. If the data model in the target system has a higher level of detail than the source system, some information will be lost or not be available to the target system ([AL-Timimi and MacKrell, 1996]).

- There are experts on geometry, materials, mechanic, etc. But only few people can understand the enabling technologies of integration ([Owen, 1997]).

- STEP is always difficult to keep up with demand for integration between different company systems ([Noël et al., 2003]). In practice, developers often need to define the product data models themselves or make expansions based on the integrated sources.

4.3.3 Other Standards

Product Data Markup Language [3] (PDML) is an eXtensible Markup Language (XML) vocabulary designed to support the interchange of product information among commercial systems, such as PDM systems, or government systems. PDML is being developed as part of the Product Data Interoperability (PDI) project under the sponsorship of the Joint Electronic Commerce Program Office (JECPO), and supported by several other federal government agencies and commercial entities in the USA. Three major PDM vendors are active participants in the PDI program and are developing prototype implementations of PDML. The initial focus of PDML development is legacy product data systems that support the operation of the Defence Logistics Agency (DLA).

To reduce the complexity of this paper, interested readers are referred to the "PDML White Paper", which is available online at *http://xml.coverpages.org/pdml.html*.

4.4 Illustration of the Data Exchange Process

In this section we will first illustrate the data exchange process among different data models using the automaton formalism theory. Then we take the light bulb models as the examples to describe the data exchange process in detail.

[3] http://xml.coverpages.org/pdml.html

4.4.1 Illustration of the Data Exchange Process using the Automaton Formalism Theory

In this section the process of data exchange will be illustrated using the automaton formalism theory which refers to the content in [Vergeest and Horváth, 1999].

Basic principle of the Automaton Formalism Theory

The meanings of the symbols used in this theory are shown in Table 4.2. Here the

Symbols	Automaton formalism	Model design aspect
M	Set of states	Design models
Q	Set of initial states mapping to M	A part of the data in design models
Q^*	Set of new states mapping to M	A part of the data in design models
m	One state in M	One model in a time
q	One state in Q	A part of the data in design models in a time
Σ	Set of possible input symbols	Operations
Δ	Set of possible output symbols	Output data
a	One of the possible input symbols	Operations
b	One of the possible Output symbols	Operations
δ	The function to determine the state transfer	The behavior to change the data
λ	The function to determine the output symbols	The behavior to determine the transferred data

Table 4.2: Formal definitions for the model design

example in Figure 4.3 will be taken to explain the basic principle of the theory.

As shown in Figure 4.3, giving an new value $a_1^x \in \Sigma^x$ to Q^x, the initial state of the data $q_0^x \in Q^x$ will be changed into a new state $q_1^x \in Q^x$ according to the function δ in the formal (4.1).

$$\delta : Q \times \Sigma \rightarrow Q \tag{4.1}$$

The function δ works on the condition that one state is changed into another one in a set. After q_0^x is changed into q_1^x, this change will be transferred to the second set of states (Q^y).

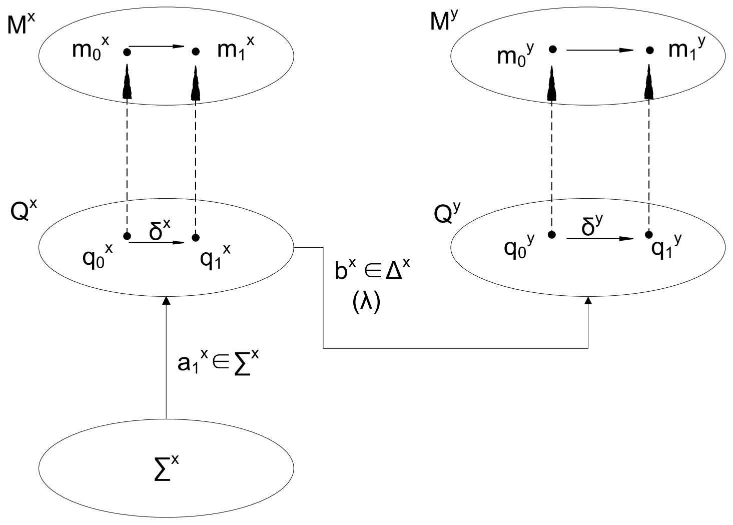
The function λ defined as

$$\lambda : Q^* \times \Sigma \rightarrow \Delta \tag{4.2}$$

decides which symbols should be output into the second set of states. $b^x \in \Delta^x$ is the output symbols from Q^x to Q^y. After receiving b^x, the initial state q_0^y is changed to a new state q_1^y. This can also be expressed as:

$$\delta(q_0^y, b^x) = q_1^y \tag{4.3}$$

Each state in the set Q is mapped to one model in the set M. So if a data state in Q is changed, the mapped model in M will be also changed.



With reference to: [Vergeest and Horváth, 1999]

Figure 4.3: Changing the Data of one Model Has an Effect on the other Model

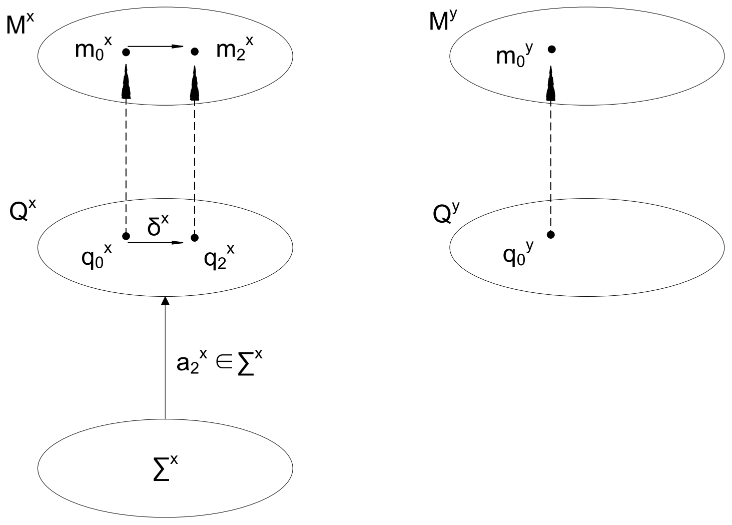
Types of Data Transfer

The data transfer between different models can be classified into three categories:

- **Changing the data of one model has an effect on the other model**
  This case is shown in Figure 4.3. After the new data value q_0^x is input, the data state is changed from q_0^x into q_1^x. As a result, the model M^x is changed. Afer the data state in the set Q^y is changed from q_0^y into q_1^y, the model M^y is also changed.

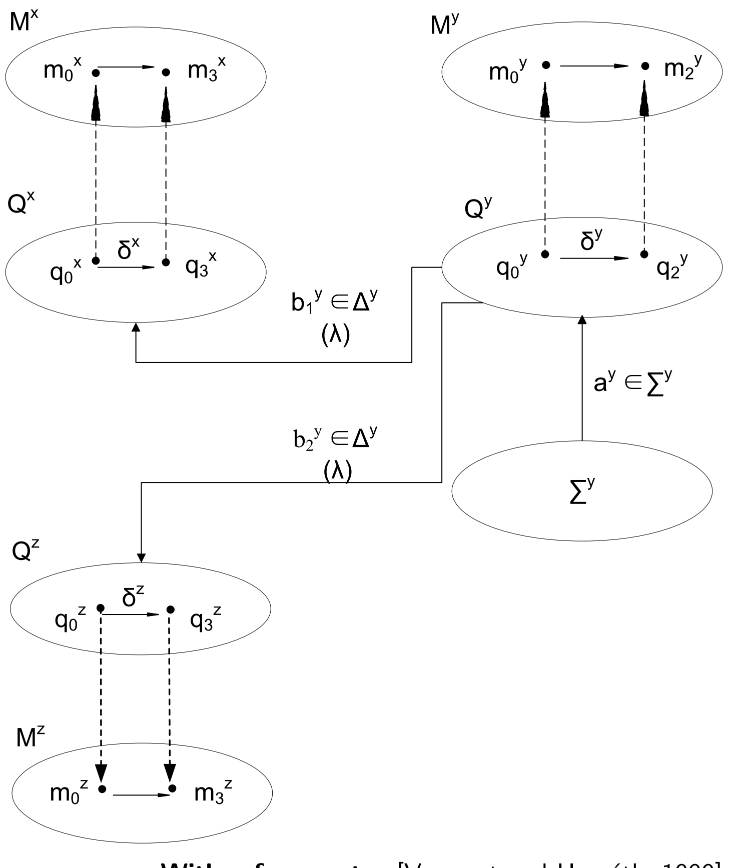
- **Changing the data of one model has no effect on the others**
  As shown in Figure 4.4, giving a new data value $a_2^x \in \Sigma^x$, one data state $q_0^x \in Q^x$ will be changed into a new state $q_2^x \in Q^x$ according to the function $\delta : Q \times \Sigma \rightarrow Q$. The model M^x is changed. But the change does not have an effect on any data state in Q^y. So it is not necessary to transfer this change to Q^y. $q_0^y \in Q^x$ which maps to $m_0^y \in M^y$ will not change. As a result, the data model M^y will not be changed.

- **Changing the data of one model has an effect on more than one other model**
  As shown in Figure 4.5, giving a new data value $a^y \in \Sigma^y$, a new data state $\delta(q_0^y, q_2^y = a^y)$ will be generated. The model M^y is changed. The change of the data state q_0^y affects on the data states in the set Q^x and Q^z. Two output symbols will be transferred from the set Q^y to Q^x and form Q^y to Q^z. b_1^y is calculated according to the function λ_1, while b_2^y is calculated according to the function λ_2. As a result, the models M^x and M^z are also changed.



With reference to: [Vergeest and Horváth, 1999]

Figure 4.4: Changing the Data of one Model Has no Effect on the Others



With reference to: [Vergeest and Horváth, 1999]

Figure 4.5: Changing the Data of one Model Has an Effect on More than one other Model

4.4.2 Illustration of the Data Exchange Process using the Light Bulb Models

Data Exchange among Models of Different applications

The model in Figure 4.3 is an abstract of the model in Figure 4.6. In Figure 4.6 three models of light bulb in different applications are defined. They are the geometry model, the thermal model and the light model. Here we handle two questions: how to exchange the length value between the geometry model and the thermal model, and how to exchange thermal energy between the thermal and the light model. Each model is stored in a table in the database. We can do many operations on the geometry model. If we change the length of the tungsten wire, it will affect the thermal energy of the tungsten whose information is stored in the table "tungsten_therm". So the change of information must be informed to the thermal model as soon as possible. Then the table of the tungsten for the thermal model makes the proper adjustment. At the same time, the thermal energy is changed, which reflects the light energy in the light model. So this information should be transferred. Finally, the three models should also be correspondingly adjusted. These three different models are designed by different systems, it is possible that the data formats in different systems are different. In this case, we need a standard to unify the formats transformation in the process of the data exchange.

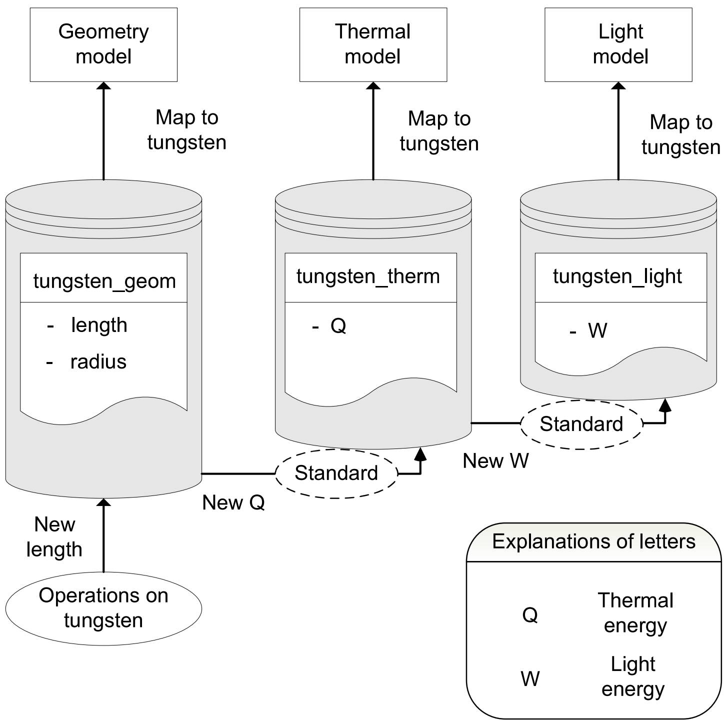


Figure 4.6: Data transfer from the geometry model to the thermal model and from the thermal model to the light model

Figure 4.3 and Figure 4.6 give only one situation, in which all the operations on the length of the first table have effect on the length of the second table. But sometimes we change the information of the first table. There is no need to inform this change to the second table, because it does not change anything for the second battery table. This case is shown in Figure 4.7.

