Environment-Based Design (EBD) Approach to Enterprise Application Integration (EAI)

Bo Chen

A thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science (Mechanical Engineering) at
Concordia University
Montreal, Quebec, Canada

November 2007

Your file Votre référence
ISBN: 978-0-494-40909-1
Our file Notre référence
ISBN: 978-0-494-40909-1

ABSTRACT

Environment-Based Design (EBD) Approach to Enterprise Application Integration (EAI)

Bo Chen

In the past decades, along with the rapid updating of computer technology and extremely expanded business competition, enterprises have begun relying more and more on different applications. However, the short life cycle of these applications and high updating cost make the integration of these applications and their business process a top priority for many enterprises. In such a context, Enterprise Application Integration (EAI) has become a popular research field. EAI is the unrestricted sharing of data and business processes among any connected applications and data source in the enterprise. The goal of EAI is to integrate different applications and let them freely share the same business data and process. The benefits of EAI include cycle time reduction, cost reduction, and cost containment. EAI has attracted many developers and institutes to activity in this field. Many different enterprise application integration methods, tools, and technologies have been developed and introduced to the market; however, there is still no easy and straightforward way to solve EAI problems. The tasks of enterprise application integration are still very challenging.

This thesis uses a different and effective approach, which takes the Enterprise Application Integration (EAI) problem as a design problem, and uses the Environment-Based Design (EBD) theory to formulate this problem as well as to generate the final solutions. The advantage of EBD is that it ensures that most

defective and imperfect concepts and solutions can be eliminated during the very first stage.

The notion of Environment-Based Design methodology was first proposed by Dr. Yong Zeng, based on his Axiomatic theory of design modeling. EBD methodology includes three main stages: environment analysis, conflict identification, and concept generation. These three stages work together to generate and refine, progressively and simultaneously, the design requirements and design solutions.

The EBD-EAI approach can help EAI developers to design a successful EAI application in functions with lower costs, and a shorter development time. A real EBD-EAI problem-solving process is described in this thesis to support this new and innovative EAI problem-solving approach.

Acknowledgements

I would like to express my appreciation to my supervisor, Dr. Yong Zeng, to our group members and to my family members who have significantly supported me during the preparation of this thesis.

I would like especially to thank my supervisor, Dr. Yong Zeng. Without his constant encouragement, his knowledgeable and helpful suggestions, this thesis would not have been possible. Dr. Yong Zeng has offered me extensive and thoughtful comments throughout my entire master's program. I have benefited enormously from discussions with him. His extensive knowledge and experience in the area of design helped me guide my thesis research in the right direction. His design theories have dramatically changed my way of thinking in design, and I will benefit greatly from this in my future work.

I would also like to thank Mr. Baiquan Yan, Mr. Zhenyu Chen, Mr. Mingbin Chen, Ms. Shenji Yao, and other group members. This thesis has benefited greatly from the discussions I have had with them.

Over the years, my wife, Yihua Teng, has continually given me love and encouragement on my research and study. Even when I lost confidence in my studies, she never gave up and supported me through very difficult periods. I would like to express my deep appreciation for her unconditional love, honesty and steadfastness.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation of The Research

Since the first personal computer (PC) was introduced by Simon in 1949, PCs have been continually progressing at ever-increasing speed, memory and storage capacity, and CPU processing power. This progress has dramatically increased during the past three decades. Along with the evolution of computers, more and more new technologies related to the PC have been continually introduced, more and more applications for the PC have also been developed. As a result of rapid development of IT technologies, more and more different applications have been developed and used for almost every aspect of business. However, as computer generations have developed, most of these applications and systems have been built to serve just a single purpose. There has not been sufficient thinking about how to integrate these applications and systems into larger systems and multiple applications. Moreover, for many reasons, the speed for updating the applications by enterprise users is always far slower than the updating speed of the applications themselves. The result of these speed differences and the rapid development of information technology (IT) leads to the reality that, in many enterprises, different applications with similar functions, or different generations of the same application are in use at the same time. Often these kinds of applications can not share data and cannot process effectively. Under the pressure of a competitive business environment, the demand for the integration of such applications is dramatically increased. More than 30 percent of IT dollars are currently being spent to connect different systems (Linthicum 1999).

Under such circumstances, the Enterprise Application Integration (EAI), which is designed to solve the applications chaos for enterprises, has become a very popular research field. What is EAI? David S. Linthicum gives a loose definition for EAI in his book *Enterprise Application Integration*: EAI is not simply a buzzword dreamed up by the press and analysis community. It is, at its foundation, a response to decades of creating distributed monolithic, single-purpose applications leveraging a hodgepodge of platforms and development approaches. EAI is the unrestricted sharing of data and business processes among any connected applications and data source in the enterprise (Linthicum 1999). According to this definition, the purpose of EAI includes:

- Data integration: ensuring multiple applications and systems can share the same data freely
- Progress integration: Sharing the same business processes across applications
- Providing a hodgepodge platform for hardware and operating systems
- Providing a hodgepodge of communication protocols and networking equipment

During the last few decades, many EAI methods and technologies have been developed or are under development. However, most of these methods and technologies are not widely used by the enterprises.

On one hand, from the implementation point of view, there are several reasons as following:

- Different enterprises have different EAI requirements. They may have various kinds of applications, business data, and business process. Therefore, the EAI developers must totally understand the EAI problem for an enterprise before developing an EAI system. For example, to solve the EAI problem of a mechanical designing and manufacturing company, the EAI developers have to understand: What kind of special applications are used in this company? What kind of problems need to be solved through the EAI? What kind of data needs to be shared? What kind of business process must they follow? At the same time, to develop a successful EAI system, the EAI developers need to prepare the essential knowledge related to mechanical design and manufacturing. All these problems make the EAI problems difficult to solve.

- Different applications need different integration methods. Many applications have built-in functionalities for adapting design, and these kind functionalities can be used for application integration, but usually they are difficult. Take AutoCAD as an example, all the recent generations of AutoCAD have built-in adapting design functionalities such as VBA (Visual Basic for Applications), ActiveX, AutoCAD script, and API (Application Programming Interface). Any of this requires the EAI developer to spend a lot time to learn if he or she wants use them as the means of integration. Imaging, if there dozens of different applications to be integrated in a company, how much time does the developer need to spend to acquire the knowledge that is necessary?

- Long developing time. Most EAI methods and technologies promise a short development time. However, the reality is that this promise is very difficult to achieve under the circumstances listed before.

- High cost. Usually, developing an EAI system costs less than replacing and updating all the legacy applications. However, the cost is still too high for many Small and Medium Enterprises.

On the other hand, from the design point of view, most of these EAI methods and technologies are just on answering the questions like "what we need to do?", and seldom consider the implementing process according to the real application and business environment. In other words, most of these methods and technologies stay on the theoretical level. Therefore, the EAI concepts and solutions provided by these methods and technologies usually can not fulfill the real needs.

With the experiences in design methodology research field, we have realized that we can take the advantage of design theories in dealing with generating design concepts, and avoid the mistakes that many EAI developers make. That is, we can take the EAI problem as a design problem, and use a design theory to generate the best EAI concepts before coding. In this way, the problem of finding a solution for an EAI problem becomes a conceptual design process.

Conceptual design is one of the most critical design stages. It is at this stage that some of the most important design decisions are made. Models and methods have been proposed to generate design concepts, based mostly on the systematic design approach (Pahl and Beitz 1988). The conceptual design process is generally divided into the

following steps: identify design specifications, establish function structures, search for solution principles for fulfilling the sub-functions, combine the solution principles to fulfill the overall function, select suitable combinations to define the design concepts, and evaluate the concepts against the technical and economic criteria (Pahl and Beitz 1988).

Environment-Based Design (Zeng 2004b) gives a design model that is derived from the axiomatic theory of design (Zeng 2002). This model differs from other design models in two aspects: 1. the design problem is defined in terms of the product environment rather than the functional structure; 2. the design process is modeled as a process that simultaneously generates design problems and design solutions (Zeng and Cheng 1991). These two characteristics closely match the characteristics of an EAI problem. Therefore, the EBD method should be the best choice for the solution of an EAI problem – a design method combination.

1.2 Objectives

This thesis aims to provide an effective approach to the solution of Enterprise Application Integration (EAI) problems. That is, we will take an EAI problem as a design problem, and use the Environment-Based Design method (Zeng 2004b), which was first introduced by Dr. Yong Zeng, to generate the best design concepts.

In the present thesis, the following sub-objectives will be achieved:

1) Introducing the EBD problem formulation method to EAI problems.

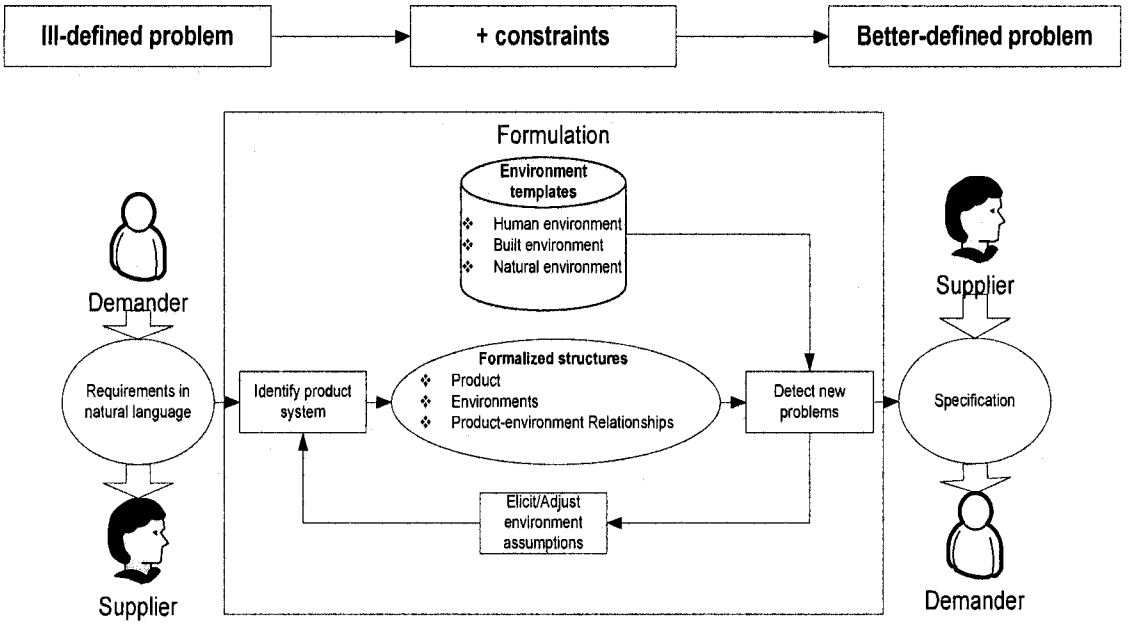


Figure 1-1 Problem formulation of environment-based design (Chen 2006)

As shown in the top row of Figure 1-1 (Chen 2006), design problems are well-known as ill-defined problems. Through the environment analysis process, more constraints will be added to get a better definition for the next design process. For EAI problems, the original requirements (ill-defined problems), or only the situation of the application chaos, are often provided to the supplier (EAI developer) by the demander (user). It is the EAI developer's responsibility to use the environment analysis method to help the user to make clear what the real problem is and also to decide the best solution for the EAI problem.

2) Implementing the Environment-Based Conceptual Design method to generate better EAI concepts.

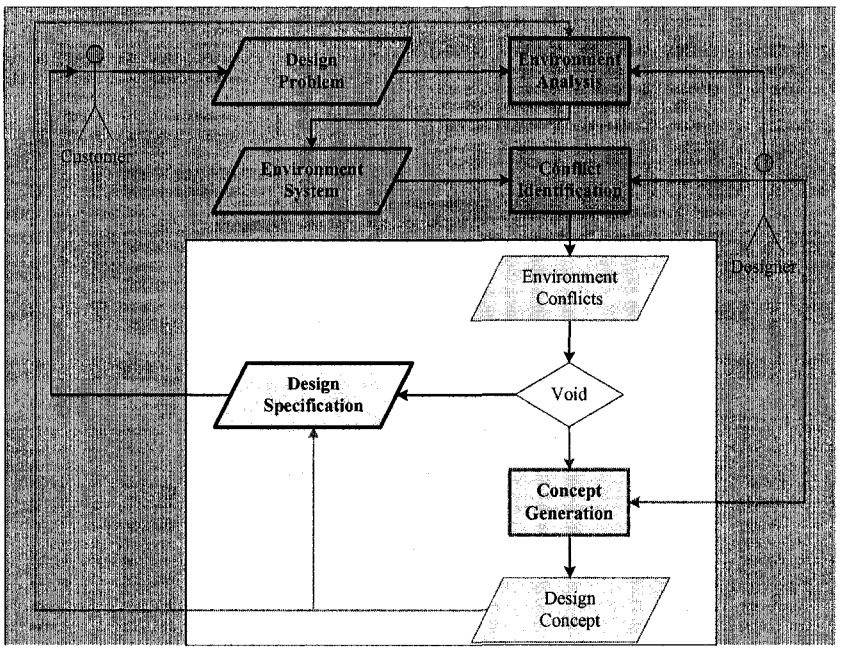


Figure 1-2 Concept generation of EBD (Zeng 2004b)

Figure 1-2 shows the process of how a design concept is generated in the EBD method. Different from other design theories, the process of concept generation in EBD is based on the environment analysis result: conflicts between environments. The design concept is must a solution for the conflicts, and must fulfill the design specifications provided by he users.

3) Building up an EAI problem-solving model based on the EBD method.

This model will provide a generated EBD-EAI pattern for EAI problems. All the important techniques and processes of Environment-Based Design theory, such as environment analysis, relations finding, and conflict identifying, will be used. This model can help the EAI developer to systematically generate better solutions for an EAI problem.

4) Solving a real EAI problem for a company

Any good theory needs practice to prove it. In this thesis, I will use the EBD-EAI method to solve a real EAI problem of the company I am currently working for. This is a metal cutting tool developing, designing and manufacturing company. It has around one hundred employees, and has more than 40 years of history in the cutting, designing, and manufacturing field. Like many other enterprises with a long history, this company had a very urgent demand for the integration of the different applications they were using. In Chapter 5, a detailed process and final solutions are given.

1.3 Challenges

1) Find out what the most difficult part of Enterprise Application Integration is,

One of the challenges facing modern enterprises is to give all their employees complete, transparent, and real-time access to information. Many of the legacy applications still in use today were developed using aged and proprietary technologies. The general idea for EAI is to put all these applications and systems together and to make them freely share all the information including data and processing. The most difficult part for EAI is how to "put all the applications and systems together". To achieve this simple goal of linking applications, an EAI system must provide a discipline and linking methods among them. Although many methods and technologies of EAI have been developed, new and innovative techniques may achieve this goal more easily while providing the enterprise with further competitive advantages. In this thesis, we take advantage of EBD theory, developing a new and

innovative EAI method that is different from most existing EAI methods and technologies.

2) Make it clear what kind of problems other EAI methods and technologies face.

To develop a new EAI method and to ensure that it has unique advantages by comparison with other methods, we must make it clear what kind of problems other EAI methods face. To achieve this, a complete literature review related to both EAI and design theory is a must.

3) Understanding and enriching the Environment-Based Design Theory

The notion of Environment-Based Design was first developed by Dr. Yong Zeng (Zeng 2004a), based on his Axiomatic design theory (Zeng 2001). The environment-based design methodology includes the following three main stages: environment analysis, conflict identification, and concept generation (Zeng 2004b). To understand the advantages of this design theory, a literature review of design theories and related comparisons will be included in this thesis.

4) How can one use the EBD-EAI method to solve real EAI problems?

Practice is always very important to a theory. We can observe and discover the theory through practice, and again through practice to verify and develop the theory. Therefore, a very important part of the present thesis is to use the EBD-EAI method to solve a real EAI problem. Because a design problem is not completely defined until the design solutions are found, many sub-problems and requirements evolve in the process. This process is described in Figure 1-3, where Rd, P, S, and t represent

design requirements, product performance, product description, and the time sequence.

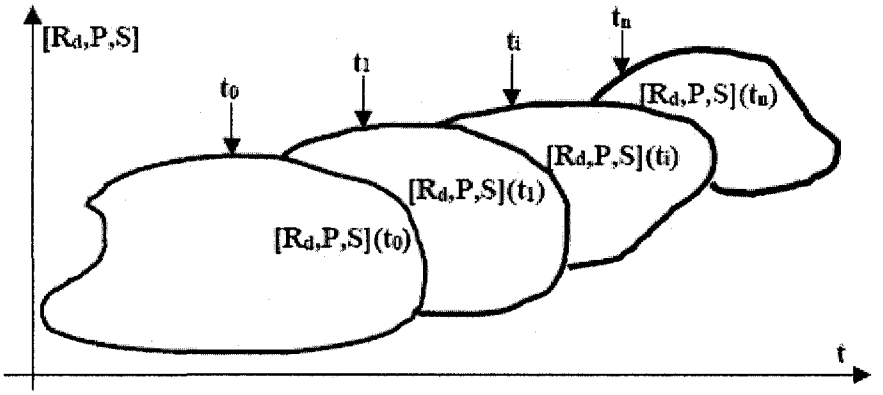


Figure 1-3 A framework of design representation (Zeng and Gu 1999a)

The challenge at this part is to use the EBD method to find the solutions for all the sub-problems evolved during the process.