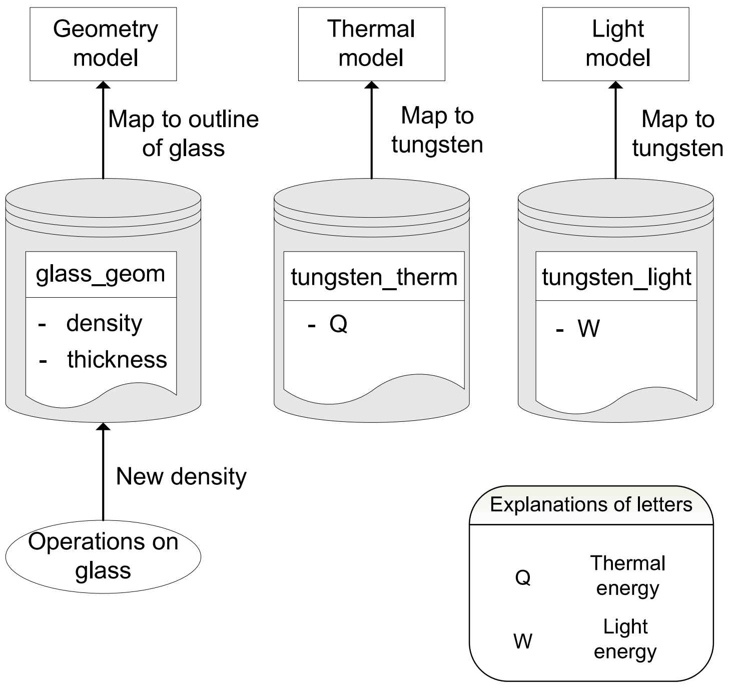


Figure 4.7: No need to transfer data between the geometry, thermal and light Models

In Figure 4.7, we only make the change operation on density. It does not make any sense to the thermal and the light model. So in this situation the exchange among these three models is not necessary. This case has been described in Figure 4.4 in an abstract form.

In Figure 4.8, four models (the geometry, the material, the thermal and the light model) are used in the process of data exchange. Giving a new value of the light energy to the light model, the original value of the light energy will be changed into a new one. Then according to the formula (3.5), the light model transfers the new value of the thermal energy to the thermal model. After the thermal model receives the new value of the thermal energy, it changes the original value into the new value of the thermal energy. After that, according to the formulas (3.1), (3.2), and (3.3), the changes of the value of the thermal energy will lead to the changes of the values of the length, the cross-sectional radius, and the resistivity. After calculation, the new value of length, radius, or resistivity will be transferred to the geometry and the material model. Changing the resistivity means that the material must be replaced by others. In practical application this case happens rarely.

Name	Value	Related to thermal model	Related to light model
Length	Value 1	Yes	Yes
Radius	Value 2	Yes	Yes
Resistivity	Value 3	Yes	Yes

Table 4.3: Table of Tungsten_geom in the geometry model

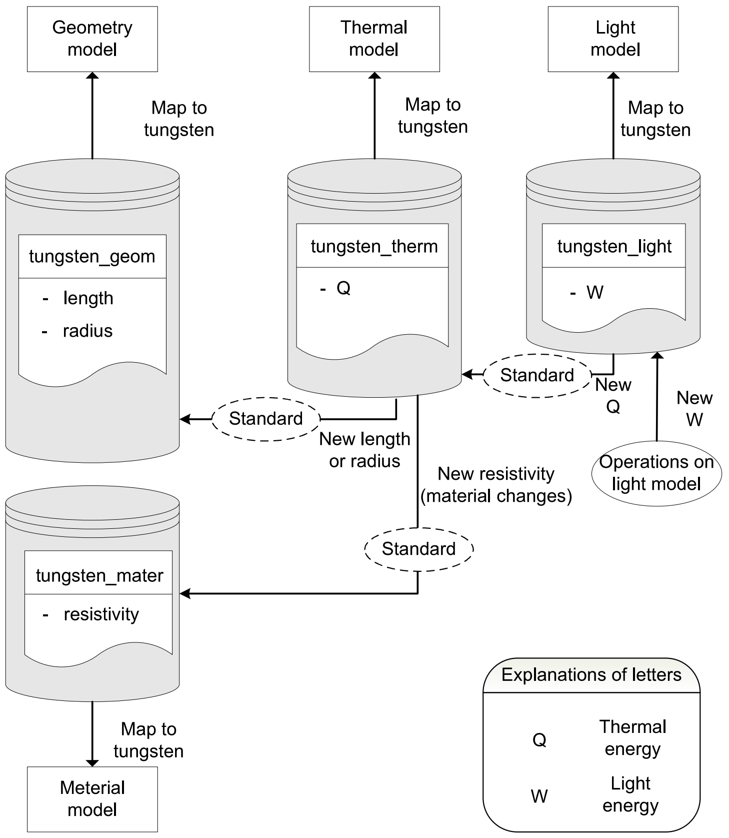


Figure 4.8: Data transfer from the light model to the thermal model and from the thermal model to the geometry model

Data Exchange among models in the same application

Figure 4.9 shows the data exchange between two geometry models of light bulb. These two geometry models are separated constructed by different CAD systems (CAD system 1 and CAD system 2). Each CAD system designs all the geometry part of the light bulb, including the tungsten wire, the outline of the glass, the contact wire and the electrical contact. CAD system 1 has different heterogeneous concepts and database representation from CAD system 2, so the two geometry models are presented in different forms. When CAD system 1 transfers the data of the geometry model to the CAD system 2, the transferred data must be transformed from the format of CAD system 1 into the format of CAD system 2. During the process of the data exchange, we need a exchange standard to implement the format transformation. This exchange standard can be IGES, STEP or other exchange standards.

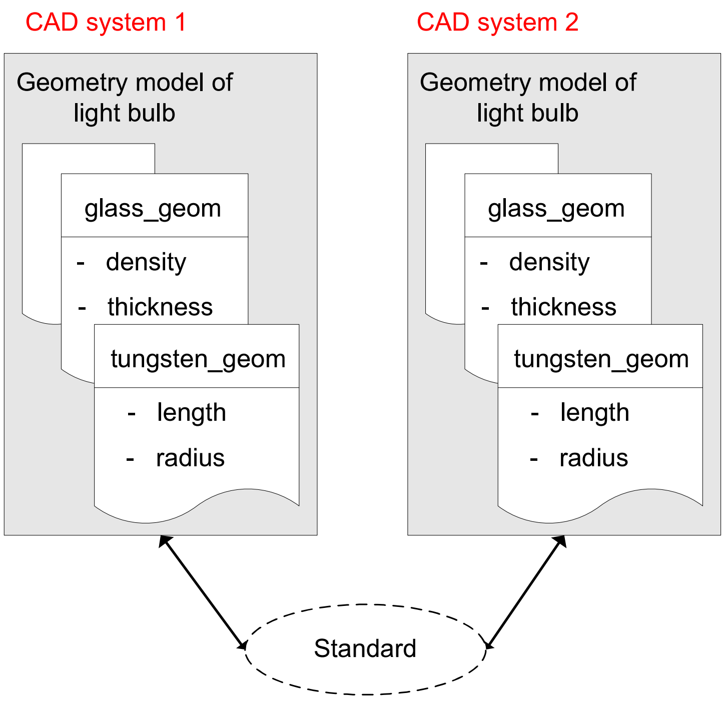


Figure 4.9: Data exchange between the two geometry models in different CAD systems

4.5 Problems during Product Data Exchange

Before explaining the problems which may occur during the process of data exchange, we introduce a theory based on [Eckert et al., 2005]. The symbols used later are explained in the table 4.4.

A function named as *T()* is defined as following:

$$b = T(a) \tag{4.4}$$

with

$$a = a_1, a_2, ..., a_m \in SchemaA \vee \Phi$$

Symbols	**Explanations**
Φ	empty set
$\in$	belongs to
$\neg$	logical not
$\exists$	logical for some (there exists)
$\forall$	logical for all
$\rightarrow$	logical imply

Table 4.4: Symbols for schema of mapping classes

$$b = b_1, b_2, ..., b_m \in SchemaB \vee \Phi$$

The funktion *T()* in the formula (4.4) is used to get how the data in the *data model A* shall be represented in the *data model B*. It has three different results, which are displayed in Figure 4.10.

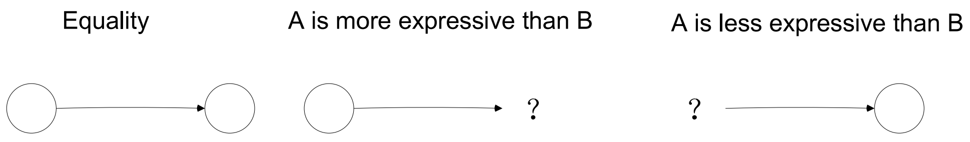


Figure 4.10: The schema of mapping classes

These three results are explained as following:

- **Equality**: When A transfers the data to B, B gets the data which has an equivalent semantics as the data in A. In this case we say that A is equal to B. This can be described in the formula:

$$\forall\ a \in A : \exists\ b \in B : a \rightarrow b \tag{4.5}$$

- **A is more expressive than B**: When A transfers the data to B, at least one data in A has no representation in B. In this case we say that A is more expressive than B. This can be described in the formula:

$$\exists\ a \in A : \neg\exists\ b \in B : a \rightarrow b \tag{4.6}$$

- **A is less expressive than B**: When A transfers the data to B, for the data in B, there is no representation in A. In this case we say that that A is less expressive than B. This can be described in the formula:

$$\exists\ b \in B : \neg\exists\ a \in A : a \rightarrow b \tag{4.7}$$

Equality is an ideal mode of data exchange. In this case, the target model gets all the needed data from the source model and the data is all represented correctly. But in practical application, it is difficult to achieve this goal. We conclude three main problems which may exist in data exchange as following:

- **Lost data**
  This problem corresponds the above theory (A is more expressive than B). Data loss during a data transformation occurs when information is transferred from the source to the target tool, where no equal or suitable structure is available. This problem is especially hard to solve, if the data is transformed between the complex structures (e.g., hierarchical and object-oriented structures).

  For example, Two tools *Statemate* and *Teamwork* are used to explain the data loss during the data exchange. *Statemate* is a software to model the systems. *Teamwork* is a proven, reliable and friendly Web based software solution for managing work and communication in any field. *Teamwork* is easy to use, so that an extended team can contribute; it is also capable of handling complex projects. The data structure of the tools *Statemate* and *Teamwork* are different. In *Statemate*, the data is organized in hierarchical structures. In *Teamwork*, the data are structured in a relational schema. This difference can result in the data loss during some mappings from one structure to another. The result of data exchange between the tools *Statemate* and *Teamwork* are listed in the table 4.5. From the comparison we can clear recognize, that data exchange between heterogeneous models can not be processed without data to lose.

From/to	**Teamwork**	**Statemate**
Teamwork	equality	A<B
Statemate	A>B	equality

Table 4.5: Data exchange results

- **Incorrect interpretation**
  This problem corresponds the above theory (A is less expressive than B). Because of the perception of the real world, different models have different semantics, e.g., cognitive and naming heterogeneity. Cognitive and naming heterogeneity as well as similar names often result in misinterpretations. This may result in, that the data representation in the target model is more expressive than in the source model. A more understandable problem is a different granularity of decimal places or different semantic units.

- **Integration of new/changed domains** ([Vornholt and Geist, 2008])
  This problem occurs when many models of different domains have data exchange between each other. When new or updated applications are integrated to design a product, a common environment must be adapted to satisfy the requirements. The adaption affects existing structures as well as transfer protocols.

Chapter 5

Product Data Integration

5.1 Motivation

In the design phase of concurrent virtual engineering, we use different computer aided systems to design product models of different domains. For example, different application systems of CAD systems (e.g., Pro/E, IDEAS and AutoCAD) and PDM systems (e.g., Windchill and Metaphase) are used as computer aided systems to design product models. Product models of different domains may be the geometry, the material, and the electric model, or models of other domains. Integrating these different computer aided systems or product models of different domains is required in concurrent virtual engineering. According to [Schönhoff et al., 1997], these requirements are classified into three aspects:

- Integration of different process systems: for example, Windchill is good at modeling the work process and management but not good at project analysis. Project2010 developed by Microsoft compensates these limitations. So solving the integration of Windchill and Project2010 is very important.

- Integration of different application models in conceptual design: for example, designing a light bulb the engineers must consider and analyze geometry, light, thermal models at the same time. So it is necessary to find a solution to solve the data sharing problem between different application models.

- Data exchange between systems in detailed design: for example, the data exchange between CAD systems and CAE systems. In the present time STEP is the major method. But the application protocols of STEP can not fully satisfy the requirements of the data integration in practical application. So the engineers need to define and extend the product data models themselves.

5.2 Architectures

This section presents the basic architectures of the data integration solutions. These architectures build the basic principle of processing a data integration. They are described as following:

- **Using Mapping between Databases**
In Figure 5.1, there are three systems, each of which stores his data in a database. For example, the data of system A is saved in the database A. In order to integrate the data of the system A with the system B, a third system C is brought into use. With a mapping between database A and database C, the data in the database A is successfully integrated into the system C. In the same meaning, the data of the system B is integrated into the system C. Now, the data integration of the system A with the system B is successfully completed. Advantages of this architecture:

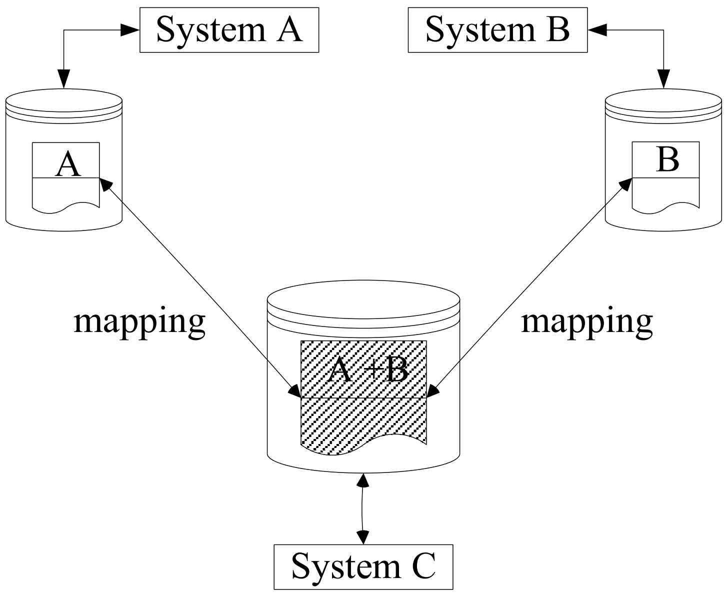


Figure 5.1: Mapping between databases

It is easy to implement and costs relative small. For example, ODBC and JDBC have been the proven technologies.

Disadvantages of this architecture: The systems must store the data to be exchanged and must provide the corresponding access method. It is suitable for small and relative simple models. If the amount of the model data is large, it is very difficult to use this architecture to integrate data models, because it is hard in this case to build the mappings between the model data. For each of the two systems, different interfaces must be developed. To integration of n systems, there must be n $\times$(n-1) interfaces. The reusability of the interfaces is poor.

Figure 5.2 shows an example of this architecture. The geometry model is stored in a XML file, while the thermal model is saved in two objects of a table. By mapping the content of the XML file to a table named filament, the information of the geometry model is integrated into a third database. The relationship between the geometry model and the thermal model is built though the value of ”resistance”.

- **Using Common Objects** (see Figure 5.3):
A common data model is designed for different applications. A translator (half-link) for the systems translates the formats into a common data model and back to the local format ([Lessa et al., 2000]). When new data is needed or stored, the

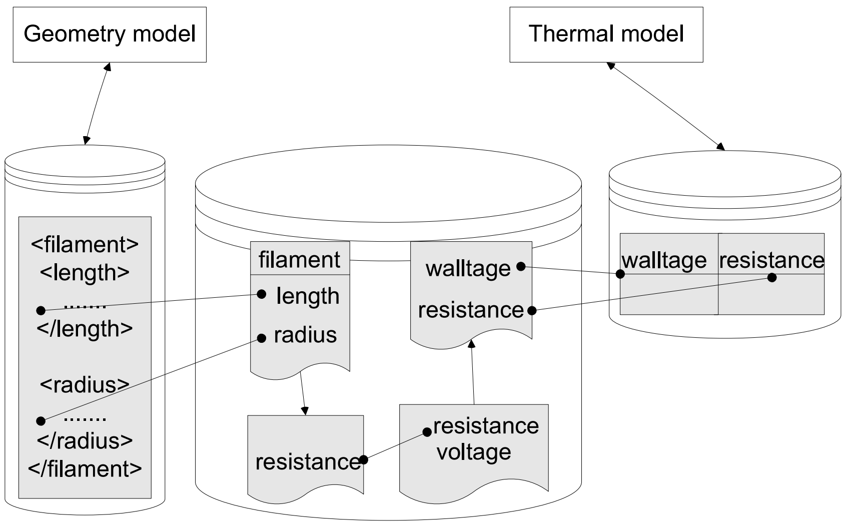


Figure 5.2: Data integration of geometry and thermal model

common database is used. Similar to the exchange formats changes or complete data are translated and integrated.

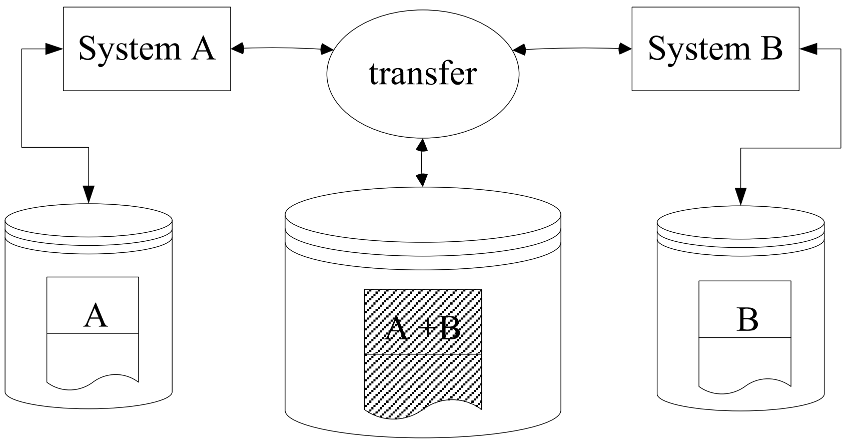


Figure 5.3: Data integration using common objects

- **Using Managed Data** (see Figure 5.4):
  This approach integrates models in one common structure (e.g., data warehouse), though still in different formats. The dependencies are controlled by an external application which manages common parameters and constraints. The management is independent from transfer or common data models. Only dependencies between heterogeneous models are managed.

- **Using Inclusion** (see Figure 5.5):
  One system is used to integrate another system. The data in system A is integrated in some data of system B. For example, a CAD system integrates simulations. The windows application - Microsoft Office Word integrates a text editor and a picture editor.

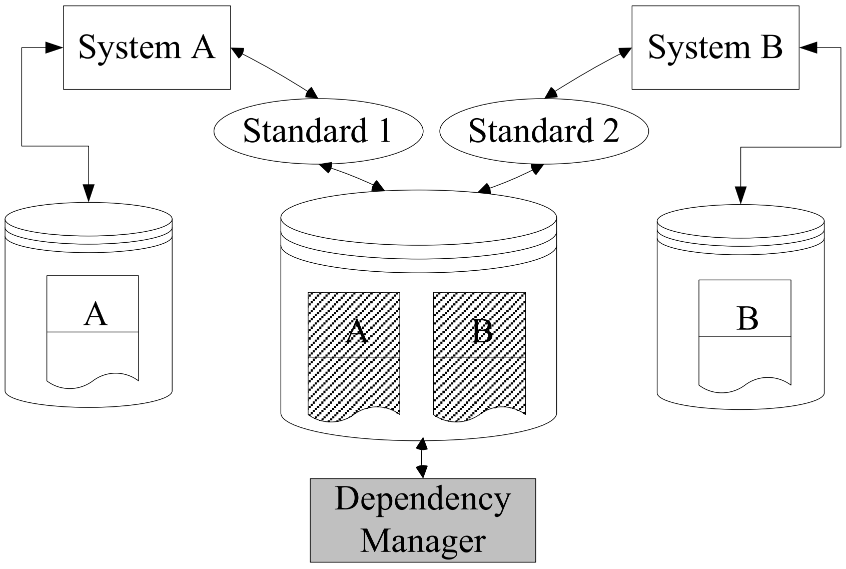


Figure 5.4: Data integration using managed data

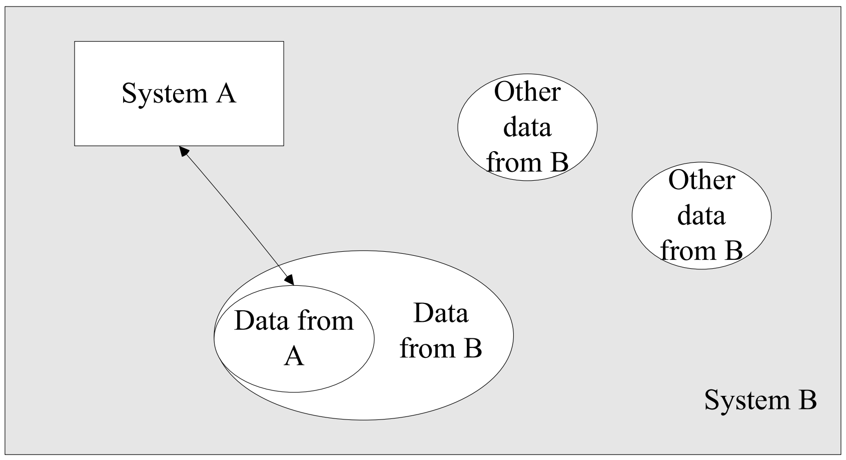


Figure 5.5: Data integration using inclusion

5.3 Approaches

In the previous section, we talked about the architecture of data integration in concurrent engineering. These architectures illustrate the base principle of data integration. In this section, a few concrete approaches are introduced to explain further, how the integration of data in concurrent engineering can be accomplished. This gives an overview of the different ways to address the integration problem. The presented classification of these approaches is based on [Dittrich and Jonscher, 2000].

5.3.1 Manual Integration

Manual integration is the process employed by users to communicate information with different systems by manually integrating the needed data. It is performed by users, when the automatic integration performed by the data system has limitations, such as several softwares have integration limitations and so on. Manual integration has a high demand on users. Users must have detailed knowledge on data location, the representation of the logical data, and data semantics.

5.3.2 Integration based on Common User Interface

In this case, the user is supplied with a common user interface (e.g., a web browser) that provides a uniform look and feel. Data from relevant information systems is still separately presented, so that homogenization and integration of data yet has to be done by the users (for instance, as in search engines). A common user interface is usually used to enable global view on the data and to combine data for new information.

5.3.3 Integration based on Application

Users can access different data sources and get integrated results by using integration applications. This solution is practical for a small number of component systems. However, applications become increasingly fat as the number of system interfaces and data formats to homogenize and integrate grows.

One example for application-based integration is the Enterprise Application Integration (EAI). Gartner defines EAI as "*the unrestricted sharing of data and business processes among any connected applications and data sources in the enterprise.*"

In EAI there are three topologies of integration:

- Point-to-Point: Different applications are directly connected to each other. This type is not widely used. It is only appropriate to few systems.
  - Advantage:It is easy to implement in small scenarios.
  - Disadvantages: If n applications are needed to be integrated, the number of the interfaces of these applications will be $\frac{n\times(n-1)}{2}$. So many interfaces are difficult to manage. The maintenance of this integration is expensive.

- Bus: Bus topology uses a central messaging backbone (bus) for data propagation. Applications would publish data to bus. Through bus the data would flow to the applications, which is agreed to be integrated. These applications get the data from bus and transform the data into a format required for the applications. This topology is appropriate to the systems which implement the data distribution or collection in the form of 1:n (e.g.,Broadcasting) or n:1 (Data warehouse).
  - Advantages: There are few interfaces using this topology. It has good scalability. The failure of one single application can not affect to the other applications.
  - Disadvantages: The functionality of the local structure of the applications is redundant.
- Hub and Spokes Topology: Hub/Spoke topology uses a centralized broker (Hub) and adapters (Spoke) which connect applications to Hub. Spoke connect to application and convert application data format to a format which Hub understands. Hub manages the data from applications and transforms the data into the format the destination application understand. Then adapter transfers the data from source application to destination application.
  - Advantages: There are few interfaces. It is easy to integrate additional systems.
  - Disadvantages: If the hub has error, it will affect on all the applications. Using a single hub makes the scalability limited.

5.3.4 Integration based on Middleware

Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It is defined as a layer of software above the operating system but below the application program. While applications are relieved from implementing common integration functionality, integration efforts are still needed in applications. Additionally, different middleware tools (e.g., Common Object Request Broker Architecture (CORBA)) usually have to be combined to build integrated systems.

The core idea of CORBA is to introduce a set of object request broker service between the client and the server and after that the objects of system A can access the objects of system B just like the objects of system A can access the objects in system A. System B may be remote or local. Using this method to realize data integration needs IDL (Interface Definition Language) to define the interfaces. Through the compiler IDL is mapped to the special program language. The generated objects include stub (The client uses it to send service requests to ORB.), skeleton (The server uses it to monitor the service requests sent by ORB.), the concrete realization of the interfaces and some other aided objects.([Mowbray and Zahavi, 1995])

Advantages: It is easy to realize the integration of heterogeneous systems (different hardware platforms, operating systems, network environments and so on), because the programmers do not need to know the implementation of the network communication.

It also has the object-oriented advantages. It is relatively a proven data integration method. The flexibility and expansibility is good.

Disadvantages: It needs the implementation of ORB and the related products should also be purchased.

5.3.5 Integration based on Agent

Multi-agent systems and ontology are developed from solving the distributed problems and knowledge sharing. In multi-agent systems, each knowledge-based system is taken as an agent. The knowledge sharing between agents is implemented by message transfer. Currently, from the perspective of system integration, KSE (Knowledge Sharing Effort) developed by Stanford University provides a set of complete methods, including KQML (Knowledge Query and Manipulation Language), KIF (Knowledge Interchange Format), three level logical structure for ontology. This method is used to solve the knowledge sharing between different knowledge systems.

Advantages: KQML realizes the communication function (General developing tools of agent provide this function), so the developers do not need to consider the communication problems too much. KQML is a general abstraction for the communication behavior and its versatility is good. Ontology expressed by KIF can be mapped to many concrete knowledge systems through the Ontolingua server provided by Stanford University. It has good maintainability, extensibility and reusability. In addition, Stanford University also provides other tools and services (e.g. the OKBC) to support the system integration based on agent and ontology.

Disadvantages: It is the integration of knowledgebase-oriented systems. Although ontology expressed by KIF can be mapped to IDL and compiled into specific programs, it is not so direct. There are also some developing tools of agent which compiles ontology expressed by UML into programs. But there is no way to use the powerful function provided by Ontolingua.

5.3.6 Integration based on Uniform Data Access

By this approach, a logical integration of data is accomplished at the data access level. Through a uniform data access, the integration of data can be implemented. ([Ziegler and Dittrich, 2004])

One example for integration based on uniform data access is federated database management systems (FDBMSs). In [Sheth and Larson, 1990], a federated database system (FDBS) is defined as "a collection of cooperating but autonomous component database systems", while a FDBMS is defined as "the software that provides controlled and coordinated manipulation of the component DB systems".

Different component DB systems have three characteristics:

- Distribution: The data is distributed in different databases. These different databases may store in a computer system or in different computer systems.
- Heterogeneity: The heterogeneity of different DB systems are due to the heterogeneity of different DBMS systems which are due to the heterogeneity of data

semantics, including the differences of data structures, query languages and other differences.

- Autonomy: A component DB system is always controlled by its own DBMS. This control is independent and separate. Different types of autonomy are provided in [Garcia Molina and Kogan, 1988].