1.4 Contributions

The objective of this thesis is to provide a new and innovative EAI approach that combines the Environment-Based Design method with Enterprise Application Integration problem solving. That is, we take the EAI problem as a design problem, and use EBD methods to find the best solution. The advantages of the EBD theory ensure that most defective or imperfect solutions are eliminated during the very first stage. The contributions of this thesis can be summarized as follows:

1) Although many EAI methods and technologies have been developed, most of them seldom consider the EAI problem from the design point of view. This leaves these methods and theories only on the purely theoretical level. Since the rapid

development and evolution in the IT (Information Technology) field, these theories that can not be used in practice will eventually be abandoned. The Environment-Based design theory ensures that the EDB approach to EAI problem-solving method can always generate the best EAI solution in terms of the current environment.

2) Environment-Based Design methodology has been proved in many fields such as mechanical design. This thesis extends the range of design theory to a new field: Enterprise Application Integration field.

3) The real EAI problem solving in this thesis provides a very good example of the relationship between theory and practice. Practice is very important for a theory. As Mr. Mao's essay "On Practice" says: Discover the truth through practice, and again through practice verify and develop the truth. Start from perceptual knowledge and actively develop it into rational knowledge; then start from rational knowledge and actively guide revolutionary practice to change both the subjective and the objective world. Practice, knowledge, again practice, and again knowledge. This form repeats itself in endless cycles, and with each cycle the content of practice and knowledge rises to a higher level. Such is the whole of the dialectical-materialist theory of knowledge, and such is the dialectical-materialist theory of the unity of knowing and doing.

1.5 Organization of The Thesis

Chapter 1: Introduction of the thesis. This chapter presents the motivation, objectives, challenges, contributions, and the organization of this paper.

Chapter 2: Literature review. This chapter reviews the literature related to the present thesis research. Since the topic of my thesis is the use of design methods to solve the enterprise application integration (EAI) problem, this review contains two parts: The first part is related to EAI, and the second part is related to design methodology. The objective of this chapter is to clarify the following questions:

- What is EAI?
- Why EAI is so important for modern enterprises?
- What are the strategies for EAI?
- What is design?
- Why it is possible to find an effective approach to EAI by using design methods?

Chapter 3: The review of the Axiomatic Theory of Design Modeling (Zeng 2002), which is the theoretical foundation of this thesis.

Chapter 4: Explanations of the Environment-Based Design (EBD) are given according to my understanding in this chapter. A reusable EBD-EAI problem solving model is given as well.

Chapter 5: Case study. In this chapter, we use the EBD method to solve a real EAI problem based the EBD-EAI problem solving model. As a conceptual design tool, EBD will be used in every process such as the problem formulation, concept generation, etc. At the end of the chapter, the finished EAI application is introduced.

Chapter 6: The conclusion. This chapter is the summary of the thesis project. It contains the following sections:

- The conclusion of this thesis project
- Future work

Chapter 2 Literature Review

2.1 Overview

This chapter contains the review of the literature related to my research. Since the topic of my thesis is the use of design methods to solve the enterprise application integration (EAI) problem, this review includes two parts. The first part is related to EAI, and the second part is related to design methodology.

2.2 Literature Review of Enterprise Application Integration (EAI)

2.2.1 Types of EAI

Before developing an EAI system, the developers have to understand both the business process and data. According to which data and process elements need to be integrated, EAI can take on four approaches (Linthicum 1999), which are illustrated by Figure 2-1:

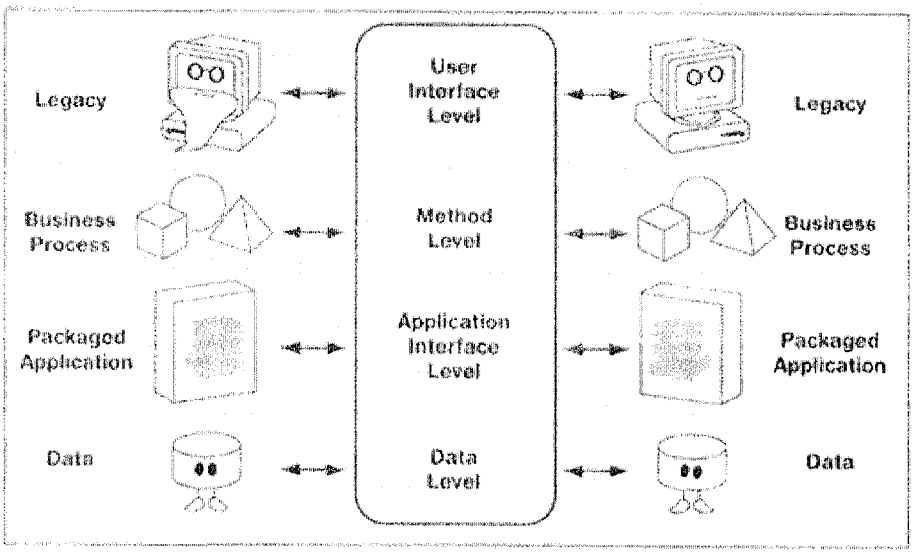


Figure 2-1 Four types of EAI (Linthicum 1999)

- Data level integration: This type of EAI refers to the approach according to which the developers, leaving the applications alone, only work on the database. They extract the data information from one database, make essential processing, and save it in another database. Certainly, sometimes we have to deal with hundreds of databases and thousands of tables. Since we do not need to change the applications, this approach usually has lower cost compared to other approaches (Zhang, Chan and Li 2003).

- Application level integration: This type of EAI refers to the approach of designing a package application with a customized interface. The developer uses this interface to bundle many different applications together and then makes them share the same business data and processes (Reiersgaard, Salvesen and Nordheim 2005). The users can use this interface to access the applications. This type of EAI includes SAP, PeopleSoft, and Baan (Lee, Siau and Hong 2003).

- Method level integration: This type of EAI refers to the approach of sharing the same business logic within the enterprise (Sutherland and Heuvel 2002).

- User interface lever integration: This approach uses the applications' user interfaces to bundle different applications together. In many instances, this approach may be the only solution; some negative issues related to this approach are reliability and performance.

According to the problem space, five types of EAI approaches can be identified (Kalakota and Whinston 1993):

- Individual computing: This kind of EAI approach focuses on integrating different applications in one computer, such as the integration of ACCESS and AutoCAD.

- Intra-Workgroup integration: This kind of EAI approach focuses on integrating a working group, so that all the members in the group can use their different applications to access and freely share the same data and process.

- Intra-Functional integration: focus on the integration of different working systems rather than of different applications. For example, the integration of legacy systems of an enterprise application.

- Inter-Functional integration: This kind of EAI approach focuses on the integration of different functional areas, such as marking, accounting, engineering, etc.

- Inter-enterprise integration: Focus on the integration of different enterprises located in different places to enable data sharing. The Internet or other kinds of networks are involved.

2.2.2 The Process of Enterprise Application Integration

The strategy and implementation of EAI is based on the planning methodology (Salmela, Lederer and Reponen 2000). Usually, the process of enterprise application integration consists of five stages:

- Identify the potential benefits for the enterprise application integration.

- Analysis the enterprise integration capacity

- Select which compatibility improvements are appropriate for the business

- Plan the integration

- Implementation

2.2.3 EAI Implementation Approaches

Since EAI can benefit enterprises in many ways, such as cycle time reductions, cost reductions, and cost containment (Lee, Siau and Hong 2003), many research studies have been carried out on EAI implementation approaches.

Sandia National Laboratories developed an EAI system in the 1990s named Product Realization Environment (PRE) (Whiteside, Friedman-Hilll and Detry 1998). It is a lightweight, CORBA-based framework for the integration of different applications. By using this framework, all the applications to be integrated are "wrapped". PRE contains many reusable components such as a product data management (PDM) system, a human resource database, finite element analysis programs, and some document format converters. Figure 2-2 shows the major architecture of the PRE framework. In this figure, CORBA serves as a bus to provide the data transportation between all the components. Application "wrappers" integrate all applications together, making all integrated applications useable as programming components. User interface developers then use these components to accomplish the tasks desired by the end users.

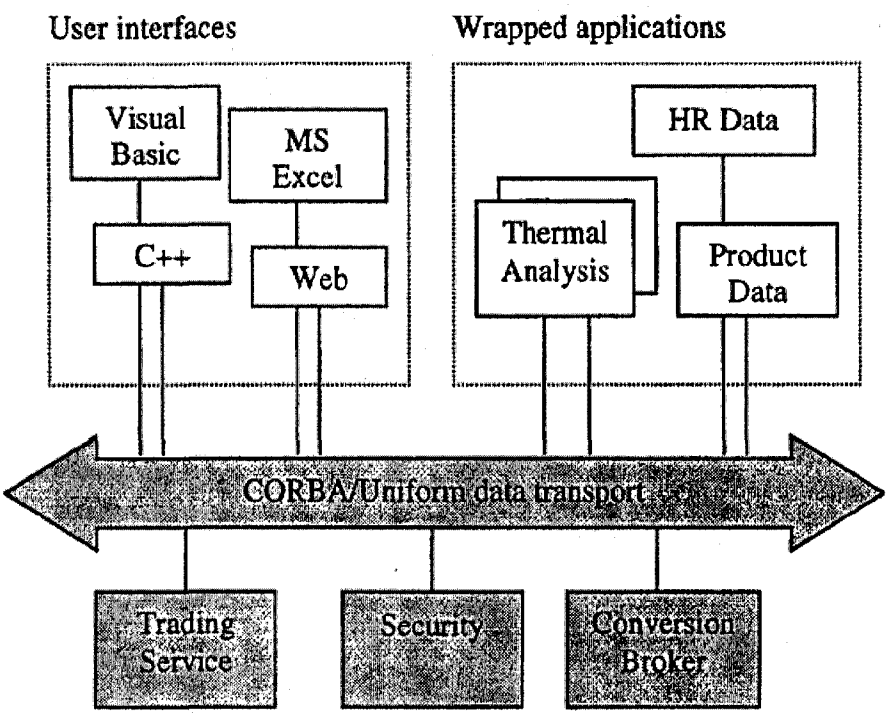
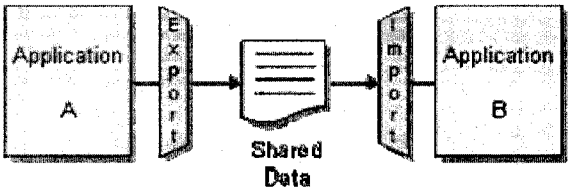


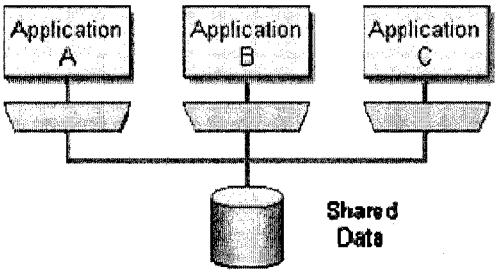
Figure 2-2 Major architecture of PRE

Gregor Hohpe (Hohpe and Woof 2003) form the message (a kind of simple data structure) point of view to find the solutions for enterprise application integration. As everybody knows, EAI is not a simple task; however, experienced EAI application developers always compare the new project with the project they finished before. They use these finished projects as "patterns". That is why the authors believe that the EAI problem-solving patterns can help EAI developers in many different ways. In this book, they generalize 65 patterns for different EAI perspectives, such as integration style, messaging system, messaging channels, message transformation, and system management. Figure 2-3 illustrates some patterns for EAI integration styles provided in the book.

Have each application produce files containing information that other applications need to consume. Integrators take the responsibility of transforming files into different formats. Produce the files at regular intervals according to the nature of the business.



Integrate applications by having them store their data in a single *Shared Database.*



Develop each application as a large-scale object or component with encapsulated data. Provide an interface to allow other applications to interact with the running application.

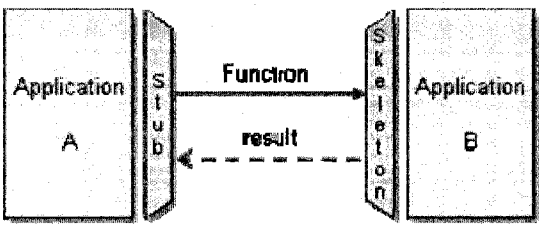


Figure 2-3 Patterns for EAI styles

2.3 Literature Review of Design Theories

2.3.1 Definitions of Design, Design Problem and Design process

Many previous researchers have provided various descriptions of the term "design": design activities are generally considered to be a form of complex problem solving (Simon 1969); design begins with a needs-analysis (Asimow 1962); design is a social activity (Minneman 1991). In most design studies, the objectives usually focus on

finding common characteristics from different engineering domains, within the framework of cognitive science (Goel 1995). Therefore, the design process can be regarded as a cognitive process intended to produce a solution to a design task.

Engineering design has been defined by Fielden (Feilden 1963) as, "the use of scientific principles, technical information and imagination in the definition of the mechanical structure, machine or system to perform prespecified functions with maximum economy and efficiency."

Design problems are well-known as ill-defined and open-ended problems. An ill-defined problem, also called an ill-structured problem as distinguished from a well-structured problem, involves two critical concepts in cognitive science. Different from a well-defined problem, such as a chess game, an ill-defined problem (a) is more complex (b) begins with the inadequate initial conditions (c) presents fewer "end" criteria. An open-ended problem does not have an optimal solution, but only a satisfying one (Simon 1969), which may have several or many correct solutions. When provided with the same list of product requirements, different design teams produce different product solutions.

The design process varies from product to product and from industry to industry. A generic diagram of the activities that must be accomplished for all projects is shown in Figure 2-4 (Ullman 2003). In this framework, any product must go though five phases; three of them are about the design process. They are project definition, specification definition, and conceptual design. Even in the product development phase, there is the probability of refining the design solution or of canceling the entire project.

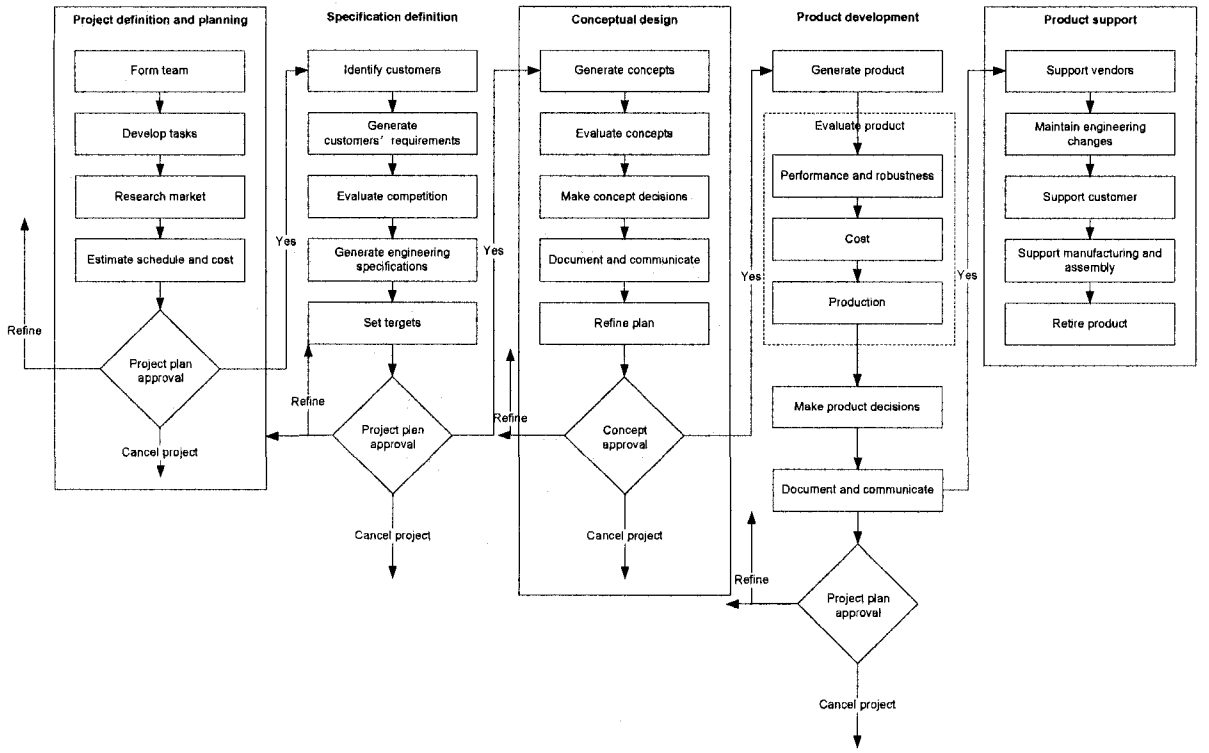


Figure 2-4 Ullman's generic design process in all projects (Ullman 2003)

Pahl and Beitz have done research into the traditional design process. Their model is different from that of Ullman. Theirs is shown in Figure 2-5 (Pahl and Beitz 1988). In this model, a product design process is divided into five phases, which are the following: clarification of the task, concept generation, embodiment design, detail design, and physical evaluation. Obviously, Ullman's model pays more attention to needs analysis; Pahl and Beitz's model emphasizes the generation of solutions and evaluations.