FDBMS systems are introduced to manage the component DB systems by implementing the collaboration work of different component DB systems, which have above characteristics. FDBMS systems provide two kinds of operations, including local and global operations, to allow control sharing of component DB systems. Using a uniform data access solution, FDBMS systems integrate data from component DBMS systems in logical level. FDBMS systems implement data models of component DB systems, support global queries, global transactions, and global access control.

5.3.7 Integration based on Common Data Storage

In this case, common data storage is used to perform the integration of data. That is, the data to be integrated will be transferred to new data storage. A Data Warehouse is an example of Integration of data using common data storage. Data from several operational sources (on-line transaction processing systems, OLTP) are extracted, transformed, and loaded (ETL) into a data warehouse. Then, analysis, such as online analytical processing (OLAP), can be performed on cubes of integrated and aggregated data [Ziegler and Dittrich, 2004].

5.4 PDM Systems

Product Data Management (PDM) systems appeared in the mid-1980s, which were driven by the development and requirements of CAD systems. During using CAD systems to design products, the engineers realize that they need a tool to manage the design documents. Currently, PDM systems provide an integrated platform for the company information management. They help the engineers to manage not only the engineering data but also the product development process throughout the product life cycle. The data managed by PDM systems mainly includes two classes:

- **Documents**
  In PDM systems, documents point to the electronic documents generated by CAD systems.

- **Bill Of Materials** (BOM[4])
  BOM is the product structure data file which can be identified by computer systems. It is the tie of the contact and communication between different systems. Production department uses BOM to decide the production methods of products. Management department uses BOM to establish the production plan and ensure

[4]Bill of materials (BOM) is a list of the raw materials, sub-assemblies, intermediate assemblies, subcomponents, components, parts and the quantities of each needed to manufacture an end item (final product) http://en.wikipedia.org/wiki/Bill_of_materials

the amount of materials and so on. So it is important for PDM systems to ensure the information of BOM correct.

CAD systems are the source of BOM. They include all the information needed by BOM, such as product structure, material, weight, volume and so on. As PDM systems manage electronic documents generated by CAD systems, how to ensure the consistency of the data between CAD systems and PDM systems is a issue deserved to be in research.

5.4.1 Integration of CAD Systems with PDM

There are two types of integration of PDM systems and CAD systems:

- **Interface**
  A intermediate interface is used to realize the information sharing between two systems.

- **Integration**
  The sharing and consistency of information are realized using integration of one system into another.

There are two kinds of CAD systems: 2-D and 3-D. They are different from each other in implement mechanisms, data storage and development environment, so integrating the CAD systems with PDM must be also discussed in two cases. The 2-D CAD systems can realize the integration of information in bidirection (see Figure 5.6). They read and write the structure and components information of the product from the PDM systems. Then the relevant electronic documents will be put on the product structure of PDM systems ([Oh et al., 2001]). PDM systems can also get the information of product structure and physical property, e.g., weight and volume from 3-D CAD systems. This is based on the unidirectional data interface.

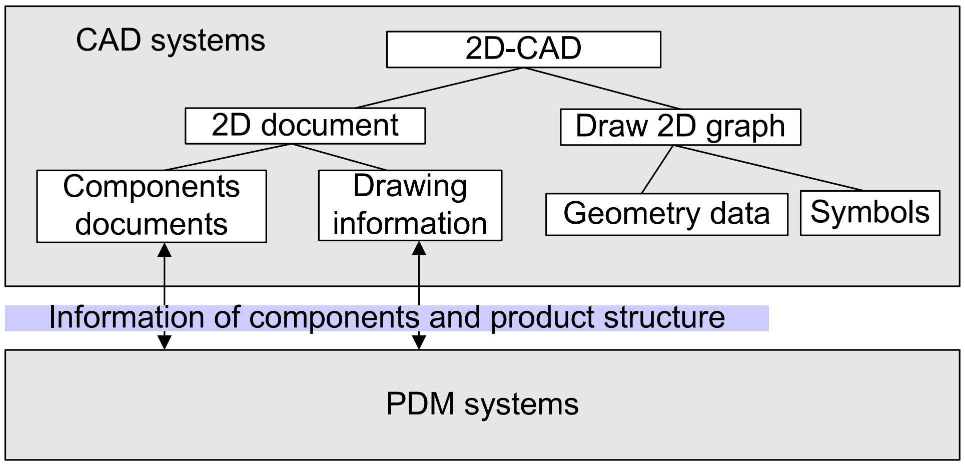


Figure 5.6: Bidirectional integration of 2-D CAD systems with PDM

The information integration of 3-D CAD systems with PDM systems is realized through the unidirectional interface (see Figure 5.7). That is, the structure and components information of the product can directly reflect from CAD systems to PDM systems.

But the reverse process is impossible. Unlike 2-D CAD systems, this unidirectional interface is developed in PDM systems. Its function is to read the structure and components information exported by CAD systems. Most of the 3-D CAD systems have the function to export the graphical product structure and components information in the form of text format files. For example, the BOM files and physical files can be exported. BOM files include the product structure information. Physical files include the basic property information of product components, such as material and so on. These two kinds of information is read and exported directly to PDM system. Because this interface is developed based on PDM systems, the information can be reflected to the inner PDM systems. In this way, the consistency of the information between CAD and PDM systems is realized.

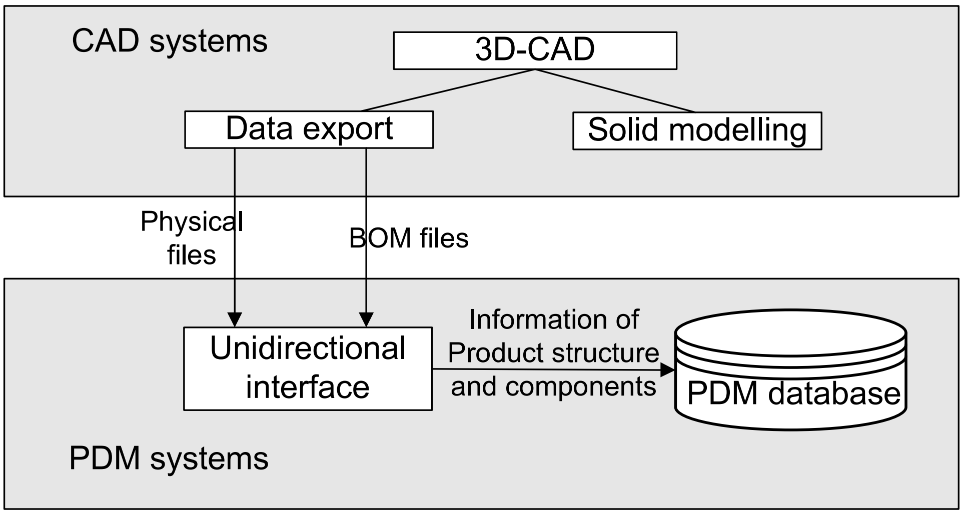


Figure 5.7: Unidirectional interface for integration of 3-D CAD systems with PDM

5.4.2 Windchill

Windchill is used to implement the PDM systems application. It is a program designed by PTC (Parametric Technology Corporation) to manage the product information and to control the process of production in the PLC. It fully utilizes information technology, Internet and related technologies and provides an application for infrastructure of the system software. It provides easy management for complex product data.

The features of Windchill are described in the following:

- **Simple**
  Since Windchill is based on the Web, it is easier to get information from the Web. So the communication between Windchill and the Web is very easy to implement.

- **Powerful**
  Windchill provides a safe way to access data, management of process change with the help of online functions, management of configuration, and automatic control of the whole product process.

- **Connection with other tools**
  Windchill can be integrated with many mechanic and electronic CAD-applications, like MCAD-Tools, AutoCAD.

5.5 Product Data Model as an Integrated Model

Figure 5.8 shows, that the product data model is an integrated model by integrating data from other models. The knowledge model provides the basic knowledge to the structural model and models of different domains. We get the needed data from the knowledge model to design the structural model or others, like the geometry model, the material model etc. In the data transferring process, we can choose different data exchange strategies. The product data model integrates data from the structural model and others in different domains. We can use the common objects, or the managed data architecture to implement the integration. The ”management center” controls the data access to the product data model if we use the common objects architecture. If we use the managed data architecture, both dependencies of the data in the product data model and the data access to the product data model are managed by the ”management center”.

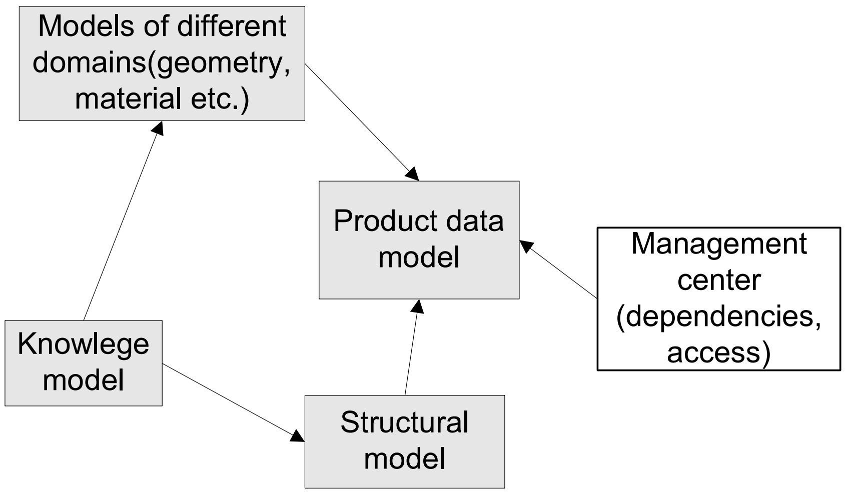

Models of different domains(geometry, material etc.)
Product data model
Management center (dependencies, access)
Knowlege model
Structural model

Figure 5.8: Product data model as an integrated model

5.6 Conclusion

In this Chapter, the basic architectures of data integration were illustrated. These architectures, like data integration using mapping between databases, using common objects or direct integration using managed data, explain the basic principle of data integration. Based on these architectures, several solutions of data integration in Concurrent Virtual Engineering were presented. These solutions will be classified according different categories in the later chapter.

Chapter 6

Classification of Data Exchange and Integration Solutions

In previous chapters, the principle of data exchange and integration are introduced. The base of data exchange and data integration is data management. So, this chapter will talk about the solutions of data management for data exchange and integration. In order to get an overview of data management solutions, they will be classified into different categories. In section 6.1, a fine granular categorization is presented. The general decisions that need to be specified in every system and lead to typical with exchange or integration solutions coupled approaches are described in those subsections. In the final subsection a summarized categorization is presented. In section 6.2, three exemplary collaboration solutions (STEP, PLM, CCI) are classified according to the presented categories of section 6.1. At last, we conclude this chapter with a discussion of the results.

6.1 Categories

This section gives a few categories, according to which the data management solutions are classified and described.

6.1.1 Exchange Control

According to different ways to control the data exchange, there are two typical forms of data management solutions. They will be described in the following:

- **Consecutive transaction management** uses neutral or system specific exchange formats to connect applications by forwarding changes in a sequence of translations. That means, that the changes are forwarded by systems to the affected domains. Therefore, less translation functions are required in exchange control. If this method is used in some affected systems, it means that the dependencies of these affected domains have to be fixed on, so that the data exchange between them does not need a special control. Once the mode of the data exchange is fixed on, the data exchange will be carried on consecutively. Figure 4.6 demonstrates consecutive transaction with the light bulb example. A change of the parameter filament (length) is first transferred into the geometry model, which translates the (complete or incremental) changes into the thermal model. Based on the data of the thermal model, the light model gets the new parameter from a known data format.

- **Central Transaction management** is the opposite of consecutive transaction management. An example is shown in Figure 6.1, where a control center is used to manage the data transaction between different models. There are three models whose data are stored in the database. All operations on the data of the models, like modification, addition, deletion and search of a value, must be carried out under the control of the control center. In this example, the value "length" of the geometry model is changed; the new value will be send to the geometry model through the control center. Then the control center gets all the values (length and radius) from the geometry model back. These values are required later to calculate the thermal energy Q and the light energy W. The new calculated value of Q will be returned to the light model. In this way, data exchange is successfully carried out between the models.

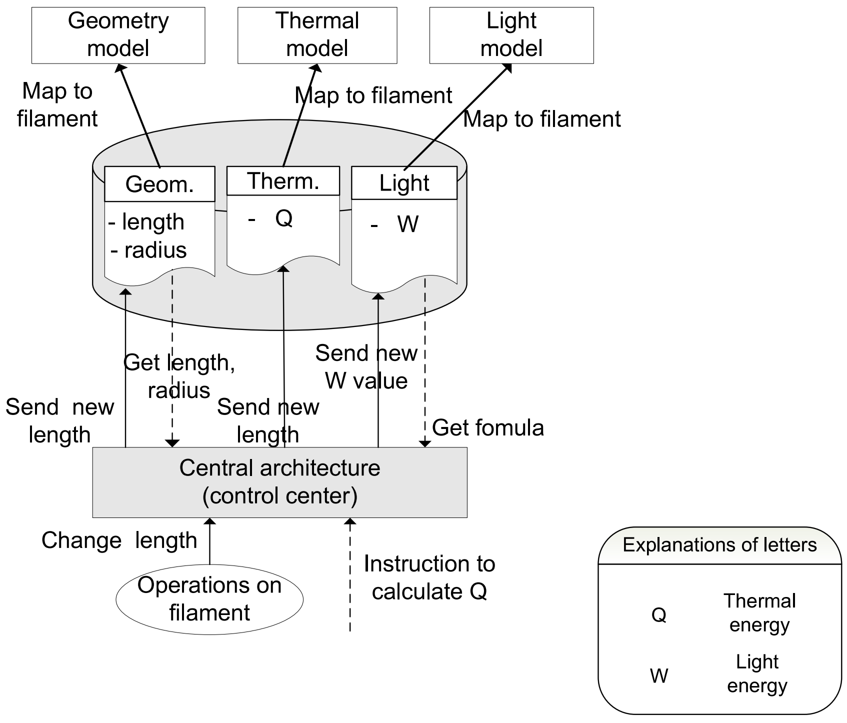


Figure 6.1: Use control center to manage the Data transfer

6.1.2 User Interface

There are two types of user interface used for data management: **specific user interface** and **common user interface**. Each domain expert requires his own representation of the data. So, they define the special views for this purpose. Each application in a domain has its own data representation, independent from the data format used. But a common user interface is usually used to enable global view on the data and to combine data for new information.

Common interfaces (integrated interfaces) for applications can calculate and represent data based on different domains. Interface integration is based on PDM systems, which can recognize CAD files and launch them to the CAD system or make some PDM functions available via CAD menu. Or it is based on a new interface independent from the CAD and PDM system. Common user interfaces enable the extension of an original geometric model by adding an thermal model to get a more complete view on the model.

6.1.3 Data Transfer

According to the way, how the changes of data are transferred, two types of data transfer are described as following:

- **Complete data transfer**
  Complete data transfer means, that the complete data will be transferred by modification of a model, without to detect, which parts have or how much has been changed in the model. This mechanism makes the transfer of data simple to implement, but a lot of resources will be wasted, if the model is very complex and needs a large space to save it.

- **Incremental data transfer**
  Some exchange or integration systems can detect the changes of data and transfer only the increment to other models. These systems use prepared dependencies to identify and compare the parameters of different data formats. The benefits of these systems are the concentration of all involved engineers on new parts instead of a new version even for uninvolved domains and the shortening time of transfer process. In most situations, there is no need to transfer the data that is not changed, since it is a waste of system resources, like memory space, cpu time, etc. The detection and identification of changes is mainly done in system specific data exchange solutions or managed by a control center (refer to Figure 6.1). The control center has a mechanism to identify changes in a model.

According to the way, how the data transfer is triggered, there are two types of data transfer:

- **Automatic transfer**
  The solutions to transfer data vary in the type of transferring changes. Some systems use an application or a middleware to control the data transfer. In this case, the data transfer will be started automatically, as soon as a change in the models is detected.

- **Transfer on demand**
  Some systems transfer the data changes only when a demand is requested. It should not be processed automatically. In this case, it is very important to define, when the data transfer is triggered. The process of data transfer can also be controlled by an application, a middleware, or by the user manually. When a demand or an event of a data transfer is detected, a data exchange among the models will be started.

6.1.4 Granularity of Dependencies

The granularity of modeled dependencies is divided into three categories:

- **Data Management** (e.g., STEP, Component DB [Vornholt and Geist, 2008]): The data management focuses on the data itself. The parameter values or the different models are stored and managed. Dependencies can be defined on the basic level of variables. A common model structure or integration is necessary.

- **Document and Process Management** (e.g., PDM, PLM): The documents and files are managed. Dependencies of documents (e.g., version, variant) are managed as well as access and responsibility roles. The document management often integrates process management.

- **Meta-Data Management** (e.g., link DB [Geist and Vornholt, 2009], Ontologies): Here meta-data, like the structure of the products, the contained models, and vocabulary, is used to define dependencies between different models.

6.1.5 Access Level

For data integration, it is necessary to obtain information from local applications. The access level for the integration is a characteristic for data exchange and integration methods. The different levels are:

- **File**: Files of the systems are used to exchange information. For example: Figures 5.4 and 5.5 use an external transfer application based on files.

- **Database**: The exchange system uses the storage database or internal storage system of the applications, e.g., used in Figure 5.2.

- **Application**: The application itself is accessed and internal or external functions are used to get the data (see Figure 5.5).

6.1.6 Storage Form

Heterogeneous data in the integration systems can be stored in different repository systems. The different repository systems are:

- **File system**: The organization of this storage form is simple, but it causes some disadvantages, such as data redundancy, not supporting to the concurrent access, not convenient to write the application programs and so on. So it is usually used to some simple systems whose functions are very clear and in which there is no data sharing.

- **Database** (DB): Compared to file system Database has many advantages to store data. It reduces the data redundancy, realizes the data sharing, reinforces the data management and so on. So DB provides a good environment for the data storage.

- **Data Warehouse** (DW): Data Warehouse has been developed based on Database. It increases the function of the strategic decision and the analysis of the data. When the data comes from different DBs, DW should be used. For example, DW is applicable to the heterogeneous data in PDM systems.

6.1.7 Granularity of Data Management

The granularity of data management is divided into two categories:

- **Files**: Data is stored in the form of files. So the data is managed in the unit of files.
- **Objects**: When data is stored in database or data warehouse, it is managed in the unit of objects.

6.1.8 Summarized Categories

In this section, we summary the categories presented in the previous sections by using the Figures 6.2 and 6.3. Figure 6.2 illustrates the categories of data exchange in virtual engineering and the main dependencies between them, while Figure 6.3 illustrates the categories of data integration in virtual engineering and the main dependencies between them.

The symbols used in these two figures are explained as following:

- A **rectangle** represents a category.
- A **ellipse** specifics a variety of selections under a category.
- A **arrow** illustrates a dependency that has to be considered during making a decision.

Data Exchange in Virtual Engineering

In Figure 6.2, on the first level a category named ”collaboration” is defined. His subcategories include linear engineering, concurrent engineering and simultaneous engineering. This means, that on the principle of collaboration, there are three choices to select: linear, concurrent and simultaneous. On the third level, there is only a rectangle named ”exchange”. This means, that his subcategories (on the fourth level) belong to data exchange in virtual engineering. For example, on the principle of the time of data transfer, there are two types of data exchange solutions: transfer on demand and automatic transfer. If the data management solutions are divided according to the way of data transfer, then there are two different categories: complete transfer and incremental transfer. The other cases have the similar meanings and will not be described.

Linear design (colored with dark gray) with data exchange is based on consecutive forwarding of system specific data. The data is completely forwarded to the next domain when the work of the first domain is finished. So the transfer time belongs to on demand while the transfer data complete. Data exchange in linear design used either system neutral format or system specific data exchange. For the user interface only the separated domain models can be viewed by users.

In **concurrent and simultaneous design**, to support the engineers, automatic exchange of changes enables current data for affected domains. The two kinds of data exchange strategies (system specific and neutral format) are both appropriate to concurrent and simultaneous design. However, in current concurrent and simultaneous design system specific data exchange is rarely used of which the disadvantages have been outlined in Section 4.2. For the user interface the separated views are used to these both designs.

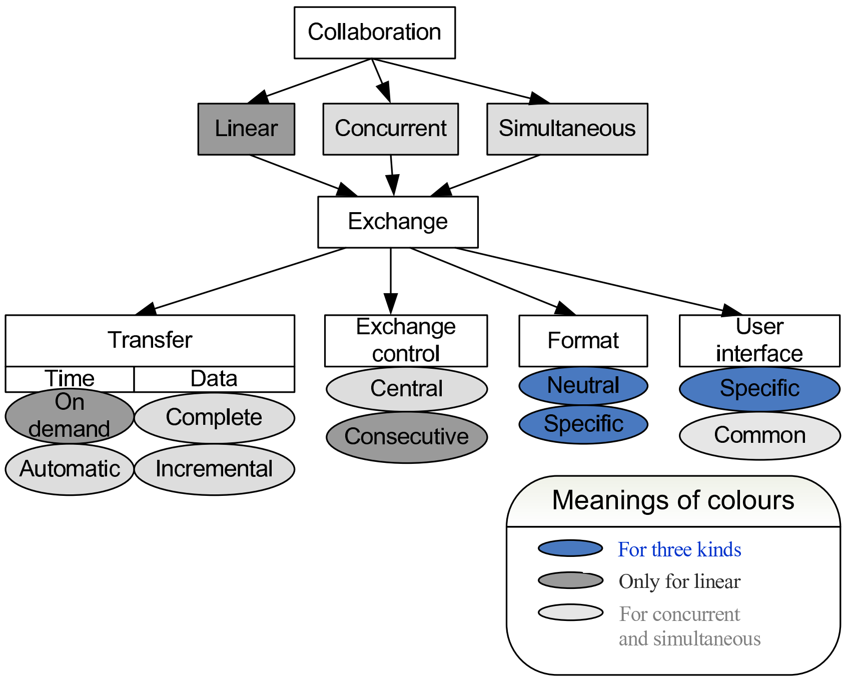


Figure 6.2: Categories for data exchange in Virtual Engineering

Data Integration in Virtual Engineering

In Figure 6.3, on the first level is the collaboration type (concurrent and simultaneous).

Here seven categories, including data transfer, data management granularity, dependency granularity, storage form, access level, architecture and user interface, are used to classify the solutions for data integration.

Some approaches require approaches of other categories: The granularity of data management is based on the storage form. If the data is stored in file system, it will be managed in the level of files (file system $\leftrightarrow$ files). If the data is saved in a database (DB) or in a data warehouse (DW), it will be managed in the level of objects (DB $\leftrightarrow$ objects, DW $\leftrightarrow$ objects). The storage form is based on the granularity level of dependencies. Files are arranged in document management systems, while databases are used to store data or meta data. Data warehouses integrate data and process information. Each system can support different specifications of one category. A central exchange control uses an architecture (exchange control central $\rightarrow$ architecture). Common object architecture has to consider a data format. An architecture including different systems can easy define a common interface (inclusion → common). The application is accessed in the inclusion architecture (inclusion ↔ application). The architectures of common objects and managed data are both defined as specific or common views on them (common objects ↔ user interface, managed data ↔ user interface), while the inclusion architecture common views (inclusion ↔ common).

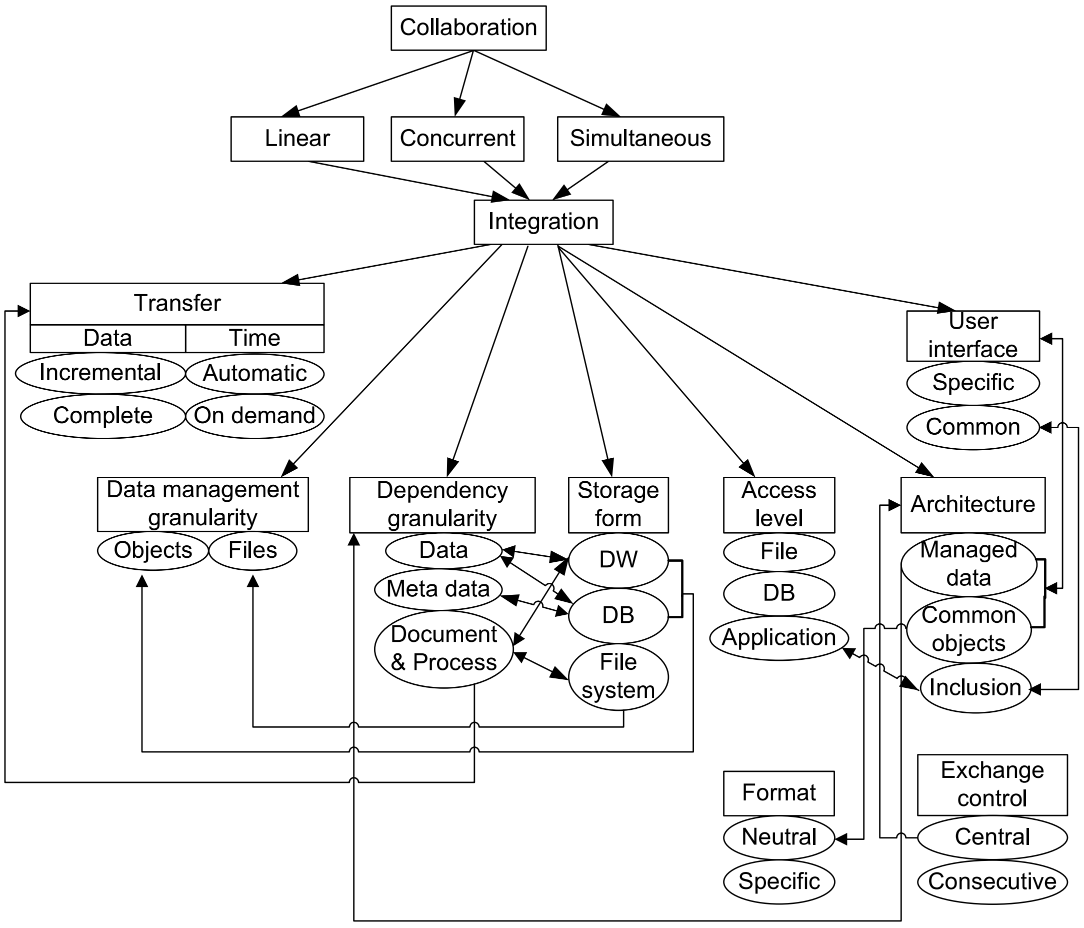


Figure 6.3: Categories for data integration in Virtual Engineering

6.2 Exemplary Classification

As introduced in section 4.3.2, STEP is an ISO standard for product data exchange, storage, and access. The STEP database also illustrates that data exchange and data integration are not disjoint categories. Even when the concept behind both solutions is different, standards can be used for data integration.

Figure 6.4 shows the classification of linear collaboration using STEP. Files are transferred on demand. STEP files exchange all existing data between the systems and the format is system neutral. Since both designers use their own application, the interface is not shared. The designers work on files and exchange data without meta-information.