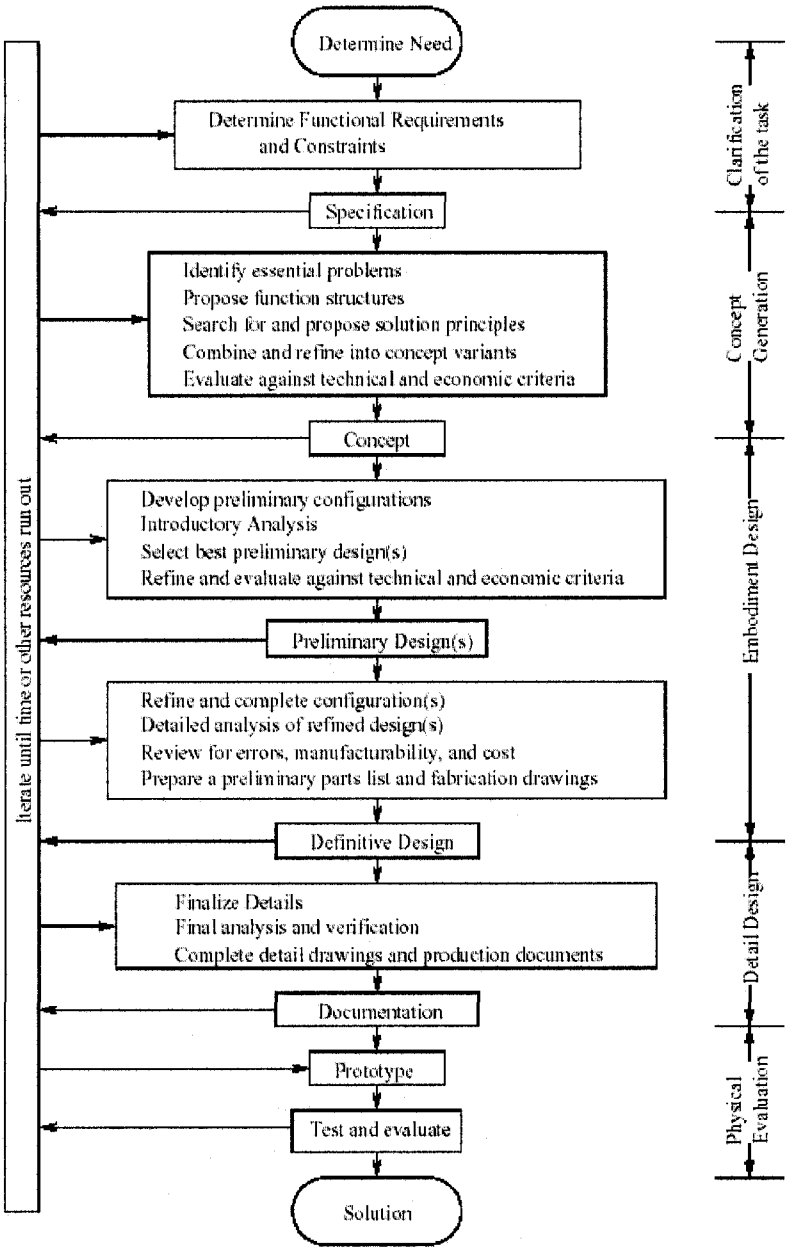


Figure 2-5 Pahl and Beitz's traditional design process (Pahl and Beitz 1988)

About "design model", Gero has conjectured that "perhaps the earliest of the widely accepted models of designing is introduced by Asimow (Asimow 1962), who divided all the designing processes into three typical classes: analysis, synthesis, and

evaluation" (Gero 1998). The relationship among the three classes is illustrated in Figure 2-6.

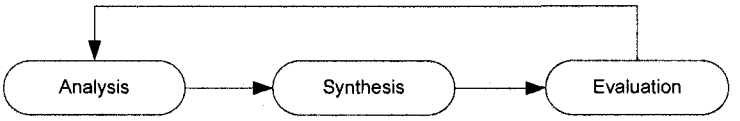


Figure 2-6 Analysis-synthesis-evaluation model (Asimow 1962)

Asimow's model was developed in subsequent research, considering that, "in the beginning of the design session that the designer not only follows evaluation by analysis but for an equal amount of time follows evaluation by synthesis. Already this behavior is different to that "predicted" by Asimov's model. As the design session proceeds, this behavior increasingly diverges from Asimov's model. Thus, in the last 75% of the time of the design session the predominant behavior is not that predicted by Asimov at all since it is: evaluation followed by synthesis." (Gero 1998). The new model of analysis-synthesis-evaluation is shown in Figure 2-7 (Lawson 1997).

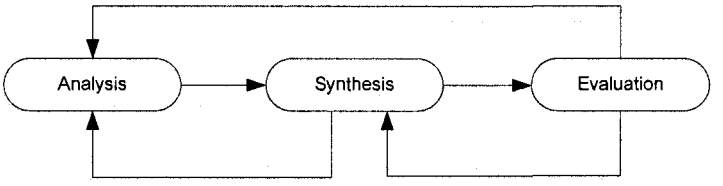


Figure 2-7 New model of analysis-synthesis-evaluation (Lawson 1997)

Two more design process classes are extended from "evaluation"; they are Revision and Implementation (Tovey 1997).

2.3.2 Review of TRIZ

TRIZ is a Russian abbreviation of what can be translated as "the theory on inventive problem solving." It was developed by Genrich Saulovich Altshuller (1926-1998) (Savransky 2000). From the time when it was first introduced in 1946, TRIZ has become one of the most powerful methods for creativity deign (Yang and Zhang 2002).

Semyon D.Savransky gives the definition of TRIZ: TRIZ is a human-oriented knowledge-based systematic methodology of inventive problem solving (Savransky 2000). Where "systematic" has two meanings: First, it means generic and detailed models of artificial systems and processes. Second, it means that the procedures of problem solving are systematically structured to provide effective solutions.

TRIZ methodology includes knowledge-based tools and problem-analysis tools. The theoretical foundations of these tools are the patterns of evolution of technological systems, which are illustrated by Figure 2-8.

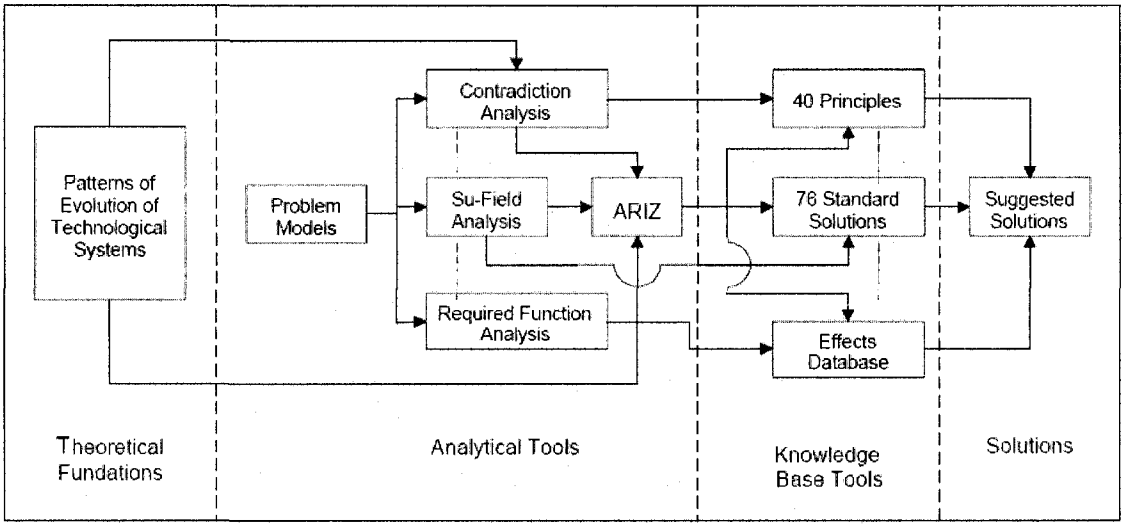


Figure 2-8 Structure of TRIZ (Zhang Jun. 2002)

The ideology base of TRIZ includes two opposite philosophic ideas: Contradiction and Ideality. These two ideas can be included in any inventive problem-solving process. Altshuller distinguished three kinds of contradiction (Altshuller 1986): Administrative contradiction, technical contradictions, and physical contradictions. Therefore, a design problem can be converted to a composition of different contradictions, and the problem-solving process becomes a contradiction-solving process.

TRIZ methodology gives forty inventive principles and seventy-six standard solutions. Each inventive principle and standard solution is a recommendation to help the designer to find a solution eliminating related contradictions.

In conclusion, the TRIZ methodology is a very powerful tool for creative design. However, the foundation for TRIZ is a big collection of patterns related to the design problem. Although we can get some useful ideas from TRIZ methodology, such as the concept of contradiction, we can not use it to effectively solve the EAI problems.

Chapter 3 Review of Axiomatic Theory of Design Modeling

3.1 Overview

The Axiomatic Theory of Design modeling (Zeng 2002) is the theoretical foundation for this thesis. This chapter contains the review of this theory.

3.2 Review of Axiomatic Design Theory

The axiomatic theory of design modeling is a logical tool for representing and reasoning about object structures(Zeng 2002). It provides a formal approach that allows for the development of design theories following logical steps based on mathematical concepts and axioms. The primitive concepts of universe, object, and relation are used in the axiomatic theory of design modeling. The definitions can be found in the Random House Webster's Unabridged Dictionary as follows.

[Definition 1] The universe is the whole body of things and phenomena observed or postulated.

[Definition 2] An object is anything that can be observed or postulated in the universe.

It can be seen from the two definitions above that universe is the whole body of objects.

[Definition 3] A relation is an aspect or quality that connects two or more objects as being or belonging or working together or as being of the same kind. A relation can also be a property that holds between an ordered pair of objects.

$$R = A \sim B, \exists A, B, R, \tag{3-1}$$

where A and B are objects. A~B is read as "A relates to B". R is the relation from object A to object B. The basic properties of relations include idempotent, commutative, transitive, associative and distributive.

Based on these concepts, two axioms are defined in the axiomatic theory of design modeling.

[Axiom1] Everything in the universe is an object.

[Axiom 2] There are relations between objects.

It can be seen from these two axioms that the characteristics of relations would play a critical role in the axiomatic theory of design modeling. We need to define a group of basic relations to capture the nature of object representation. Two corollaries of the axiomatic theory of design modeling can be used to represent various relations in the universe.

[Corollary 1] Every object in the universe includes other objects. Symbolically,

$$A \supseteq B, \forall A \, \exists B \tag{3-2}$$

B is called a subobject of A. The symbol $\supseteq$ is inclusion relation. The inclusion relation is transitive and idempotent but not commutative.

[Corollary 2] Every object in the universe interacts with other objects. Symbolically,

$$C = A \otimes B, \forall A, B \, \exists C \tag{3-3}$$

where C is called the interaction of A on B. The symbol $\otimes$ represents interaction relation. Interaction relation is idempotent but not transitive or associative. Based on the above Corollary 1 and 2, a new operation can be developed as follows:

[Definition 4] Structure operation, denoted by $\oplus$, is defined by the union of an object and the relation of the object to itself.

$$\oplus O = O \cup (O \otimes O) \tag{3-4}$$

where $\oplus O$ is the structure of object O.

The structure operation provides the aggregation mechanism for representing the object evolution in the design process.

This chapter presents the basis for representing product requirements derived by using the axiomatic theory of design modeling. This base is the foundation for the formal study of product requirements. It forms a coordinate system for representing various requirements. Especially, results from this formal study will be useful for developing computer software systems that support the gathering and specification of product requirements.

In order to identify the base for representing customer requirements, some definitions are given as follows:

[Definition 5] A product system is the structure of an object (Ω) including both product (S) and its environment (E).

$$\Omega = E \cup S \tag{3-5}$$

According to (3-5), the product system ($\oplus\Omega$) can be represented as

$$\oplus\Omega = \oplus(E \cup S) = (\oplus E) \cup (\oplus S) \cup B \tag{3-6}$$

where B is the product-environment boundary defined as follows:

[Definition 6] The product environment boundary, denoted by B, is the collection of interactions between a product and its environment.

$$B = (E \otimes S) \cup (S \otimes E) \tag{3-7}$$

Graphically, a product system can be represented in Figure 3-1.

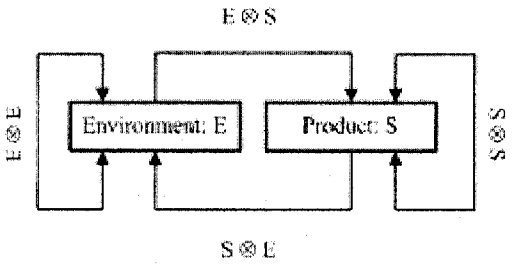


Figure 3-1 Product system

The product environment boundary includes the structural boundary and the physical interactions. The structural boundary (B^s) is the shared physical structure between the product and its environment. The physical interactions include the actions (B^a) of the environment on the product and the responses (B^r) of the product to the environment. Therefore, the product environment boundary can be further represented as

$$B = B^s \cup B^a \cup B^r, \tag{3-8}$$

where $\cup$ denotes logical "and".

Based on the definition of the product system, a design problem can be formally defined as follows.

[Definition 7] A design problem can be literally defined as a request to design something that meets a set of descriptions of the request. Based on the axiomatic theory of design modeling, both "something" and "descriptions of the request" can be seen as objects and can be further seen as product systems in the context of formulating design problems. Thus a design problem, denoted by P^d, can be formally represented as,

$$P^d = \lambda(\oplus\Omega_0, \oplus\Omega_s), \tag{3-9}$$

where $\oplus\Omega_0$ $(\Omega_0 = E_0 \cup S_0, E_0 \cap S_0 = \Phi)$ can be seen as the descriptions of a request for the design, $\oplus\Omega_s$ $(\Omega_s = E_s \cup S_s, E_s \cap S_s = \Phi)$ is something to be designed, and λ is the "inclusion" relation ($\supseteq$) implying that $\oplus\Omega_s$ will be a part of $\oplus\Omega_0$ so that the designed product will meet the descriptions of the design. Obviously, if $\oplus\Omega_s$ is a part of $\oplus\Omega_0$, then equation (3-9) is satisfied. At the beginning of the design process, $\oplus\Omega_s$ is an unknown and $\oplus\Omega_0$ is the only thing defined. The true value of P^d is undetermined, which means the request is yet to be met.

According to (3-6) and (3-7), we have

$$\begin{aligned} \oplus\Omega_0 &= (\oplus E_0) \cup (\oplus S_0) \cup B_0, \\ \oplus\Omega_s &= (\oplus E_s) \cup (\oplus S_s) \cup B_s. \end{aligned} \tag{3-10}$$

Since $E_i \cap S_j = \Phi, \forall i, j = 0, s$, we have

$$P^d = \lambda(\oplus E_0, \oplus E_s) \wedge \lambda(\oplus S_0, \oplus S_s) \wedge \lambda(\oplus B_0, \oplus B_s), \tag{3-11}$$

Where the symbol $\wedge$ denotes logical "and".

Substitute (3-8) into (3-11), we have

$$P^d = \lambda(\oplus E_0, \oplus E_s) \wedge \lambda(\oplus S_0, \oplus S_s) \wedge \lambda(B_0^s, B_s^s) \wedge \lambda(B_0^a, B_s^a) \wedge \lambda(B_0^r, B_s^r). \tag{3-12}$$

Equation (3-12) indeed can be organized into three parts:

- $\lambda(\oplus E_0, \oplus E_s)$ corresponds to requirements on product environment.
- $\lambda(\oplus S_0, \oplus S_s) \wedge \lambda(B_0^s, B_s^s)$ defines direct constraints on products.
- $\lambda(B_0^a, B_s^a) \wedge \lambda(B_0^r, B_s^r)$ defines direct constraints on actions and/or responses.