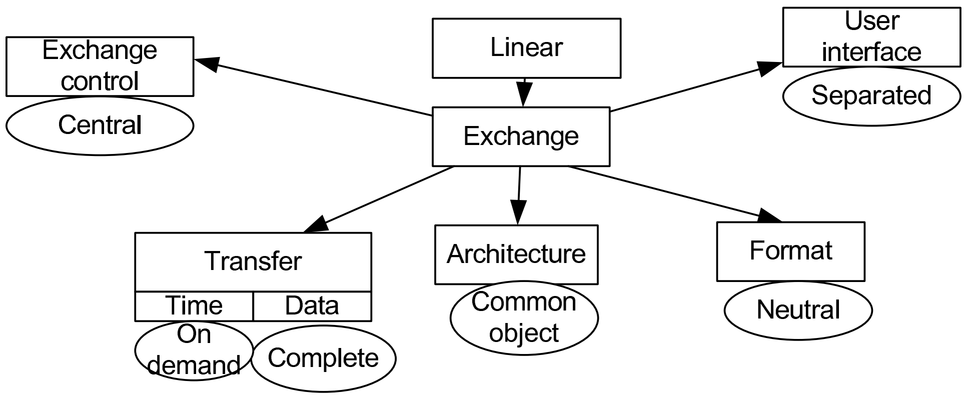


Figure 6.4: Classification of STEP

PDM systems manage the product life cycle and make the optimization for the interaction of product design, manufacturing, and life cycle activities. Product structure and composition can be read from the CAD or PDM systems.

PDM systems (e.g., Windchill designed by PTC) are used to control the product information and processes in the PLC. The functionalities in those systems are wide spread. They provide 3D-visualization and common user interfaces of CAD systems. Some applications are integrated, e.g., for Windchill: MCAD-Tools, AutoCAD. Furthermore, it is connected to many mechanic and electronic CAD-applications by external online functions. The granularity is limited to data management approaches. However, the data item storage is less integrated and dependencies between parameters require further external or internal defined functions.

Figure 6.5 describes a typical PDM solution. Managed exchange, concurrent, and simultaneous work are supported. Furthermore, it enables common as well as separated views. However, it is limited to the document and meta management dependency level.

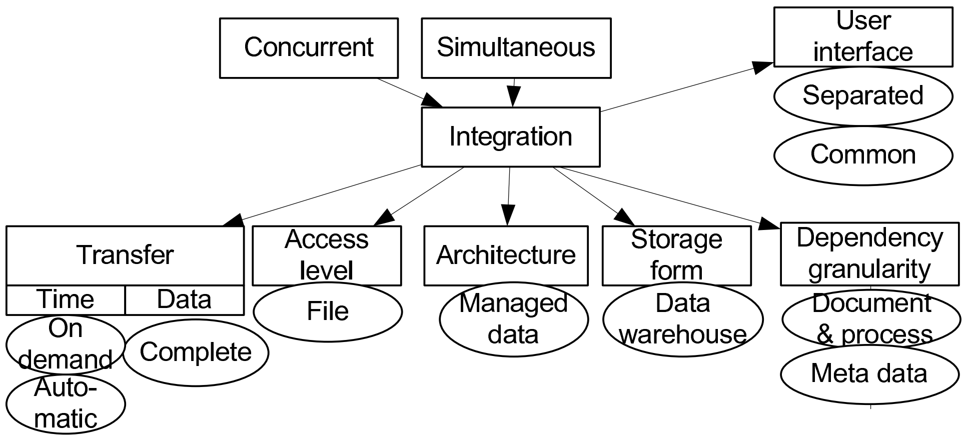


Figure 6.5: Classification of PDM

Integrated Database, e.g., Concurrent Component Interface (CCI): There exist data incompatibilities and inconsistencies between legacy and new database schemata [Abbasifard et al., 2006], so in order to deal with these problems the extraction of data from legacy systems and integration into new systems are necessary. The incompatibilities may be semantic or quantitative. In [Vornholt and Geist, 2008] the authors describe a data integration model based on components which contain heterogeneous model-parts. The containers for different part-models have their own definable service interface to define parameters. The main task of this approach is to enable consistency control between different models on the parameter level, by considering the possibility of an own view for every designer and easy definition of combined views.

Figure 6.6 classifies the described component interface integration solution. The scope of this approach lies on the data dependency level, with common objects and incremental data transfer. Thus, it can be used for concurrent and simultaneous design.

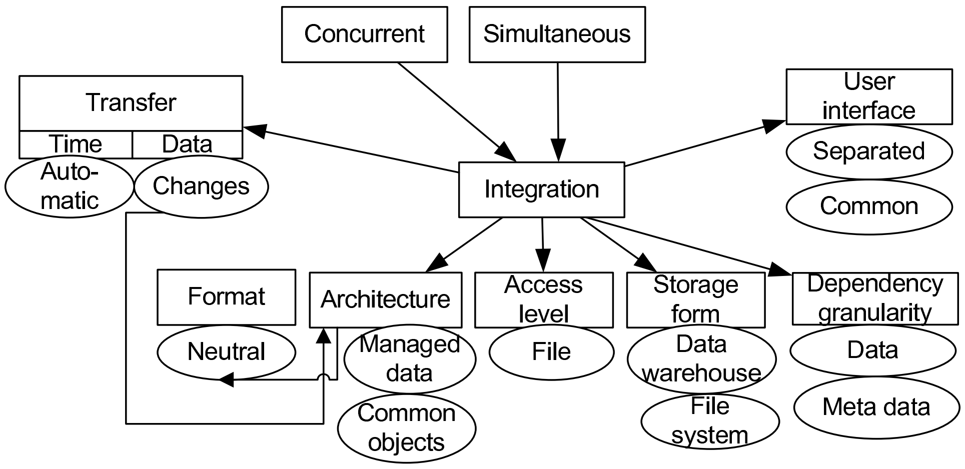


Figure 6.6: Classification of CCI

6.3 Discussions

There is a lack of methods for data exchange between various tools in Virtual Engineering. Therefore, integration of design, analysis, and simulation tools becomes important as a research topic. Thus a fault-tolerant system should ensure trust-worthy data and any possible errors should be quantified. Since a Virtual Prototyping describes the product data from different aspects, it should provide a collaborative design environment. An open architecture is required, where different new domains from Virtual Engineering can be integrated. Then it helps to get the necessary, combined information.

Based on the above presented requirements typical advantages of the exemplary classified concurrent support applications are discussed (see Table 6.1):

- All time thrust-worth data means to an engineer to work on the current data. He will be informed when changes occur. So the collaboration needs managed dependencies or a common data model with automatic propagation of changes. This characteristic is only existing at integrated data models and limited to the information of changes (if the engineer is affected by them or not) using PDM solutions.

- The supported collaboration form by exchange formats is limited to the exchange. CCI and PDM systems enable simultaneous work. However, simultaneous and concurrent solutions should use a form of integration and a common storage system.

- Integration of new domains into PDM and CCI is complex because a new common structure has to be defined. System neutral exchange standards are easy to maintain, system specific need two half links per connected system.
- While the granularity of exchange standards and CCI enable the access on the parameter, PDM systems are limited to the structure and transfer of files.

	CCI	Exchange standard	PDM
always trust worthy data	yes	no	limited
collaboration	simultaneous	linear	simultaneous
new domains	complex	2 or 2 × n half-links	complex
granularity	parameter	parameter	structure

Table 6.1: Comparison of CCI, exchange standards, PDM

6.4 Conclusion

This chapter presented a categorization of approaches for concurrent virtual engineering. Based on three types of collaboration, the main approaches for exchange and integration of product data were described. These categories and their dependencies as well as the different possibilities to handle them were in details illustrated. This categorization enables an overview and classification of different applications. On this base, it is possible to identify advantages and disadvantages of approaches in collaborative virtual engineering.

Chapter 7

Analysis of Data Exchange and Integration Solutions

For concurrent engineering, we have discussed the concurrent design and the simultaneous design (see the section 2.3.1). And for linear engineering, we have introduced the linear design (see the section 2.3.1). Besides, we have also presented the base principles and approaches of data exchange and data integration. At the same time, we have pointed out a few factors which could affect on choosing a approach to design models in a engineering. Theses factors are classified into several categories in the last chapter. In order to illustrate the process of the choosing a suitable solution for data exchange or integration, we introduce the concept of "decision tree" in this chapter.

7.1 About Decision Tree

This chapter answers the questions, what is a decision tree, why it is needed and how it is used to get a decision.

7.1.1 Concept of Decision Tree

A decision tree is one of the most systematic tools of decision-making theory and practice. It is a classifier in the form of a tree structure, where each node is either a leaf node or a decision node. A leaf node indicates the value of the target attribute which represents a classification or decision, while a decision node specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test. That is, each branch node represents a choice between numbers of alternatives.

As a decision making tree, it is essentially a diagram that represents, in a specially organized way, the decisions, the main external or other events that introduce uncertainty, as well as possible outcomes of all those decisions and events.

7.1.2 Usage of Decision Tree

A decision tree is particularly helpful in situations of complex multistage decision problems. For example, when you need to plan and organize a sequence of decisions and take into account how the choices made at earlier stages and the outcomes of possible external events determine the types of decisions and events at later stages of that sequence. It can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.

Decision tree induction is a typical inductive approach to learn knowledge on classification. The key requirements to do mining with decision trees are:

- Attribute-value description: object or case must be expressible in terms of a fixed collection of properties or attributes. This means that we need to discrete continuous attributes, or this must have been provided in the algorithm.
- Predefined classifications (target attribute values): the categories to which examples are to be assigned must have been established beforehand (supervised data).
- Discrete classifications: a case does or does not belong to a particular classification, and there must be more cases than classifications.
- Sufficient data: usually hundreds or even thousands of training cases.

7.1.3 Evaluation

The advantages of decision tree methods are:

- Decision trees are simple to understand and interpret. After a brief explanation people are able to understand decision tree models well.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods are:

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
- Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.
- Decision trees do not treat well non-rectangular regions. Most decision-tree algorithms only examine a single field at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.

7.2 Analysis using Decision Trees

The last section illustrates the basic concepts and the usage of decision trees. In this section, we will build decision trees for data exchange and data integration. With the help of these decision trees we can analyze, in which case which approach should be accepted to process a action (data exchange or data integration).

7.2.1 Factors for Classification

- **Version control**
  In PLC modifications and revisions of the products can be done. So there are different kinds of versions for the products. Keeping the different versions stable and having a good control of the version is very important to data exchange and integration.
- **Consistency proof**
  The changes of the data may happen at every time. In order to keep the data in both sending systems and receiving systems consistent, the changed information should be sent to the receiving system in time and both sending systems and receiving systems should update in time.
- **Synchronous communication**
  If the different systems depend on each other, then these systems are not isolated. They should communicate to each other about the exchanged data synchronously.
- **The number of working systems at the same time**
  The work of data exchange carried on more easily when there are less systems working at the same time.
- **The complexity of the data to be exchanged**
  This factor has an effect on the kinds of data exchange and integration.
- **Control management**
  In the process of data exchange and integration adding the control management mechanism can improve the ability to the control of the time and the data.

7.2.2 Explanation to Decision Trees

This section analyzes the different factors that have an effect on the decision of data exchange and integration. To do this we build at first a decision tree for both data exchange and data integration. This decision tree resides in Figure 7.1. Besides, there are two decision trees created in this chapter. The one is for data exchange and displayed in Figure 7.2. The other is for data integration and displayed in Figure 7.3.

In these three decision trees, the meanings of the used symbols are explained as following:

- **A white rectangle** represents a category.
- **A gray rectangle** represents a specific approach.

- **A Ellipse** represents a decision factor.
- **The dashed rectangle** in Figure 7.2 represents that its subtree is illustrated in Figure 7.3.
- **The dashed rectangles** in Figure 7.3 represent that their subtrees are illustrated in Figure 7.2.

The implementation of concurrent design and synchronous design reduces the design time. Since linear design is carried out step by step, it costs longer time than concurrent design or synchronous design by building the same model. So taking the measure of linear design means, that there are no constraints for the **demanded design time**. Both concurrent design and simultaneous design have the **synchronous communication**, but linear design does not (see the red arrows in Figure 7.1). With concurrent design, the designing of one model does't depend on the designing of others. But in simultaneous design, the designing of a model can't begin until the models on which he depends finish their designs. That is, in concurrent design **the degree of dependency** of one model on others is lower than in simultaneous design.

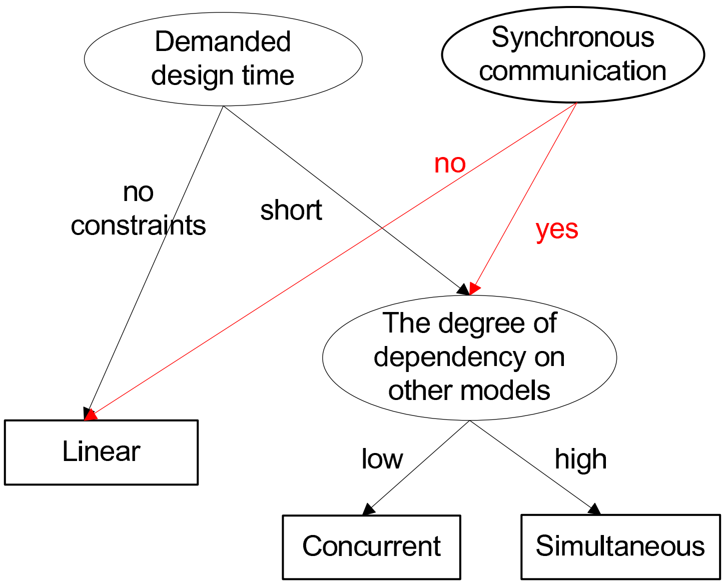


Figure 7.1: Decision tree for design of models

The dependency between different systems is relative small in data exchange (see Figure 7.2). According to the different approaches of exchange control (**central** or **consecutive**), there are five elements which can be used as the decision conditions. If **the amount of the data to be exchanged** is large, we need to construct a central management to control the data exchange, including the time intervals of each pair of the exchanged data and the update time of each system. Only if the amount of the exchanged data is not too large and can not cause the chaos in the data exchange process, we let the implement of data exchange be performed consecutively. If the exchanged data is too complex, it is better to use the central control. Otherwise the consecutive data exchange is appropriate. If we want the updating of the changed data to be fast performed, we can use the central control. The synchronous communication is implemented in consecutive data exchange. But in central, whether control synchronous communication happens depends on the previous settings.