Theorem of Design Problem Structure. A *design problem* is implied in a product system and composed of three parts: the environment in which the designed product is expected to work, the requirements on product structure, and the requirements on performances of the designed product.

This theorem can be shown in Table 3-1.

Table 3-1 Elements of design problem

Design Problem: P^d	
Product Environment	E
Performance Requirements	$\lambda(B_0^a, B_s^a) \wedge \lambda(B_0^r, B_s^r)$
Structural Requirements	$\lambda(\oplus S_0, \oplus S_s) \wedge \lambda(B_0^s, B_s^s)$

Chapter 4 Environment-Based Design and EBD-EAI Problem Solving Model

4.1 Overview

Explanations of the Environment-Based Design (EBD) are given according to my understanding in this chapter. A reusable EBD-EAI problem solving model is given as well.

4.2 Understanding of Environment-Based Design Theory

The notion of Environment-Based Design was first proposed by Dr. Yong Zeng (Zeng 2004a), based on his axiomatic theory of design modeling. Different from traditional design methodologies, which are largely based on the understanding that a generic design process comprises analysis, synthesis, and evaluation, the environment-based design methodology includes the following three main stages: environment analysis, conflict identification, and concept generation. These three stages work together progressively and simultaneously to generate and refine the design requirements and design solutions. The EBD methodology is illustrated in Figure 4-1. The objective of environment analysis is to find out the key environment components, in which the product works, and the relationships between the environment components. From the environment implied in the design problem described by the customer(s), the designer will introduce extra environment components that are relevant to the design problem at hand. The results from this analysis constitute an environment system. Following the environment analysis, conflicts should be identified among the relations between environment components. At the third stage of EBD, a set of key environment conflicts will be chosen to be resolved by generating some design concepts, which will

be new environment components for the proceeding design. This process continues until no more unacceptable environment conflicts exist.

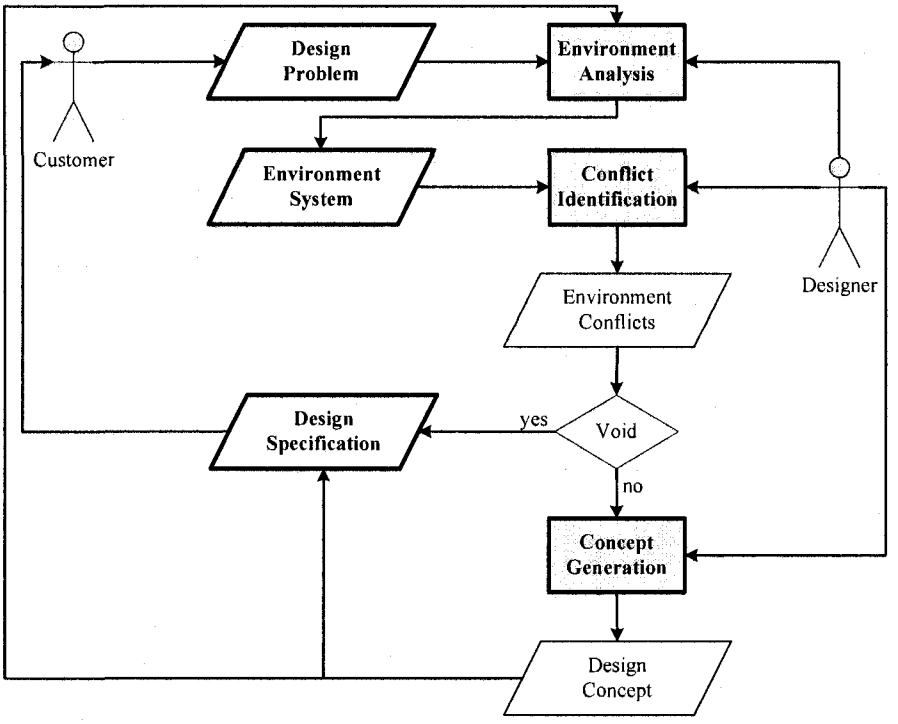


Figure 4-1 Environment-Based Design: general procedures.

4.2.1 Fundamental Concepts of EBD

There are several important fundamental concepts of Environment-Based Design theory that are discussed in the present section, such as design problem, environment, conflicts, solutions, etc.

1) **Understanding of the Product System**:

A product system is composed of the product, the environment of this product, and the relationships between them. A product system is illustrated in Figure 3-1.

The definition of the product system perfectly describes how a product exists in the universe. We can understand the product system notion through following steps:

- First, we get an entire picture of the universe, which is illustrated in Figure 4-2. Like the definition of "universe" defined in the Axiomatic design theory, the universe is the whole body of things and phenomena observed or postulated; the universe contains all the existing objects (discovered and undiscovered).

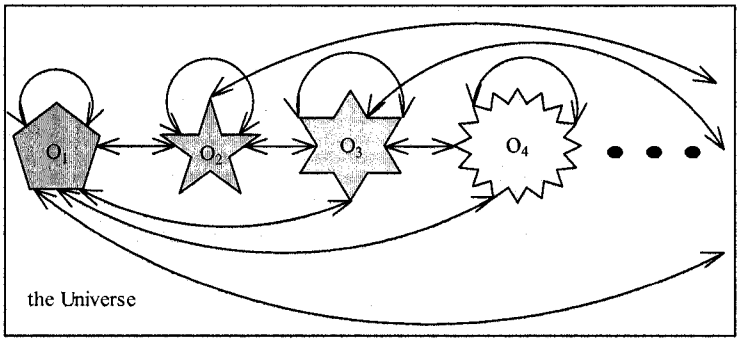


Figure 4-2 The Universe

- Second, we take Object 3 in Figure 4-2 as an example. If we move Object 3 out of the universe, Figure 4-2 can be changed to Figure 4-3.

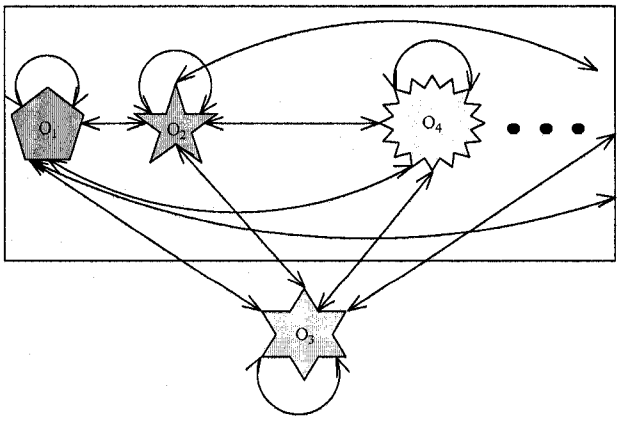


Figure 4-3 Separate object 3 with the universe

- Finally, Figure 4-3 can be simplified to Figure 4-4, which is in fact the product system denoted by Figure 3-1.

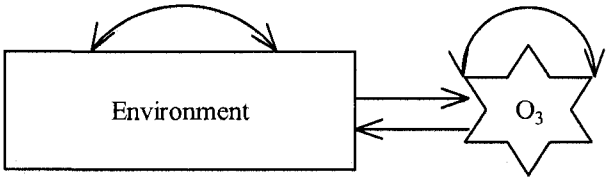


Figure 4-4 Object 3 and its environment (product system)

A product system can be denoted as:

$$\oplus \Omega = \oplus E \cup \oplus O \cup (E \otimes O) \cup (O \otimes E) \tag{4-1}$$

where $\oplus \Omega$ is the structure of the product system, and $\oplus E$ is the structure of the environment, which can be denoted by:

$$\oplus E = E \cup (E \otimes E) \tag{4-2}$$

2) Basic concepts

Based on this understanding of a product system, a set of important fundamental concepts related to the Environment-Based Design theory will be defined as follows:

A. **Environment**:

According to Dr. Yong Zeng's theory, the environment of a product is everything except the product itself (Zeng 2002).

From Figure 4-4, we can get:

- The environment of an object is the universe without this object.
- An object and its environment compose a new universe.

$$\oplus E \cup \oplus O \cup (E \otimes O) = U \qquad (4\text{-}3)$$

Relate these concepts to a product design problem and we can get

- The current universe is the environment of a new product to be designed and manufactured.

B. Engineering design:

Based on the understanding of the universe and an object's environment above, we can define the engineering design as follows:

- Design is a process of creating something new (does not exist before) which can change the universe to a desired status.
- Design is a kind of human action to modify and change the world.

These definitions clarify the notion that design is a very important human action to change the world. There is no doubt that any designed products and inventions have some impact upon the world we live in. The only difference is the impact scope and the impact degree. Some inventions (results of a design with much creativity) can dramatically change the world, while others just lead to slight changes.

C. Design problem:

"A design problem can be literally defined as a request to design something that meets a set of descriptions of the request (Zeng and Gu 1999a)." This definition clearly defines the goal of a design problem. In this definition, "something" is the design product; "a set of descriptions of the request" refers to the design requirements.

Corresponding to the definitions we made before, we refine the definition of a design problem as:

- A design problem is a problem about how to change the existing universe to a desired status.

According to this definition, the core of a design problem is the desired status. The base of a design problem is the existing universe, in other words, the environment that the designed product would exist in.

D. Design product

A design product is a blueprint generated from a design process. If an object or objects are made according to a design and have been put into the universe, the universe will be changed to a desired status predefined in the design problem.

E. Conflicts

One of the most important stages for the EBD process is the stage of identifying the key conflicts between environment components. So, we must make it clear what a conflict is. The WEBSTER dictionary gives a definition of conflict: competitive or opposing actions of incompatibles: antagonistic state or action (as of divergent ideas, interests, or persons). A conflict is composed of three basic elements: two competing objects and one resource object that the former two objects contend for. Clyde H. Coombs and George S. Avrunin classified conflicts as being of three types: The first type is conflict within an individual because he or she has to choose between options, each of which may be better than any other in some respect. The second type is conflicts between individuals because they want different things but must settle for the same thing. The last

type is conflicts between individuals because they want the same things but must settle for different things (Coombs and Avrunin 1988).

From the design point of view, conflicts are directly related to the design problem. A design concept is a composition of the conflict solutions. Therefore, it very important to identify all the important conflicts between key environment components of a design problem in the very first stage: Environment Analysis.

4.2.2 Environment Analysis

The key objective of Environment Analysis is to find out all the key environment components for a design problem and the relationships between the environment components. From the environment implied in the design problem described by the customer(s), the designer will introduce extra environment components that are relevant to the design problem at hand. The results from this analysis constitute an environment system. An environment decomposition method has been developed (Zeng and Chen 2005).

4.2.3 Environment Decomposition

The environment of a product can be decomposed to sub-environments through different methods. For example, according to the properties of the environment, the environment is the aggregation of the natural environment, the built environment, and the human environment (Zeng 2004a). According to the relative importance of the product, the environment can be further sub-divided into the close environment and the remote environment. Since it is impossible to list all the environments of a product before we decompose and analyze the environment, the first step to decompose the environment will be done according to the relative importance. In this way, we can easily eliminate

some relatively useless environments. For instance, when the product is a desk, the air is not important to the desk, so the air should be part of the remote environment of the desk, and it can be ignored. After this step, we can decompose the rest of the close environment to nature, built and human environment, and further to nature, technology, manufacture, transportation, assembly and market environment, etc.

$$E = \bigcup_{i=1}^{n} E_i, \exists E_1, E_2, \cdots, E_n, \forall i, j, i \neq j, E_i \cap E_j = \Phi,$$
$$E^a = \bigcup_{i=1}^{a_e} e_i^a, \tag{4-4}$$

where e_i^a is a part of primitive environment E^a (Zeng and Gu 1999).

The structure of the environment of a product can be illustrated in Figure 4-5.

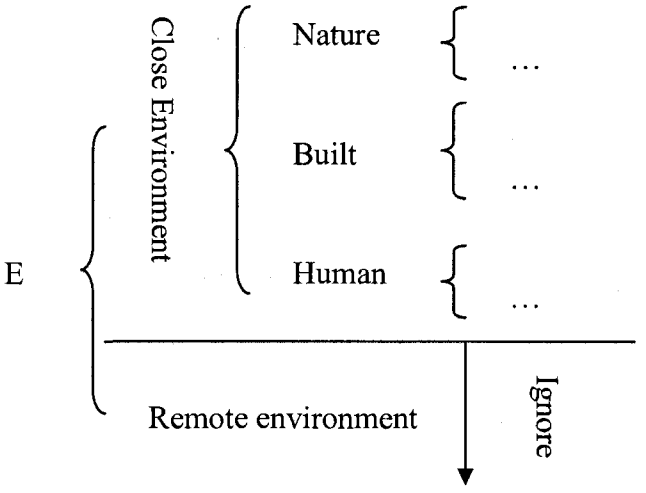


Figure 4-5 The structure of environment

4.2.4 Environment-Based Concept Generation

A concept is an idea that is sufficiently developed so that one can evaluate the physical principles that govern its behavior. It is a primary goal for the design concepts to meet

the targets. Concepts must also be refined enough to evaluate the technologies and to realize them (Ullman 2003).

According to the Environment-Based Design theory, the concept-generation process is the process of transforming forms $\oplus \Omega_0$ to $\oplus \Omega_s$ (Zeng 2004a).

$$S_{i+1} = S_i \cup s_{i+1}^a \tag{4.5}$$

This conceptual generation process is also illustrated in Figure 4-6.

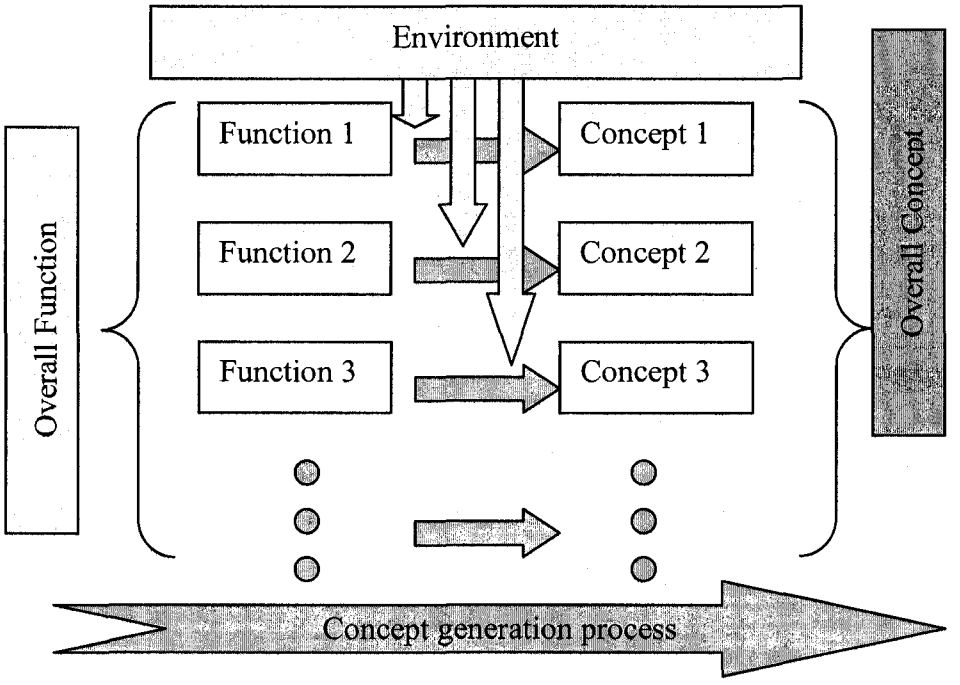


Figure 4-6 Conceptual generation process

4.3 EBD-EAI Problem Solving Model

4.3.1 The EBD-EAI problem solving model

To solve the EAI problem effectively, an EBD-EAI problem solving model is an essential. Considering the characteristic of EAI problems and the process of EBD method, we built up a reusable EBD-EAI problem solving model, with is illustrated by Figure 4-7.

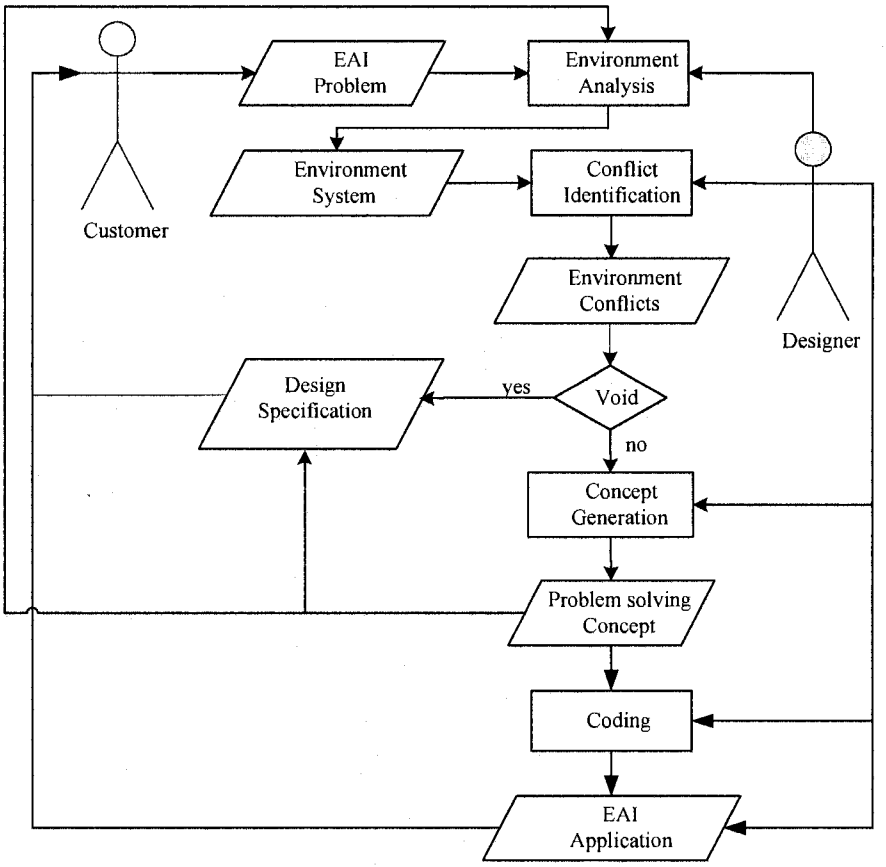


Figure 4-7 The EBD-EAI problem solving model

4.3.2 The regular pattern of the environment analysis process

A regular pattern of the environment analysis process for the EBD-EAI model is also given as Figure 4-8:

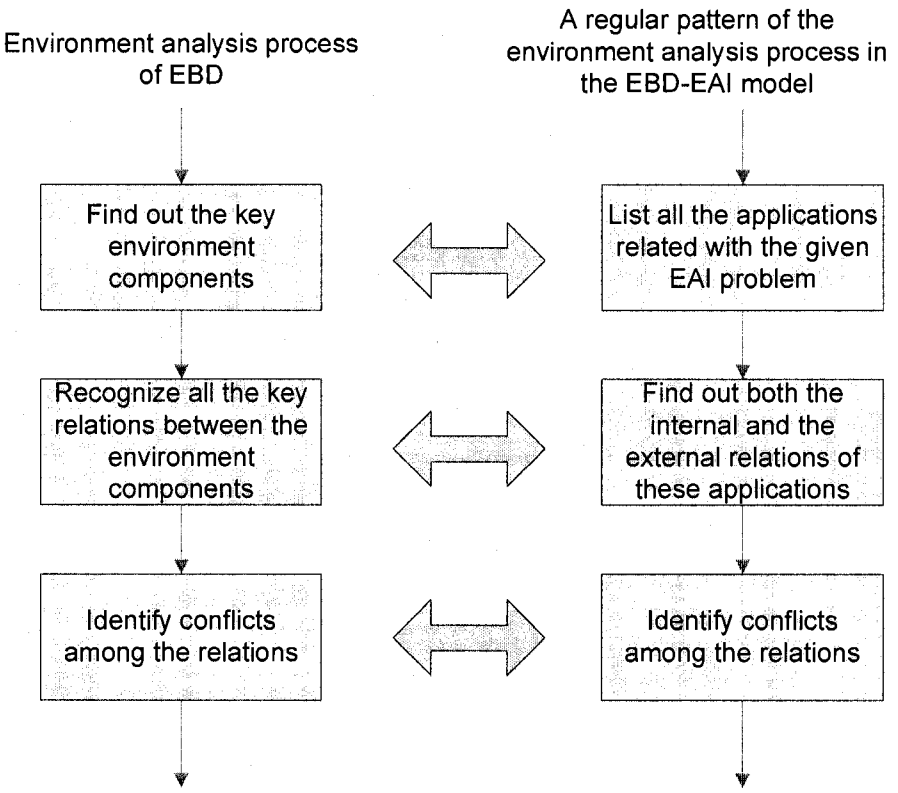


Figure 4-8 The regular pattern of the EBD-EAI environment analysis process

According to this regular pattern, the environment analysis process in the EBD-EAI problem solving model contains three main steps:

- List all the applications related with the given EAI problem.

Since the main goal the EAI is to solve the chaos between different enterprise applications, one of the most important kinds of environment components is the applications. Therefore, in this stage, all the applications related with the given EAI problem need to be listed. Within this stage, other key environment components such as the computers and the users also need to be listed.

- Find out both the internal and the external relations of these applications.

The internal relations of an application include the pros and cons of the application. By clarifying these relations, most pros will be kept and most cons will be eliminated by the final EAI solutions. On the other hand, a clear mapping of the internal relations of an application can help the designers to find the best integration way of the application.

The external relations of applications reflect how these applications working together and what is the bottleneck of their cooperation.

- Identify the conflicts among the relations.

A set of conflicts will be chosen and to be resolved by the concepts generated during the later stages.

Chapter 5 Case Study – EAI Problem Solving

5.1 Introduction

5.1.1 The Enterprise Application Integration Problem

During the last two years, I have been working in a company. This company is a metal-cutting tool developing, designing and manufacturing company. It has around one hundred employees, and was founded more than 40 years ago. This company's products include milling tools, drills, and special cutting tools for the aerospace, automotive and metalworking industries.

In the past two decades, the company has constantly introduced new technologies to improve its management, product development and designing, and manufacturing level. For example, they introduced an ERP (Enterprise Resource Planning) System, MAPICS, 10 years ago, which makes the company among the few companies who first began using ERP as a powerful enterprise manage assistance tool in North America. They were almost the first to start using the first version of AutoCAD to design and make engineering drawings. Moreover, they also introduced many other applications for different purposes. However, because of the short life cycles of computer applications, most of these applications have various problems now. Most importantly, they can not share the business data and process freely throughout the company. The demand for the integration of these applications is increasing dramatically day by day.

5.1.2 Objectives

According to the results of the research on Environment-based design (EBD) methodology, we believe that, as a powerful and effective design problem analysis and solving tool, the EBD–EAI combination can be an effective and creative approach to solving Enterprise Application Integration problems. The working experience in that company has given me an excellent opportunity to test this method. In this chapter, the whole implementation process will be described.

Since the EBD-EAI problem-solving process is a typical design process, it has all the characteristics that a design process has: Many new sub-problems will continue to arise and be solved. To clearly show this complex process, a pattern language will be used as a tool to organize the rest of this chapter.

5.1.3 Pattern Language and Organization of This Chapter

5.1.3.1 What is Pattern Language

A Pattern Language is a special documentation form. It can be used to describe a design problem and also to successfully, effectively, and clearly document a concept and solutions for a design problem. The concept of Pattern Language was first pointed out by Christopher Alexander in his book *A Pattern Language* (Alexander, Ishikawa, Silverstein, Jacobson, Fiksdahl-King and Angel 1977). He used it to classify common civil and architectural design problems as well as solutions. In this chapter, a kind of pattern language will be used to organize all the problems related to the EDP-EAI problem-solving process, as well as all the solutions.

5.1.3.2 Pattern Structure for This Chapter

A pattern language must have a special structure to illustrate particular kinds of problems.

In this chapter, the following structure will be used:

1. **Name**

   The name is the identifier of a pattern. From the name, we can know what the pattern does, and what kind of problem belongs to. Short sentences or phrases will be used as pattern names.

2. **Problem**

   This field is to clarify and explain the details of the problem: What is the problem of the present pattern? How is the problem related to the parent patterns?

3. **Environment Analysis**

   Environment analysis is a very important part of EBD methodology. It involves every stage of the EBD process. During the problem formulation stage, a complete environment analysis can help the designer to have a better understanding of the design problem. During the conflict identification stage, the environment analysis can help the designer to find out all the key environment components, relations (the internal relations of each environment component and the relations between each environment component), and corresponding conflicts. During the concept generation stage, the environment

analysis can check whether the concepts meet the functional goal. Also, the environment analysis can give a way to evaluate the technologies needed to realize the design concepts.

4. Conflicts and Solutions

Through the Environment analysis process, all important conflicts between the key environment components will be identified. This section is actually a summary of the results from the preliminary design problem environment analysis stage. In this section, we will answer the following questions: What are the key conflicts? How do they affect the whole design problem? What kinds of sub-problems are related to these conflicts? What is the difficulty in solving these problems?

If the conflicts are easy to solve, we will give the solutions or answers in this section. Otherwise, a new pattern or patterns will be needed.

5. Figures and tables

Figures and tables are always powerful. They can give a clear illustration of an idea, concept, or problem. They can also demonstrate a design solution in a vivid way. Figures and tables will be used to clarify both problems and solutions.

6. Conclusion

In this section, a short conclusion to this pattern will be given. It includes: the summary of the problem; the results of environment analysis; the key conflicts and their solutions. The answers to the following questions will be given as

well: How does the conflict solution solve the related problems? How to apply the solutions? What kind of new challenges are related to this pattern? What kind of children patterns will be following?

5.1.3.3 Diagram Notation for the patterns

Figure 5-1 illustrates the diagram notations used in the patterns.

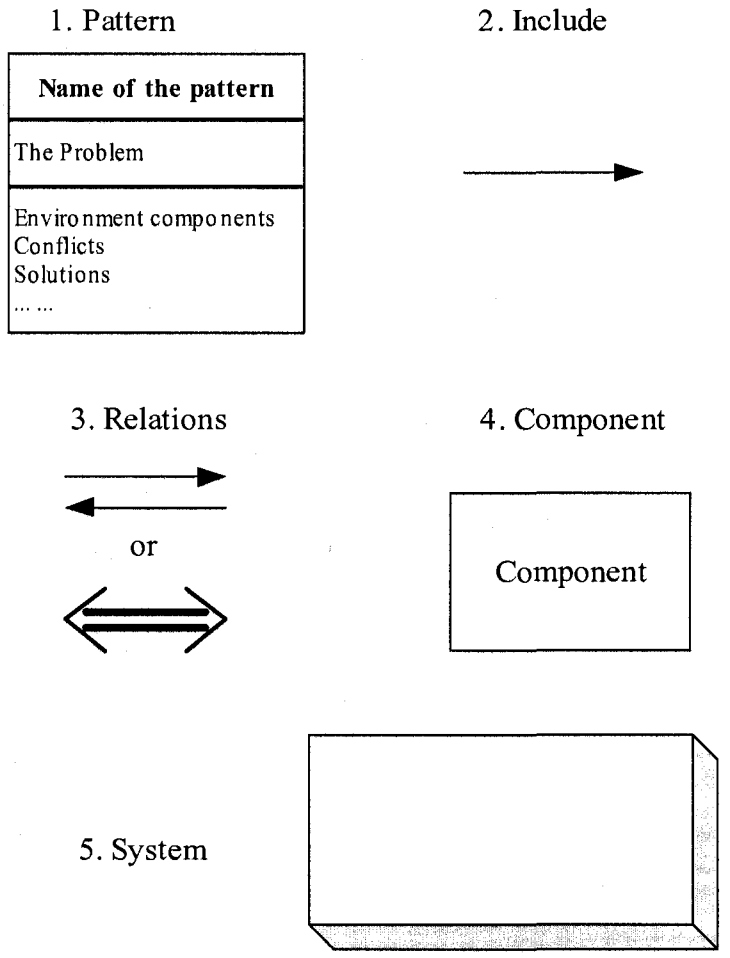


Figure 5-1 The diagram notations

5.1.4 Organization for the Rest of This Chapter

Figure 5-2 illustrates the organization of the rest of this chapter. A problem-solving network is composed of parent patterns and its children patterns. Each pattern contains a particular problem that needs to be solved. Within a pattern, the EBD method will be used to analyze the environment of the problem, and to identify the conflicts. This thesis will also introduce how the EBD method helps finding the best solutions for each problem. After all the problems have been solved, the finished EAI (Enterprise Application Integration) application will be briefly introduced. At the end of this chapter, a conclusion will be given.

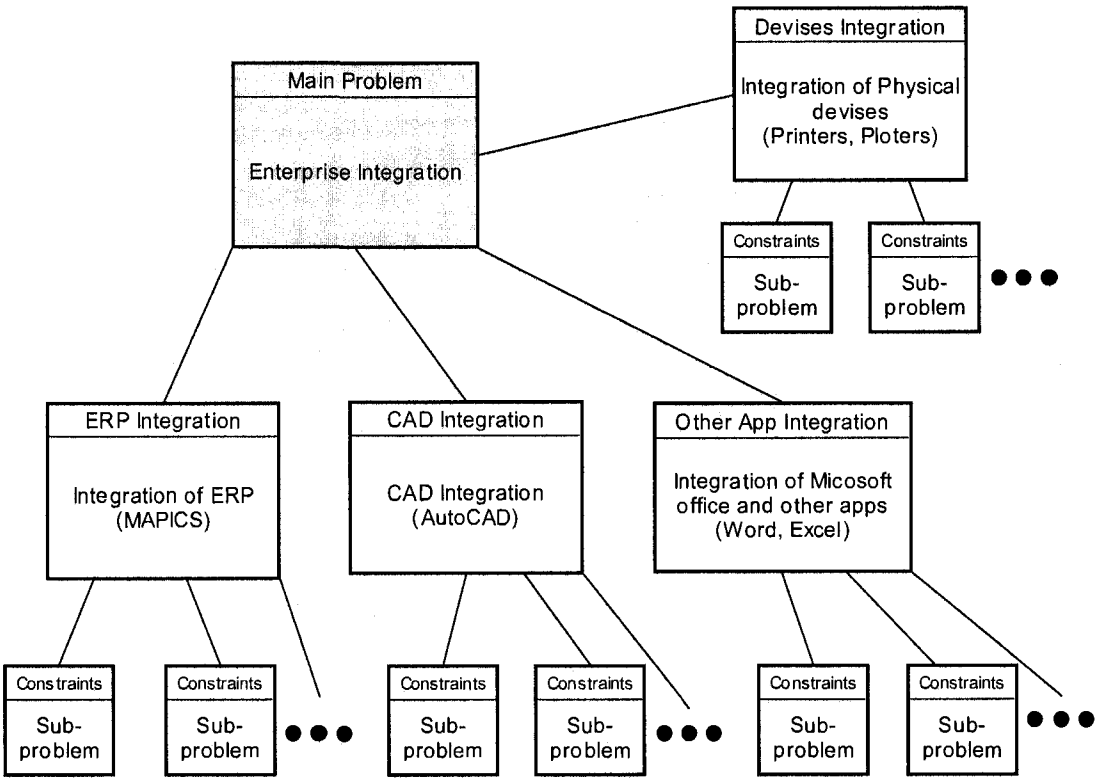


Figure 5-2 Organization for the rest of this chapter

- **Introducing to the integration system**

5.2 The Parent Pattern (pattern No. 1): Application Chaos

5.2.1 Name

Finding the way to solve the application chaos in the company

5.2.2 Problem

According to the situation of this company, we will only focus on three applications and one kind of physical device: printers. The three applications are the following:

1) Enterprise Sources Planning (ERP) system. The ERP system used in this company is MAPICS, which was developed by IBM in the 1990's.

2) CAD Software. The company is current using Pro-Engineer, Solidworks, and AutoCAD. We will focus on AutoCAD since it is the most important CAD application in the company.

3) Microsoft Office.

The purpose of this pattern is to refine the problem. It is a process of changing an ill-defined problem to a better-defined problem. Within this pattern, EBD method will be used to find out what the problems related to these applications and devices are, and what the best solution for these problems is. The selected solution will serve as the input for the following patterns, which contain the rest of the design process.