There are three decision conditions according to which a format (neutral or special) is used in data exchange. If new systems for exchanging data are added, it is easier to use neutral format than specific format. The reason is that if we use the specific formats and if we want to add a new system to n systems, we need to write 2 × n new programs to realize the data exchange. However, if we use the neutral formats, what we must do to add a new system is to perform only once data translation. That is, we translate the data format into the neutral format and then the work is finished. Neutral formats are necessary for the synchronous communication. Different systems translate the data into neutral formats; the changes of the exchanged data are updated in time. When using specific formats the programs for exchanging data can not be updated or rewritten in time. This is not practical, so there is no synchronous communication using specific formats. There is no consistency proof when specific formats are used. Once the programs are written, they are fixed because of the complex work of rewriting programs or its modification. The changes of the data can not transfer to other system right now, so the consistency of the data can not be guaranteed. There is no problem with programs when using neutral format, so the consistency is guaranteed.

There are two forms of transfer time. If we want to **control the transfer time**, then we choose the approach of on demand. Otherwise we choose the automatic approach. If **the dependencies of the exchanged data** in different systems are big, we must make a control on the data exchange. So the approach of on demand should be used. Otherwise the automatic approach works equally.

If the data models of one system are translated into the form of the other system, we can translate all the data or part of the data. If the data models include not too much data, then we could translate the complete models. If there is much data, only the changes of the data can be translated.

After choosing the data integration there are several decision factors on the sub level of the data integration (see Figure 7.3). In access level there are approaches of file, database and application. If the integrated systems are not too complex, the data access can be accomplished by the access on the application of the system itself. If the integrated systems are less complex, the data access should be accomplished with the help of files and database. If **the number of the accessed data** is small, we access the data stored in files. Otherwise we access the data stored in database.

The architectures of the data integration include common objects, managed data and inclusion. The common feature of common objects and managed data architectures gets the needed data from the source systems to generate the integrated systems. So these two architectures have the need of integrating the existing systems into one. Inclusion means that one system gets data from other systems to become an integrated system itself. It does not mean generating a new integrated system from these existing systems.

The storage form is dependent on **the amount of storage data**. The file system is usually used to some simple systems whose functions are very clear and in which there is no data sharing. Otherwise, the database system and data warehouse can organize data in very complex forms. Besides, data warehouse can store a large amount of data.

There are three categories of the dependency granularity, including data management, document and process management, and meta data management. Because not only the

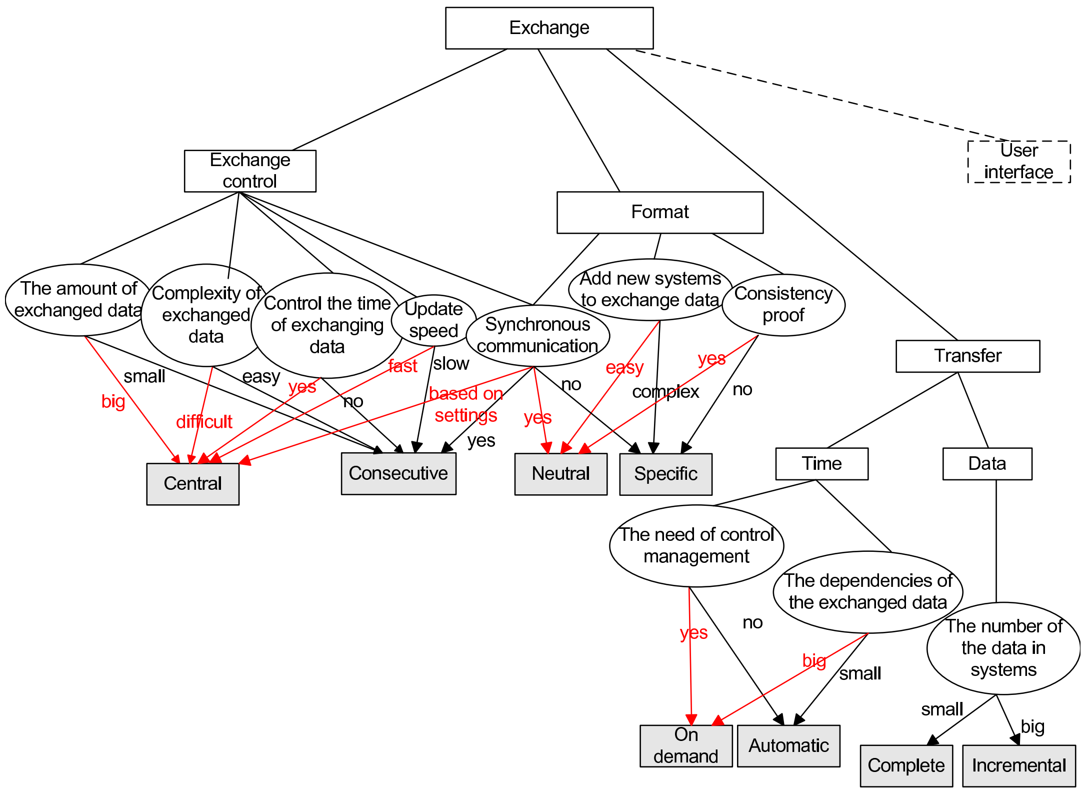


Figure 7.2: Decision tree for data exchange

data it self but also more management information is included in the document and process management and meta data management, the **version control** is relative easy to these both categories. These two categories have more completeness of the data management than the first category.

Special user interface and common user interface are two types of user interface used in data management. If we integrate different systems of different domains into new systems, the new systems cover the data from different domains. In order to get a global view and keep **the unity of the integrated system domains**, we should choose the common interface. If we only want to the part of data of some domain in the integrated systems, separated interface is a god choice.

7.3 Conclusion

This chapter illustrated the process of choosing a suitable solution for data exchange or integration by building the decision trees of them. These decision trees point out, in which case which approach should be accepted to process the data exchange or integration.

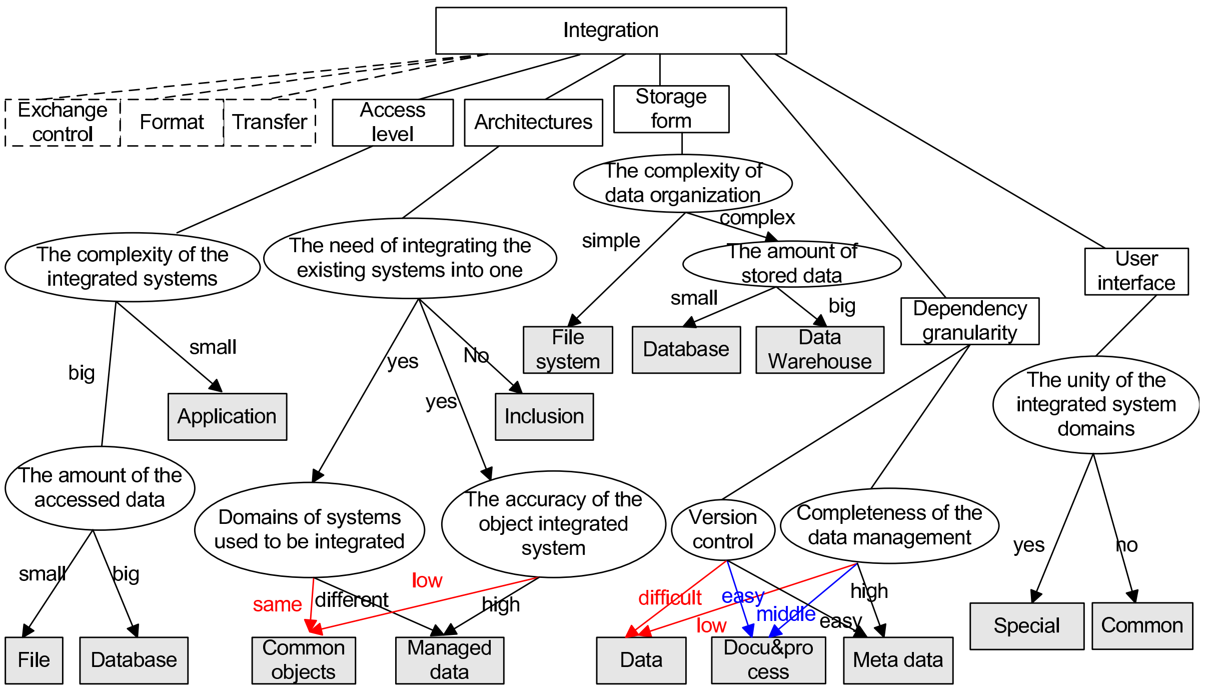


Figure 7.3: Decision tree for data integration

Chapter 8

Conclusion and Further Works

8.1 Conclusion

In this thesis we have discussed different solutions for heterogeneous data exchange and integration. A good solution for that optimizes the Concurrent Virtual Engineering whose goal is to satisfy the requirements of enhancing the product quality and reducing the costs and the time needed to bring product to market.

As the first step we outlined the Concurrent Engineering and Virtual Engineering and described both of them in detail. The combination of them is named Concurrent Virtual Engineering, which has large advantages in current engineering. So the traditional Linear Engineering has been replaced by it gradually.

This thesis puts his focus on the data modeling and solutions for data exchange and data integration in Concurrent Virtual Engineering. To explain these points exactly, we took the concept of a light bulb as example and built several models like the geometry model, the material model, the thermal model and the light model. During building these models, the process of product data modeling was introduced, at the same time the relationship and the dependency of one model upon the others were concrete defined. This provided a base for explaining the data exchange and data integration in the next two chapters, because the data exchange between models and the integration of one model into another one are both performed depending on the relationship between the models. By using these models as example, the strategies and the process of data exchange were detailed. Beside this, the basic architectures of data integration solutions and the approaches to process data integration are also described using the previous defined models of a light bulb.

We also introduced several mathematical theories to explain some complex concepts and solutions in this thesis. The automaton formalism theory introduced in the section 4.4 with the reference to [Vergeest and Horváth, 1999] was used to illustrate the process of data exchange. They theory introduced in the section 4.5 with the reference to [Eckert et al., 2005] is used to explain the problems occurring during the process of data exchange. In the chapter 7 we used the theory of decision trees to analyze the solutions for data exchange and data integration. To this purpose, we built the decision trees to present the categories of the solutions. It is possible to identify the advantages and disadvantages of a approach on these decision trees. Besides, they described how to choose a suitable solution according to the practical requirements. With the help of these theories and tools the key points of thesis were concrete introduced.

In addition, we have illustrated some standards, like IGES, STEP and PDML, which are developed for specification of data exchange between models in Concurrent Virtual Engineering. The most popular solution for data exchange is to use the neutral format. STEP is a good tool for such a purpose. For data integration we introduced the PDM systems which provides a platform for the company information management, where data exchange, data integration and data analysis are integrated in a platform.

8.2 Further Works

Based on the results in this thesis, further works of data exchange mechanisms can be performed. They are described in the following:

- The basic unit of the exchanged data contains both the content of the data and the representation of data. This form can make the preparation of the exchanged data faster. The receiving system can more easily understand the received data. Of course this situation demands more for the standards.
- Mechanisms should be appropriate not only to static data but also to dynamic data. This demands a high level of the control mechanisms to the exchange of the dynamic data.
- Mechanisms can balance the differences between different systems. If the receiving system has a data model with a higher level of detail than the sending system, the standards must provide the details of the exchanged data from the sending system as much as possible.
- Mechanisms should try to avoid the data loss during data transfer. Satisfying this requirement needs the data exchange mechanisms to have relative complete exchange standards.
- The data exchange between systems must solve the problem of semantics consistency. Ontology engineering has provided one solution for this problem.

The following are some suggestions to the future works of data integration.

- Mechanisms that integrate different databases into one should be deeply researched. These mechanisms are combined both data storage and data management.
- Templates for data integration in one domain can be designed. With the help of the templates the generation of the different translating mechanisms becomes much easier.
- The widespread use of Web technologies and electronic business makes the combination of systems integration and Web technologies necessary. How to build the interfaces between the XML files (or the agents) used in product data modeling and Web applications becomes a research topic.

Bibliography

[Abbasifard et al., 2006] Abbasifard, M. R., Rahgozar, M., Bayati, A., and Pournemati, P. (2006). Using automated Database Reverse Engineering for Database Integration. *World Academy of Science, Engineering and Technology*, 19:338–342.

[AL-Timimi and MacKrell, 1996] AL-Timimi, K. and MacKrell, J. (1996). *STEP: Towards Open Systems*. CIMdata.

[Alshawi and Ingirige, 2003] Alshawi and Ingirige (2003). Web-enabled Project Management: an Emerging Paradigm in Construction. *Automation in Construction, v12 i4*, 12(4):349–364.

[Aziz et al., 2004] Aziz, H., X.Gao, J., M.Ceung, W., and G.Maropoulos, P. (2004). A Design Environment for Product Knowledge Management and Data Exchange. In Tichkiewith, S. and Brissaud, D., editors, *Methods and Tools for Co-operative and Integrated Design*, pages 257–266. Kluwer Academic Publishers: Dordrecht.