5.2.3 Environment Analysis of the Problem

5.2.3.1 Find Out Important Environment Components

The key environment components of this pattern are illustrated in Figure 5-3

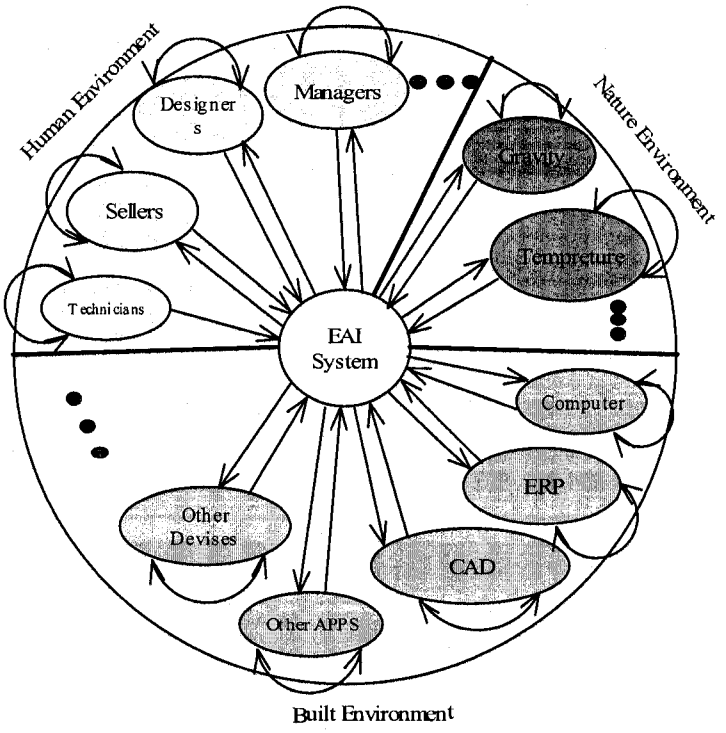


Figure 5-3 The key environment components

5.2.3.2 Identify the Internal Relations of Key Environment Components

1) ERP system: MAPICS

MAPICS is an ERP business solution system developed by IBM in the 1990's, and it is one of the most successful ERP systems in the world. The company introduced this system in 1997. In 10 years, it has been proved to be a good ERP system with great stability, reliability, performance and cost savings. However, since it was developed and introduced 10 years ago, there are many problems with this system today.

Pros:

- The system is very stable.

The server of MAPICS is running on an IBM server with a UNIX System, and it is working almost 24/7 (the system is automatically restarted and maintained only on Sunday mornings). In 10 years, it has almost never stopped working.

- The system is powerful.

  The system includes Inventory Management, Product Data Management, Production Monitor, Custom Orders Management, Accounting Management, Sales Analysis and other modules. So, almost the entire company can be managed by this system.

- The system is widely used.

  In 10 years, the ERP system has been adapted and combined with the company at a very high level. It acts as the heart of the company. The ERP system is so important that if the system stopped working, the company would stop running. In this company, every employee must know how to use the modules in the system related to his or her own work. As managers, they must know how to operate all the modules to check the current company status in every aspect, and to make the right management decisions. As engineers, they must know how to use PDM, COM and other modules related to product design and manufacturing. As salesmen, they must know the IM, COM, PM modules to help them to know what kind of products the customer has ordered, what kind of products there are in the inventory, and what kind of products there are in the production. As works, they must know how to report their daily work through the PM module, etc.

Cons:

- Difficult to operate

  Since the ERP system was developed more than 10 years ago, all the interfaces look like the DOS command widow, and all the operations are based on commands. That is, if you want the system to do something for you, you have to know the corresponding commands first. So, if you use this system a lot, you have to remember many commands. For example, if you want to check the inventory records of a product, you have to follow these steps:

  -> F24 (Force stop the previous job, and return to the main menu)

  -> IM (go to the Inventory Management module)

  -> 5 (Inventory Management)

  -> 1 (Item Availability)

  -> Type the Item-Master of the product and return

  -> Wait until the information page appears

- Difficult to manage the data

  All the product-related data in the system has to be input, deleted, or modified manually. For example, if you want to create a new ITEMMASTER of a new product, you have to go though the following steps:

  -> F24 (Force stop the previous job, and return to the main menu)

-> IM (go to the Inventory Management module)

-> 7 (Item-Master Management)

-> 1 (Create a new Item-Master)

-> Open at least 5 different data entry windows to input the new Item-Master, description, lot size, order number, costing and other information.

-> F7 (Finish data entering)

-> 2 (Go to warehouse management page and relate the new Item-Master to the warehouse).

-> PDM (Go to the Product Data Management module)

-> 2 & F4 (Edit or add material to the new item)

-> F7 (get out)

-> 4 & F4 (Entering all the manufacturing steps and operations)

.....

It must be remembered that the company receives orders with new products everyday, and the program user needs to perform all these steps for each new item. The amount of time spent on all of this repetitive work is considerable. The psychological impact is also important. In fact, it is quite normal for the company to get orders with more than 20 totally new items in a single day!

- Easy to make errors

  Usually, there are two kinds of errors made in the use of the ERP system. The first kind of errors is human-related errors. Since all the data has to be input manually, it is very easy to make mistakes. If an error is found, all of the steps listed before must be repeated and the correction must be made. The second kind of errors is related to operation mistakes such as conflicts between jobs submitted by different users. For instance, if you open PDM to edit the product information of an item while another employee is working with the same module, a conflicting error will occur. All the submitted jobs will be stacked there, and the whole module will be halted until someone notices the error and has the error fixed.

- Difficult to perform trouble shooting

  Although the system is quite stable, it has problems sometimes (most of them caused by operation mistakes). Since this system is too old, even IBM doesn't have employees who know much about this system. Hence, when the system has problems, most of the time the users will spend time to find and fix it by themselves. If they fail to do that, they have to call some old guys who worked for IBM many years ago and who are familiar with MAPICS.

- Cooperation with other applications is difficult

  This ERP system does not have any built-functions that allow it to cooperate with other applications. Using the reporting generation as an example, every

month the company needs many different reports about the company's employment, production, sales and finance status. Although the ERP system has many build-in report forms, and users can also define their own required report, all the reports can only be produced from a special printer. Ten years ago, this kind of report would have been acceptable, but now, more and more businesses are handling their reports through the Internet. It happens very often in business that reports must be send around the world by E-mail. In this case, all that can be done is to get the data from the screen or from print, and type it up manually into a Word or Excel document.

- Difficult to update to the newest version

  In fact, this old version ERP system has even been abandoned by the developer. They have developed an ERP system from the bottom up. It is named INFOR (they even abandoned the name). The problems of updating to the new system mainly include:

  a) Budget – The new system is very expensive (several hundreds of thousands of USD)

  b) Adapting – For a long time, the old system has been highly adapted to fulfill the company's needs. If they updated it to a new one, they would need to spend more money and a lot of time on the new system.

c) Training – All the employees would need to be retrained. This would definitely affect the production of the company.

d) Data transferring – This is the biggest problem. Although the developer of INFOR can provide a data transferring tool, since the two systems are totally different, they can not guarantee that everything can be transferred properly. Because of the importance of the old ERP system, most managers at the company reject updating the ERP system, a type of change that was first proposed 3 – 5 years ago.

2) CAD Applications – AutoCAD

Currently, the company is using Pro Engineer, Solidworks and AutoCAD as designing tools. Among these applications, AutoCAD is the most important. The company started using AutoCAD as a computer-aided design tool many years ago, when AutoCAD was still in its DOS application format. Since they keep updating the version from 2.3, 10.0, 1998, 2002, and 2006, 95 present engineering drawings of the company are made by using different versions of AutoCAD. The differences between the drawings made by different AutoCAD versions are causing a lot of problems to the company. Some of them have tremendous negative effects on the product. The solutions of these problems are being eagerly wanted by the company. The main problems with these drawing include the following:

- Drawings made by old versions of AutoCAD can not be previewed in the new version.

AutoCAD has a very good function, namely the drawing preview. This function can display all the drawings in a folder as thumb nails. By using this function, every one easily finds his desired drawing without opening it. However, this function only works for drawings created by new versions of AutoCAD (From version 2000 on). To make this function work for all the drawings, they have to convert all the old drawings to the new format. Unfortunately, there are no good converting tools for these drawings. There is a very easy way to do it: open an old drawing in the new version of AutoCAD, and resave it. The problem is that the company has almost twenty thousand old drawings, so it is impossible to convert all the drawings manually.

- Naming chaos of drawings

Because of the long history of using AutoCAD as a designing tool, a huge number of the drawings have been made by many different designers. Since the company does not have any drawing name standard, people always save drawings under any kind of name they want. For example, in my opinion, all the drawings should be saved with the item's name. People could find the drawing using this kind of name without opening it. However, many people have not saved their drawings in this way. An examination of the drawing folders turns up countless drawings with names like too1.dwg, newmill.dwg, etc.

For these reasons, it is very difficult to find a wanted drawing quickly. It will be even worse for a new employee who has no idea where and under what

possible name the desired drawing has been saved. In fact, even a person who has worked there for a long time sometimes needs to spend a considerable length of time looking for a single drawing.

- Duplicated drawings

  For different reasons, there are also many duplicated drawings. Some of them are the same drawings but with different names. Some of them are only small differences. Those duplicated drawings cause a great many problems for the designing and production people, who need to find a way to get rid of these problems.

- Batch printing

  Every day, the company needs many printed drawings for production, and sometimes they need a whole series with hundreds of drawings to be printed for checking, modification, and backup purposes. However, AutoCAD does not have a batch printing function. You can print these drawings only one by one.

3) Microsoft Office

Microsoft Office has been wildly used by countless companies and personal users around the world. Nowadays, more and more business documents are transferred through the Internet. Most of these documents are MS Office documents, such as Word documents, Excel documents, etc. Like other companies, this company also uses MS Office as the main editing and reading tool for various documents.

5.2.3.3 Identify Relations between Applications

1) ERP (MAPICS) vs. CAD Applications (AutoCAD)

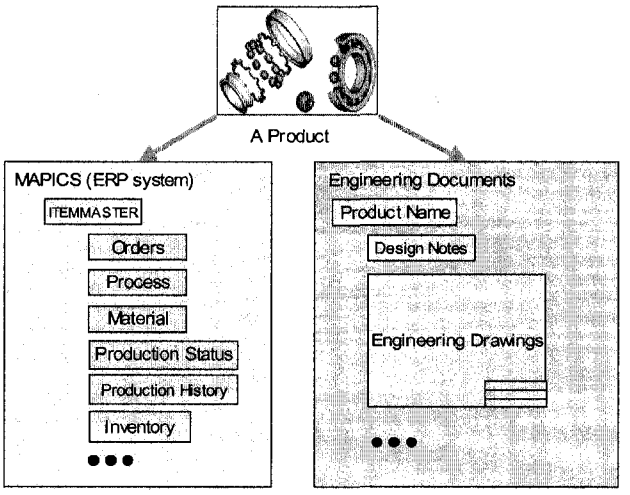


Figure 5-4 The relations between ERP and CAD applications

Figure 5-4 illustrates the relations between the ERP (MAPICS) and CAD applications. For any product, there are mainly two kinds of documentations. One is the records in the ERP (MAPICS) system, which includes all the production information, market information, financial information, inventory information, etc. The other is engineering documents such as meeting memos, design reports, calculations, and engineering drawings. When an order is received, both of these documents need to be reviewed or created. However, there is no automatic way to link these two together. They can be accessed only separately by employees, who then put the results together.

2) ERP (MAPICS) vs. Other Applications (MS Office)

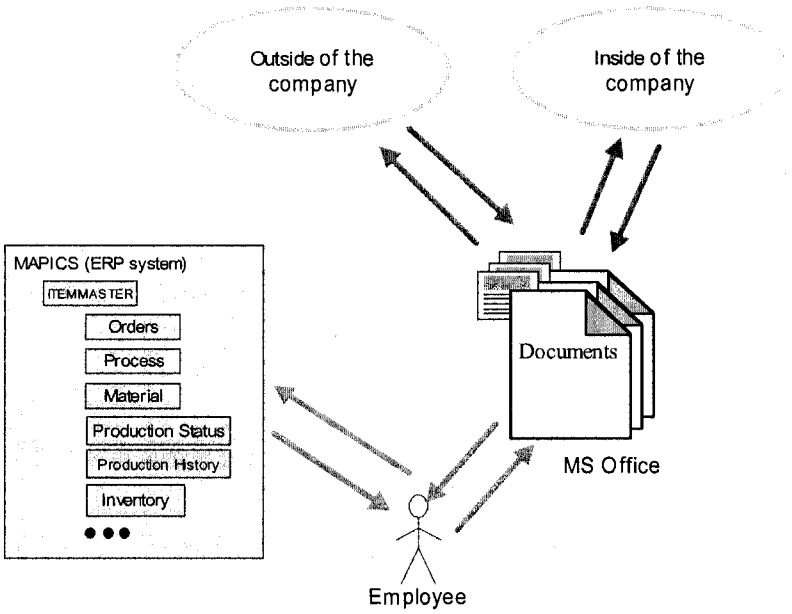


Figure 5-5 The relations between ERP (MAPICS) and MS Office

Figure 5-5 shows how MS Office documents work with the main information resource of the company, MAPICS. MAPICS can not produce MS office documents. In fact, the MAPICS can produce some electrical format reports, like text files, but they are difficult to use with MSOffice directly. So, employees are always used to serve as the link between these two. Certainly, they need to type a lot to do this kind of job, and the efficiency always is a problem.

3) Applications vs. Physical Devices (Printers and Plotters)

Printers are very important devices for a company. All reports and other documents need to be printed. For instance, in this company, every day they need to pint around 100 drawings from AutoCAD. Since AutoCAD does not have a batch print function, all these drawings have to be printed one by one manually.

5.2.4 Conflicts and Solutions

1) Updating all the applications vs. developing an EAI application

From the environment analysis, we know that the current application chaos has to be solved. The problem is how to solve it. Should the company buy and update all the applications or develop an EAI application?

Problems	Update all the applications	Develop an EAI application
Cost	High	Low
Better interface of the ERP	Yes	Yes
Affect the current production	Yes	No
Data transfer	Yes	No
Employee Training	Yes	No
Integration of ERP and CAD applications	No	Yes
Integration of ERP and MS Office	Yes	Yes
Integration of CAD application with other physical devises	No	Yes

Table 5-1 Updating or EAI developing

Table 5-1 clearly shows that the best solution for this company's problem is to develop an EAI application that can maintain the pros of current applications, especially the MAPICS system, and eliminate the cons related to these applications.

However, there is a new problem: What kind of integration application should be developed? Through the environment analysis process, a solution was found. The concept of this solution is using an "operator" application to replace the "user" in Figure 5-6. In this way, maximum freedom for the development of the EAI application would be achieved. This would fulfill needs of the company. Moreover, the modifications to the old system would be kept to a minimum. The concept is illustrated in Figure 5-7.

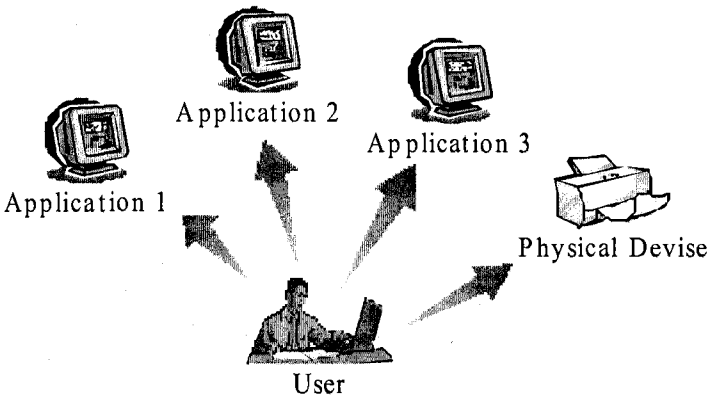


Figure 5-6 Traditional working model

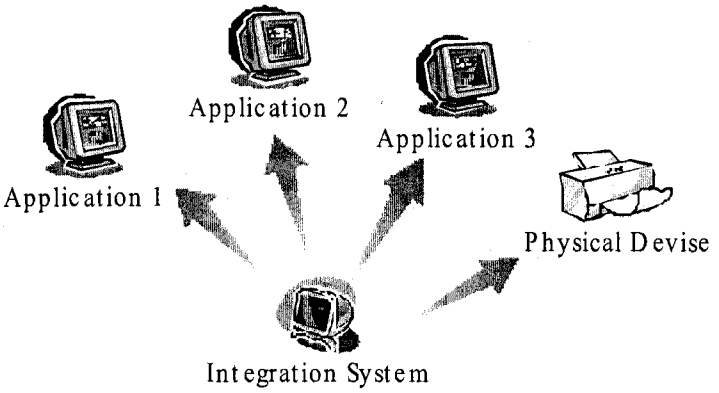


Figure 5-7 The concept of EAI application

5.2.5 Conclusion

The purpose of this pattern is to refine the real problem and its solution: finding a way to solve the application chaos between the ERP, CAD, other applications and some physical devices in a company. Through the environment analysis process and conflict identification, we finally get the best solution for the application problems in that company, which is to develop an application to integrate all the applications and devices and keep all the pros and eliminate most of cons. The new challenge for this solution is how to find a better way to develop this application. A child pattern "Developing an Enterprise Application Integration Application" will follow.

5.3 Child pattern (Pattern No.1.1): Developing an EAI Application

5.3.1 Name

Developing an Enterprise Application Integration system for the company

5.3.2 Problem

From the parent pattern we find that the best solution for the company is to develop an enterprise application integration system that can eliminate all the key negatives, keep all the pros, and enhance them as well. The problems that need to be solved in this pattern include: How to design this application? What kind of developing tools will be chosen? What kind of data base should be used? What are the challenges?

5.3.3 Environment Analysis

5.3.3.1 Find Out Important Environment Components

The key environment components of this problem are illustrated in Figure 5-8. The question marks denote unknown extra environment components, which can be discovered and added later during the environment analysis process or other researching processes.

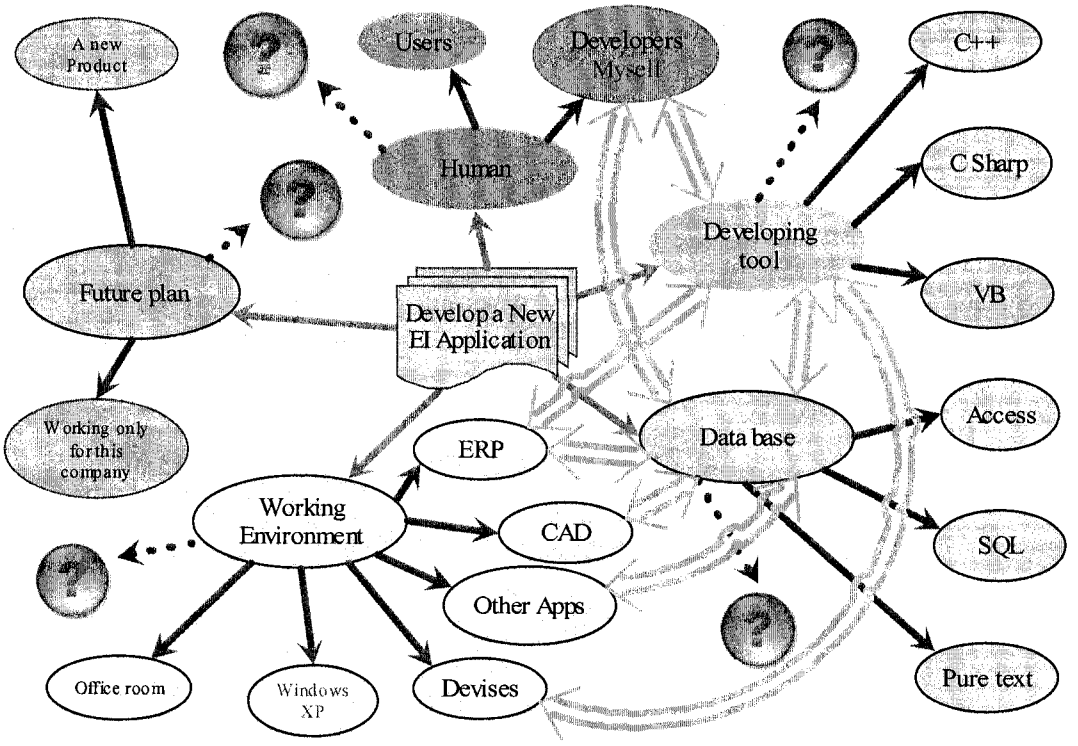


Figure 5-8 The key environment components

5.3.3.2 Identify the Internal Relations of the Key Environment Components

1) Developer

The developers are very important environment components of a project. A good understanding of the developers for a project is necessary. Here, I'll use myself as an example:

- Knowledge I have
  - ✓ VB, VC, C# Programming
  - ✓ Mechanical design and manufacturing
  - ✓ Operating of MAPICS
  - ✓ Operation of MS Office
  - ✓ AutoCAD
- Knowledge needed
  - ✓ Enterprise Application Integration

2) Data base

The database is the foundation for any kind of EAI applications. The database we can choose from includes: SQL, ACCESS, etc. Text files can also be used to save data under some circumstances.

3) Developing tool

Actually, what kind of developing tool should be used is not an important issue. We have the freedom to choose any developing tool that can easily achieve the function we need.

5.3.3.3 Identify the Relations between Key Environment Components

1) ERP (MAPICS) vs. Developing tool

By checking the user manuals and other documents of MAPICS, we found that this ERP system has a script language built for adapting designs. However, since it was developed a long time ago, the function of this script language is limited and very difficult to use. Moreover, we have no way to access the source code of the system. This makes it even more difficult to add new functions to the system by programming.

2) ERP (MAPICS) vs. Data base

The MAPICS system has a complex database in the server. To avoid any mistakes, we should use another database for the integration application rather than the main ERP database directly.

3) CAD vs. Developing tool

Here we would like to discuss only the developing tools for AutoCAD. AutoCAD has many built-in methods for adapting design and secondary development.

- ObjectARX

  The ObjectARX programming environment provides object-oriented C++, C# and VB .NET application programming interfaces for developers to use, customize, and extend AutoCAD software and AutoCAD-based products like AutoCAD Architecture, AutoCAD Mechanical, and AutoCAD Land Desktop software. ObjectARX libraries provide a versatile set of tools for application developers to take advantage of the open architecture of AutoCAD software and provide direct access to the AutoCAD database structures, graphics system, and native command definition. (Autodesk).

- VBA for AutoCAD

VBA (Microsoft Visual Basic for Applications) for AutoCAD is a combination of the AutoCAD ActiveX object model in AutoCAD and Microsoft Visual Basic for Applications (VBA), which presents a compelling framework for customizing the AutoCAD software program (Autodesk).

- Visual Lisp

Visual LISP is a tool provided by Autodesk for code creation in the AutoCAD application. It is a full-featured, interpretive programming language that you can use to call AutoCAD commands, system variables, and dialog boxes (Autodesk).

- AutoCAD script

AutoCAD script is another tool built into AutoCAD; it has the ability to access the AutoCAD commands and variables. Compared with the methods and tools listed before, AutoCAD script has more limitations in AutoCAD secondary development, but it is very easy to use.

- ActiveX Automation

ActiveX Automation is a built-in feature in AutoCAD since Version 14. Using this feature, you can customize AutoCAD and create custom applications by combining AutoCAD objects with objects from other applications that support ActiveX Automation. With ActiveX Automation, you can develop custom applications by using other programming application such as C++, VB, and C#.

5.3.4 Conflicts and Solutions

1) Developing tools vs. desired functions

To develop the EAI application, the developing tool (programming language) must be selected first. A better developing tool can make things that much easer. According to the EAI application concept illustrated in Figure 5-7, the EAI application should have some special function:

- Database access – C#, VB, VC
- Interface design - C#, VB, VC
- Keyboard and mouse control

Since the concept of the EAI application is to imitate a human operator operating different applications, which was presented in pattern No.1, it must have functions to control the keyboard and mouse. If using C#, VB or VC, we must write functions to achieve this. Therefore, we did some research on programming tools. Finally, we chose AutoHotkey as the developing tool for this project. There are several main reasons as follows:

- ✓ AutoHotkey is totally free and open-source. This feature is very important because the company does not have any commercial programming applications. Using AutoHotkey, we can write the application anywhere.
- ✓ Using AutoHotkey one can easily write a program with mouse and keyboard functions.

- ✓ AutoHotkey can design a windows GUI very quickly.
- ✓ The syntax of AutoHotkey is very close to C++. We don't need to spend too much time to learn it.
- ✓ AutoHotkey has a very powerful text-file manipulating capacity.

2) Database vs. desired functions

Since AutoHotkey handles text files very easily, we decided to use formatted text files to save data for the EAI application.

3) Developing tools vs. the EAI application concept

After the developing tools and database were selected, we refined the design concept. A more detailed structure of the EAI application is illustrated in Figure 5-9.

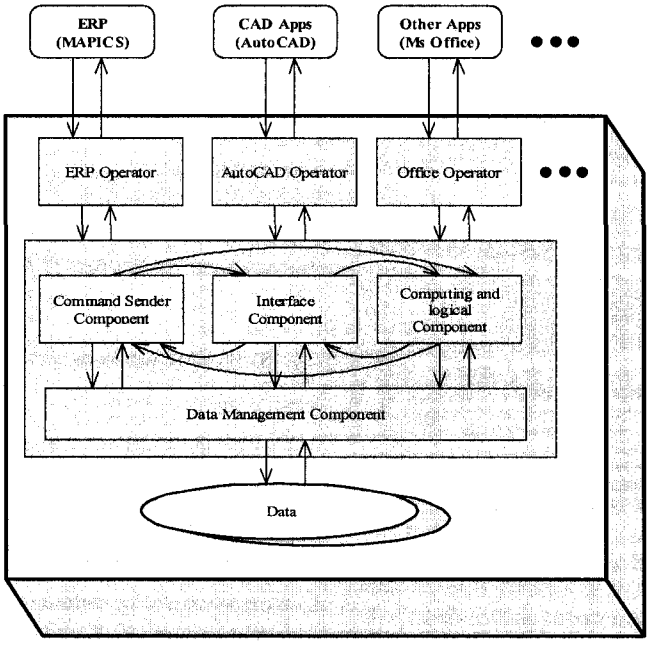


Figure 5-9 The structure of the EAI application

5.3.5 Conclusion

The purpose of this pattern is to find the best way to develop the EAI application for the company. Based on the environment analysis, a new developing tool, AutoHotkey, was selected. Formatted text files were selected to serve as the database of the entire system. After the developing tools and database are selected, a refined and detailed concept was given according to the refined concept. Four child patterns will be needed to solve four functional requirements: 1. MAPICS operation component; 2. AutoCAD operation component; 3. Office operation component; 4. device operation component.

5.4 Child pattern (Pattern No.2.1): ERP (MAPICS) Operation Component

5.4.1 Name

MAPICS operator designing

5.4.2 Problem

The key aim of this pattern is to find the solutions for the integration of the MAPICS system. In other words, the goal of this pattern is to find ways of communicating with MAPICS. The sub-problems include: How to get the information form MAPICS? How to send commands to the right place? What kind of database should be used? How to access the data?

5.4.3 Environment Analysis

5.4.3.1 Find Out Important Environment Components

Figure 5-10 illustrates the important environment components for the ERP (MAPICS) operation component (MAPICS operator).

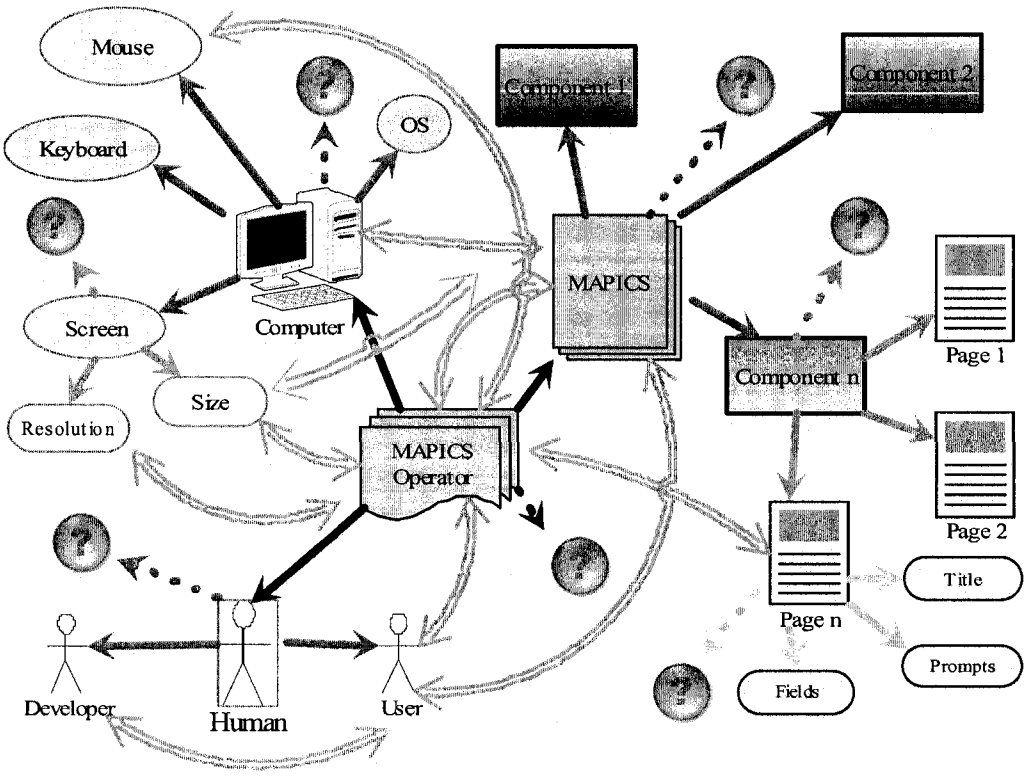


Figure 5-10 The key environment components for MAPICS operator

5.4.3.2 Identify the Internal Relations of Key Environment Components

1) MAPICS

As I mentioned before, MPPICS is the first generation ERP system developed by IBM more than 10 years ago. The client application of this system is a primitive Windows application. It contains many pages that look like Windows' DOS application windows. All production data, sales data, financial data, and other data would be entered, modified, displayed on particular pages. To access these different kinds of data, the users have to type corresponding commands into the command field. For example, if you want to check the description of a tool, you follow these steps:

➔ IM (Open the page of Inventory management main menu)

➔ 1 (go to the page of Item-Master management menu)

➔ Type the Item-Master of the tool in a special field

➔ An information page of the tool will appear, and you can get the description and other information from this page.

2) Computer

Because the EAI system will imitate a human operator operating the applications automatically, what kind of computer the EAI system will be running on is very important. The configuration of the computer is the following: Pentium4, Windows XP, 15" screen, standard mouse and keyboard, etc.

5.4.3.3 Identify the Relations between Key Environment Components

1) Computer screen and Applications

When we are operating an application, we use mouse and keyboard to send commands to the application's GUI displayed on the screen, and we give the responses to the messages given by the application. The most important issue for this operating cycle is the position of the messages: from which window, from which position can get the needed information? To which window, to which position should we give the feedback? These positions are directly related to the resolution of the screen; the title (handle) of the message window, and the initial position of the window as well.

2) Applications and database

Take the creation of a report in Word as an example:

➔ Start MAPICS and WORD. (initialize the working environment)

➔ Send command to the MAPICS and get the page containing the required information. (send messages)

➔ Write the notes of the information. (save information to a database)

➔ Activate the Word window and read the notes and type it into the report. (Get information from the database and send it to another application)

The relations between the database and applications are illustrated in Figure 5-11

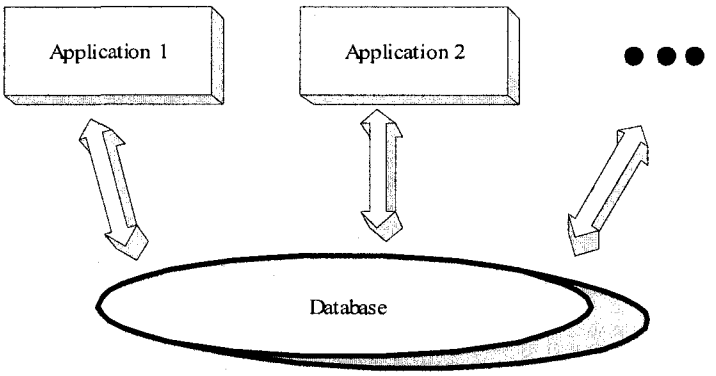


Figure 5-11 The relations between the database and applications

5.4.4 Conflicts and Solutions

1) Exchanging messages with MAPICS manually vs. automatically.

Since we want to let the EAI system imitate a human operator operating MAPICS and other applications, we must find a way for the EAI system to communicate with MAPICS automatically.