[Beech and Feldman, 1983] Beech, D. and Feldman, J. S. (1983). The integrated data model: A database perspective. In *VLDB '83: Proceedings of the 9th International Conference on Very Large Data Bases*, pages 302–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Bernard, 2005] Bernard, A. (2005). Virtual Engineering: Methods and Tools. In *Keynote Paper of International Conference on Advanced Research in Virtual and Rapid Prototyping*, pages 413–421. Professional Engineering Publishing.

[Bettaieb et al., 2004] Bettaieb, S., Frédéric, and Tichkiewitch, S. (2004). Interface between CAD/CAM Software and an Integrative Engineering Design Environment. In Tichkiewitch, S. and Brissaud, D., editors, *Methods and Tools for Co-operative and Integrated Design*, pages 315–326. Kluwer Academic Publishers: Dordrecht.

[Bullinger, 2002] Bullinger, H.-J. (2002). Virtual Engineering - Neue Wege zu einer schnellen Produktentwicklung. Fraunhofer Publica.

[Burdea and Coiffet, 2003] Burdea, G. C. and Coiffet, P. (2003). *Virtual Reality Technology*. John Wiley & Sons, Inc., New York, NY, USA.

[Canty, 1987] Canty, E. J. (1987). Simultaneous Engineering: Expanding Scope of Quality Responsibility. Digital Equipment Corporation. White Paper.

[Capoyleas et al., 1996] Capoyleas, V., Chen, X., and Hoffmann, C. M. (1996). Generic naming in generative, constrain-based design. *Computer-Aided Design*, 28(1):17–26.

[Cecil and Kanchanapiboon, 2007] Cecil, J. and Kanchanapiboon, A. (2007). Virtual Engineering Approaches in Product and Process Design. *Int J Adv Manuf Technol*, 31:846–856.

[Chen and Hoffmann, 1995] Chen, X. and Hoffmann, C. M. (1995). On editability of feature-based design. *Computer-Aided Design*, 27(12):905–914.

[Dai et al., 1996] Dai, F., Felger, W., Frühauf, T., Gobel, M., Reiners, D., and Zachmann, G. (1996). Virtual Prototyping Examples for Automotive Industries. In *In Proc. Virtual Reality World*, pages 13–15.

[de Andrade and Forcellinib, 2007] de Andrade, L. F. S. and Forcellinib, F. A. (2007). *Interface Design of a Product as a Potential Agent for a Concurrent Engineering Environment*, chapter 10, pages 503–510. Geilson Loureiro and Richard Curran.

[Dittrich and Jonscher, 2000] Dittrich, K. R. and Jonscher, D. (2000). All together now: Towards integrating the world's information systems. In *JISBD*, page 7.

[Dyla, 2002] Dyla, A. (2002). *Modell einer durchgänig rechnerbasierten Produktentwicklung.* Phd thesis, Technische Universität München, Germany.

[Eastman et al., 1991a] Eastman, C., Bond, A., and Chase, S. (1991a). Application and Evaluation of an Engineering Data Model. *Research in Engineering Design*, 2(4):185–208.

[Eastman et al., 1991b] Eastman, C. M., Bond, A. H., and Chase, S. C. (1991b). A formal Approach for Product Model Information. *Research in Engineering Design*, 2(2):65–80.

[Eckert et al., 2005] Eckert, R., Mansel, W., and Specht, G. (2005). Model Transfer among CASE Tools in Systems Engineering. *System Engineering*, 8(1):41–50.

[Estublier and Vega, 2007] Estublier, J. and Vega, G. (2007). Reconciling software configuration management and product data management. In *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 265–274, New York, NY, USA. ACM.

[F-L et al., 1994] F-L, K., Kieswetter, T., and Kramer, S. (1994). Distributed Product Design. *CIRP Annals - Manufacturing Technology*, 43(1):149–152.

[Fowler, 1995] Fowler, J. (1995). *STEP for Data Management, Exchange and Sharing.* Technology Appraisals.

[Gabbert and Wehner, 1998] Gabbert, U. and Wehner, P. (1998). The Product Data Model as a Pool for CAD-FEA Data. *Engineering with Computers*, 14(2):115–122.

[Garcia Molina and Kogan, 1988] Garcia Molina, H. and Kogan, B. (1988). Node autonomy in distributed systems. In *DPDS '88: Proceedings of the first international symposium on Databases in parallel and distributed systems*, pages 158–166, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Geist and Vornholt, 2009] Geist, I. and Vornholt, S. (2009). Eine Link-Datenbank zur Integration von Virtual Engineering-Daten. In *Grundlagen von Datenbanken*, pages 45–49.

[Hamri and Léon, 2004] Hamri, O. and Léon, J.-C. (2004). Interoperability between CAD and simulation models for cooperative design. In *Methods and Tools for Co-operative and Integrated Design*, pages 451–462. Kluwer Academic Publishers: Dordrecht.

[Hislop et al., 2004] Hislop, D., Lacroix, Z., and Moeller, G. (2004). Issues in Mechanical Engineering Design Management. *SIGMOD Rec.*, 33(2):135–138.

[Hoberman, 2009] Hoberman, S. (2009). *Data Modeling Made Simple: A Practical Guide for Business and IT Professionals.* Take IT With You(r) Series.

[Jian et al., 2006] Jian, C. Q., Lorra, M. A., McCorkle, D., and Bryden, K. M. (2006). Applications of Virtual Engineering in Combustion Wquipment Development and Engineering. *ASME International Mechanical Engineering Congress and Expo, IMECE2006*, 119:1159–1164.

[Jo et al., 1993] Jo, H. H., Parsaei, H. R., and Sullivan, W. G. (1993). Principles of Concurrent Engineering. In *Concurrent engineering*, pages 3–23. Parsaei and Sullivan.

[Kawabata et al., 2002] Kawabata, R., Hasegawa, A., Kumagai, S., and Ithoh, K. (2002). Integrated Collaborative & Concurrent Engineering Environment. In *International Conference on Integrated Design and Process Technology (IDPT).*

[Kripac, 1997] Kripac, J. (1997). A Mechanism for Persistently Naming Topological Entities in History-based Parametric Solid Models. *ADM Trans on Computer Graphics*, 29(2):113–122.

[Lessa et al., 2000] Lessa, A. F., de Miranda Freitas, A. A. d. M. F., Cameira, R. F., and Walker, R. A. (2000). Evaluation of an integrated engineering environment in an enterprise resources planning system.

[Maletic et al., 2001] Maletic, J. I., Leigh, J., and Marcus, A. (2001). Visualizing Software in an Immersive Virtual Reality Environment. In *in Proceedings of ICSE'01 Workshop on Software Visualization*, pages 12–13. Society Press.

[McDonald, 1997] McDonald, J. E. (1997). Data Sharing, Interoperability and Testing in a Virtual Environment. SimTecT Papers.

[Mowbray and Zahavi, 1995] Mowbray, T. and Zahavi, R. (1995). *The Essential COBRA: System Integration Using Distributed Objects.* Wiley & Sons.

[Newman et al., 2003] Newman, S., Allen, R., and Rosso, R. J. (2003). CAD/CAM Solutions for STEP Compliant CNC Manufacture. *International Journal of Computer Integrated Manufacturing*, 16:590 – 597.

[NIST, 1993] NIST (1993). Federal information processing standards publication 184 - announcing the standard for integration definition for information modeling. Website. Available online at `http://www.itl.nist.gov/fipspubs/idef1x.doc`; P.71 visited on May 20th 2010.

[Noël et al., 2003] Noël, F., Brissaud, D., and Tichkiewitch, S. (2003). Integrative Design Environment to improve Collaboration between Various Experts. *Ann CIRP*, 52(1):109–112.

[Oh et al., 2001] Oh, Y., hung Han, S., and Suh, H. (2001). Mapping Product Structures between CAD and PDM Systems using UML. *Computer-Aided Design*, 33:521–529.

[Owen, 1997] Owen, J. (1997). *STEP an introduction*. Information Geometers, Winchester UK. 2nd ed.

[Pennel and Winner, 1989] Pennel, J. and Winner, R. (1989). Concurrent Engineering: Practices & Prospects. In *Global Telecommunications Conference and Exhibition Part 1.*, volume 1, pages 647–655. Institute for Defense Analyses, IEEE.

[Pop et al., 2004] Pop, A., Johansson, O., and Fritzson, P. (2004). An Integrated Framwork for Model-driven Product Design and Development using Modelica. In *45th Conference on Simulation and Modelling of the Scandinavian Simulation Society (SIMS2004)*, pages 23–24, Copenhagen, Denmark.

[Prasad et al., 1997] Prasad, B., Wang, F., and Deng, J. (1997). Towards a Computer-supported Cooperative Environment for Concurrent Engineering. *Concurrent Engineering*, 5(3):233–252.

[Raghothama and Shapiro, 1998] Raghothama, S. and Shapiro, V. (1998). Boundary Representation Deformation in Parametric Solid Modelling. *ADM Trans on Computer Graphics*, 17(4):259–286.

[Roucoules and Tichkiewitch, 2000] Roucoules, L. and Tichkiewitch, S. (2000). Code: a Cooperative Design Environment- a new Generation of CAD Systems. In *Concurrent Engineering: Research and applications*, volume 8, pages 263–280. Academic Press.

[Schönhoff et al., 1997] Schönhoff, M., Strässler, M., and Dittrich, K. R. (1997). Data Integration in Engineering Environments. In *EFDBS*, pages 45–56.

[Sheth and Larson, 1990] Sheth, A. P. and Larson, J. A. (1990). Federated Database Systems for Managing Distributed,Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236.

[Shukla et al., 1996] Shukla, C., Vazquez, M., and Chen, F. F. (1996). Virtual Manufacturing: an Overview. *Computers ind. Engng*, 31(1/2):79–82.

[Spur and Frank-Lothar, 1984] Spur, G. and Frank-Lothar (1984). *CAD-Technik: Lehr- und Arbeitsbuch für die Rechnerunterstützung in Konstruktion und Arbeitsplanung.* Carl Hanser Verlag München.

[Summers et al., 2001] Summers, J., Vargas-Hernandez, N., Summers, J. D., Shah, J., Lacroix, Z., Vargas-hernandez, N., Zuozhi, Z., and Shah, J. J. (2001). Comparative Study of Representation Structures for Modeling Function and Behavior of Mechanical Devices. In *Behavior of Mechanical Devices, 21st Computers and Information in Engineering Conference, Design Engineering Technical Conference*. ASME Press.

[Vergeest and Horváth, 1999] Vergeest, J. S. M. and Horváth, I. (1999). Design Model Sharing in Concurrent Engineering: Theory and Practice. 7(105).

[Vornholt and Geist, 2008] Vornholt, S. and Geist, I. (2008). Flexible Integration Model for Virtual Prototype Families. In *Proceedings on the 5th International Conference on Product Lifecycle Management (PLM08)*. Inderscience.

[Vornholt et al., 2010] Vornholt, S., Geist, I., and Li, Y. (2010). Categorisation of data management solutions for heterogeneous data in collaborative virtual engineering. In *International Workshop on Digital Engineering (IWDE)*. ACM. to appear.

[Wang, 2002] Wang, G. G. (2002). Definition and Review of Virtual Prototyping. *Journal of Computing and Information Science in Engineering*, 2(3):232–236.

[Webster, 1989] Webster (1989). *Webster's Encyclopedic Unabridged Dictionary of the English Language*. Rh Value Publishing.

[West and Fowler, 1999] West, M. and Fowler, J. (1999). Developing high quality data models - the european process industries step technical liaison executive (epistle).

[Whitten, 2004] Whitten, Jeffrey L.; Lonnie D. Bentley, K. C. D. (2004). *Systems Analysis and Design Methods*. Irwin/McGraw-Hill.

[Whyte et al., 2000] Whyte, J., Bouchlaghem, N., Thorpe, A., and McCaffer, R. (2000). From CAD to Virtual Reality : Modelling Approaches, Data Exchange and Interactive 3D Building Design Tools. *Automation in Construction*, 10(1):43–55.

[Wiebe, 1998] Wiebe, E. N. (1998). Impact of Product Data Management (PDM) Trends on Engineering Graphics Instruction. In *Proceedings of the 1998 American Society for Engineering Education Annual Conference & Exposition.*

[Wu et al., 2001] Wu, J., Zhang, T., Zhang, X., and Zhou, J. (2001). A Face based Mechanism for Naming, Recording and Retrieving Topological Entities. *Computer-Aided Design*, 33:687–698.

[Yang et al., 2004] Yang, J., Han, S., Cho, J., Kim, B., and Lee, H. Y. (2004). An XML-based Macro Data Representation for a Parametric CAD Model Exchange. In *ComputerAided Design and Applications*, volume 1, pages 153–162.

[Yeh and You, 2000] Yeh, S.-C. and You, C.-F. (2000). Implementation of STEP-based Product Data Exchange and Sharing. *Concurrent Engineering Research an Applications*, 8:50–60.

[Zhang et al., 2000] Zhang, Y., Zhang, C., and H.P.Wang (2000). An internet based STEP data exchange framework for virtual enterprises. *Computers in Industry*, 41:51–63.

[Ziegler and Dittrich, 2004] Ziegler, P. and Dittrich, K. R. (2004). Three decades of data integration - all problems solved? In *In 18th IFIP World Computer Congress (WCC 2004), Building the Information Society*, volume 12, pages 3–12.

Declaration of Authorship

I here by declare that I am the sole author of this thesis and used nothing but the specified resources and means.

Magdeburg, den May 30, 2010

Yuexiao Li