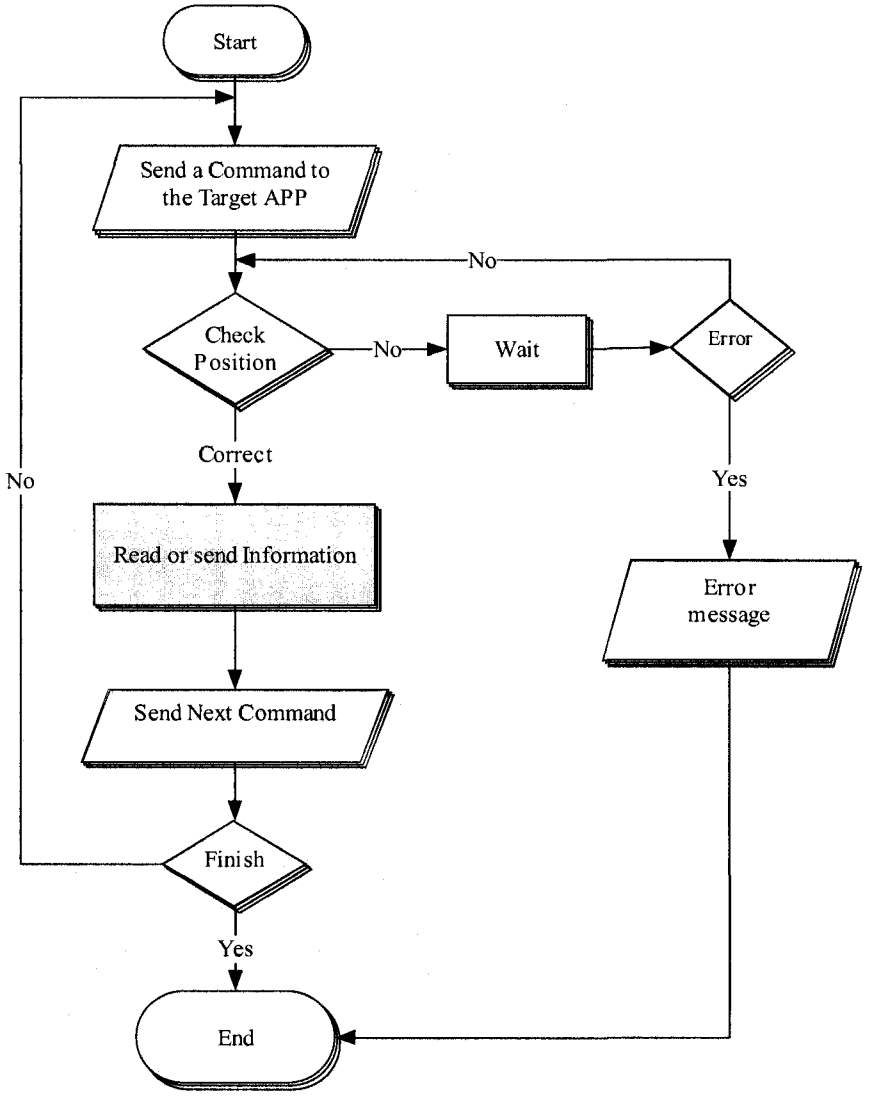


Figure 5-12 Communicating with another application automatically

Figure 5-12 shows the concept for the EAI system to automatically communicate with another target application. There are two sub-problems related to this concept. The first problem is: How to check whether the target application is displaying the desired window or data page? The second problem is: How to make the EAI system automatically do a series of work?

i) How to check whether the displayed page is the desired page.

Human operators can easily verify whether the running program is the program they want by checking the title and other displayed information on the program window. They can also easily determine whether the displayed page of a program contains the information they want by checking the promotions, pictures, etc.

The EAI system needs to do the same thing as if it were a human operator. To achieve this, we must give the EAI the ability to recognize a running application through the application windows, messages, special displayed information, or promotions.

ii) How to automate the program

Many methods can be used, such as a repeated working cycle.

5.4.5 Conclusion

This pattern focuses on the communication method between the EAI system and MAPICS (The ERP system currently used by the company). The concept of this MAPICS operator is the following: When the EAI application receives a job command, it will act like a human operator and automatically operate the MAPICS system. It should do anything a human operator can do, namely adding new entries, modifying the data base, acquiring the required information, sending command signals, etc. The only difference between the EAI system and a human operator is that the EAI system is much faster, is more efficient, and makes fewer errors. Moreover, it can do many repeated jobs continually, even beyond working time. A few sub-problems have been solved within this pattern. The first one is to give the EAI

application the ability to determine whether the desired program (MAPICS) is running, and whether the opened page is the page it needs. The second is to find a way to let the EAI application manage its job list and data base.

5.5 The Child Pattern (Pattern No. 2.2: CAD Applications (AutoCAD) Operation Component

5.5.1 Name

AutoCAD operator designing

5.5.2 Problem

Since the company has a very long history of using AutoCAD as their most important computer-aided design tool, most engineering drawings in this company are AutoCAD drawings. Hence, compared with SolidWorks, Pro Engineer, and other engineering applications used in the company, AutoCAD has more problems. In this pattern, the problems that need to be solved include: How to solve the naming chaos of the engineering drawings in the company? How to convert 20 thousand drawings to the newest drawing version? What kind of database should be used to save the drawings-related information? How to retrieve the contents information from more than 20 thousand drawings and save them into the database?

5.5.3 Environment Analysis

5.5.3.1 Find Out Important Environment Components

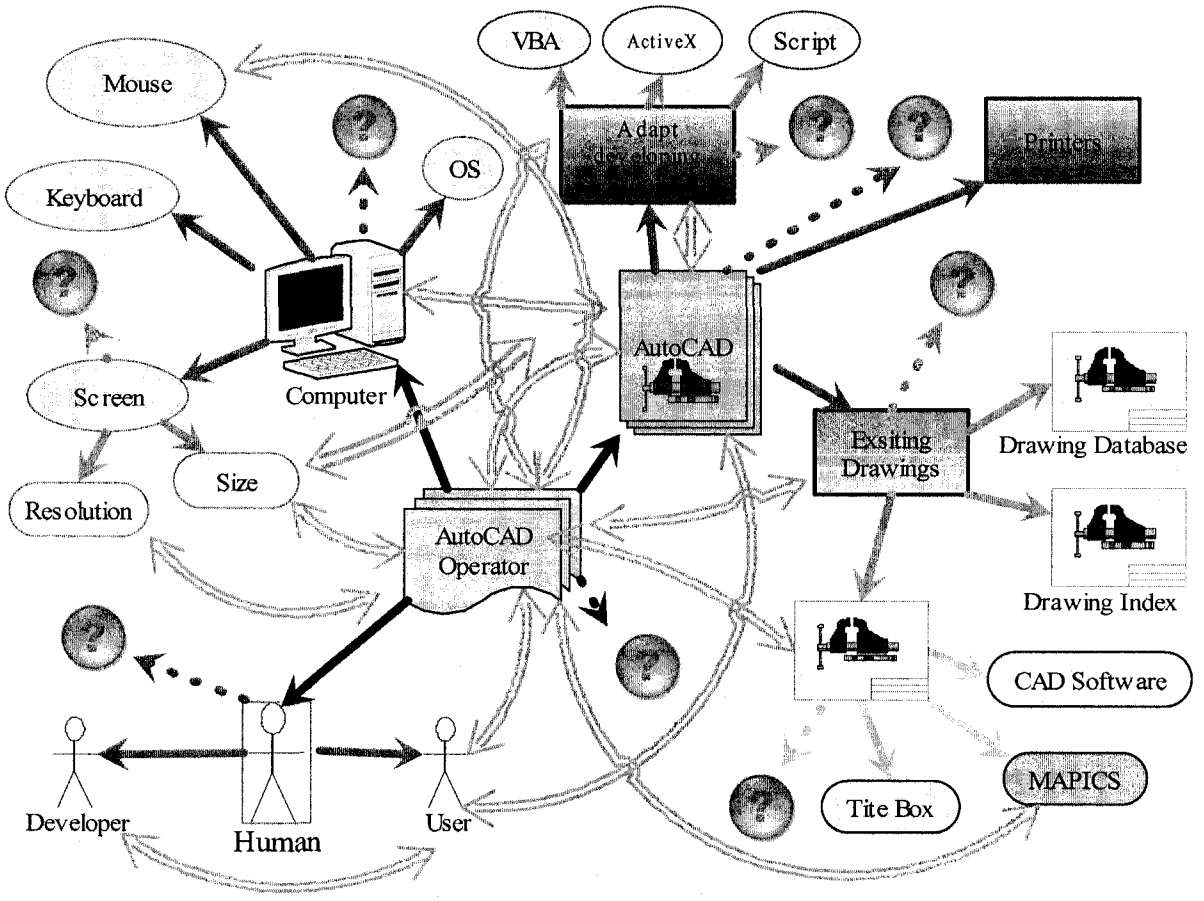


Figure 5-13 The key environments for AutoCAD operator

5.5.3.2 Identify the Internal Relations of Key Environment Components

1) AutoCAD

Discussed in Pattern No.1

2) MAPICS

Discussed in Pattern No. 2.1

3) A DWG AutoCAD Drawing

The most widely used drawing format for AutoCAD is the DWG format. It is the internal format for AutoCAD and is used for storing two and three dimensional design data. The first version of DWG was developed in the 1970's. Since that time, new versions have been continually introduced along with the development of AutoCAD. Old versions of DWG drawings usually are supported by new versions of AutoCAD. For example, AutoCAD 2008 will have the ability to open versions of DWG drawings back to 2.0, and to save them back to 14.0. A DWG drawing contains layers, blocks, line types, fonts, and many other objects. Through the secondary development tools provided by Autodesk, we can control most of the properties of the objects in a DWG drawing.

4) Secondary development tools for AutoCAD

Discussed in Pattern No. 1.1

5.5.3.3 Identify the Relations between Key Environment Components

1) Engineering drawings vs. CAD applications

The relations between engineering drawings and CAD applications in the company have been discussed in the previous chapters. Here I would like to give just a brief description. In the company, more than 95 percent of the engineering drawings are made by different AutoCADs. These drawings are all DWG format drawings. They are being used for production and as the references for new product designs. The problems related to these drawings include format variance, naming chaos, and big quantities. Other relations between engineering drawings and AutoCAD include how the AutoCAD works with a drawing. How are different versions of drawings managed by AutoCAD applications and in-built tools, etc?

2) AutoCAD vs. computers in the company

To find a best solution for this pattern: AutoCAD operator designing. We must clarify the relations between AutoCAD and the computers that run it.

5.5.4 Conflicts and Solutions

1) The Conflicts between MAPICS and engineering drawings.

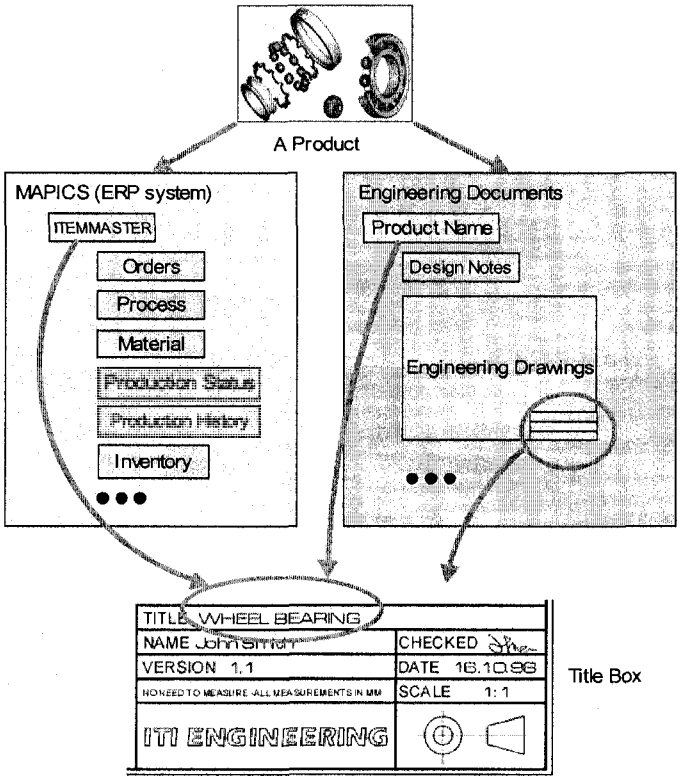


Figure 5-14 The connection between MAPICS and engineering documents

In this company, for any product, they have two kinds of documentation. One is the recodes in MAPICS; the other one is engineering documents such as drawings, designing reports, engineering notes, etc (Figure 5-14). This means that every time they receive a new order they have to create a new Item-Master with a unique name for a product in the ERP system. All the information like orders, customer's name,

product description, cost, unit price, manufacturing procedures, materials, production history, and current inventory status will be saved in the ERP system by using this Item-Master. The next step is designing. The engineers make the design in CAD applications and save the engineering drawings on the server by using the names they have given. Take the bearing in Figure 5-14 as an example.

- They received an order with one item: a special wheel bearing, 2, 5, 2.5, RPM 20000.

- In the ERP system, they will create an Item-Master like *SPB2x5x2-1/2*, which must be unique in the system

- Enter the order and the customer information into the system. Other information will be entered by different employees later.

- An engineer will make the design and save the drawing on the server by using a name like *Special Bearing 2x5x2.5.dwg* or *newbearing.dwg*

From this example, we can easily notice the conflicts between ERP and CAD naming methods. For the ERP system, we need a unique name (Item-Master) that contains as much information as possible about the product (the length of Item-Master usually is limited by the ERP system). For CAD, people can freely use different file names as they want. So, to integrate MAPICS and AutoCAD, we must solve this conflict.

Here we named this conflict the Naming Conflict. Since this is a relatively complex problem, we need a separate pattern (Pattern No. 2.2.1) to generate the concepts and final solutions.

2) The conflicts between AutoCAD's built-in functions and the user's needs.

As we discussed before, to integrate AutoCAD with MAPICS, some functions must be added to the EAI system to compensate for the shortages of AutoCAD, such as batch printing, batch drawing loading, batch converting.

5.5.5 Conclusion

The objective of this pattern is to create and find the solutions for the AutoCAD operator, and through the operator to integrate the AutoCAD application with the MAPICS ERP system. Through the environment analysis, relation finding, and conflict identifying, we finally found a solution for the problem: Using the name of the drawings and the corresponding Item-Masters created for the same products in the MAPICS as the bridge to link AutoCAD drawings and MAPICS. However, the naming chaos can not be solved within this pattern; a new pattern for this problem will be as follows:

5.6 The Child pattern (Pattern No. 2.2.1): Solutions for Naming Conflict

5.6.1 Name

Generate the solution for Naming Conflict

5.6.2 Problem

The target of this pattern is to find the best solution for the *Naming Conflict* identified in pattern No 2.2. The detailed design problem for the current pattern is to find a method that can link any unique Item-Master in MAPICS with the corresponding

engineering drawing saved in the drawing server without changing the existing drawings and the designers' drawing naming habits.

5.6.3 Environment Analysis

5.6.3.1 Find Out Important Environment Components

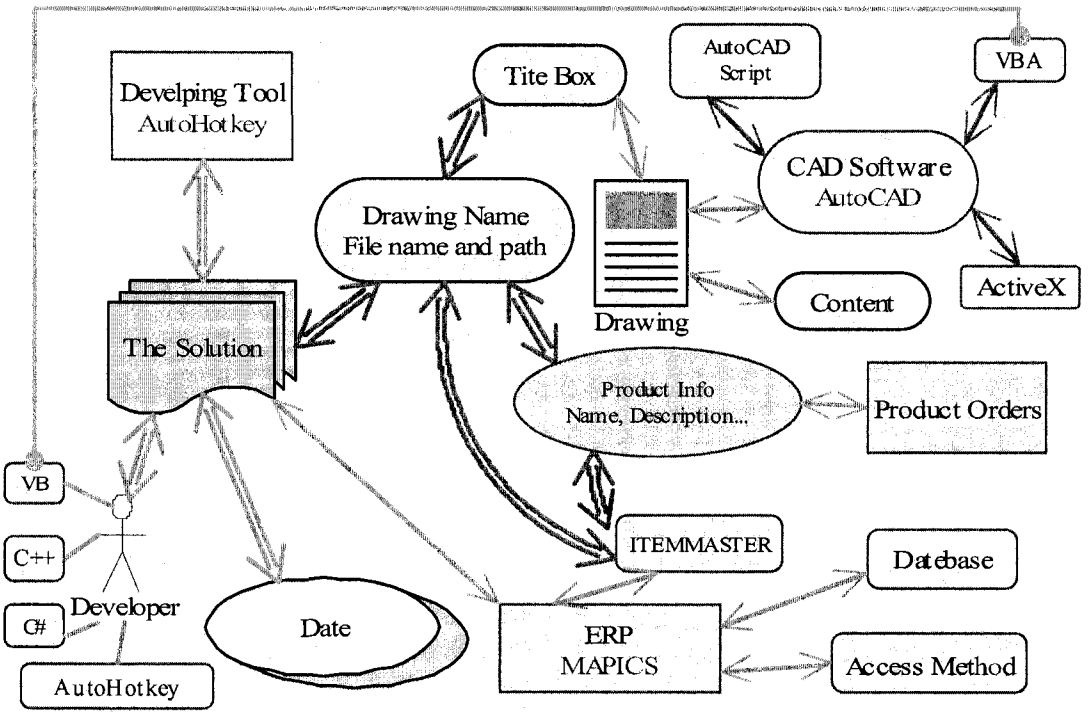


Figure 5-15 The key Environment components for the solution of Naming Conflict

5.6.3.2 Identify the Internal Relations of Key Environment Components

To solve this problem, we must make clear what the internal relations for each key environment components are.

1) Inside of an engineering drawing

Usually, an engineering drawing contains three important parts: Drawing, Requirements and Title box. After checking many of the drawings in the company, we obtained a result: although many drawings were saved with strange names that

have no relationship to their content, there is something in common with the title boxes. Almost all the engineering drawings in this company have a similar format of the title box, which contains the information such as the product name, item name, product description, custom, designer, etc. Since all the drawings need to be printed and sent to the workshop for production, the title box is usually filled out carefully. Especially, the item name usually almost the same as that which the Item-Master created for this product in MAPICS. For example, in MAPICS, the Item-Master is IBM3230. In the title box of the corresponding drawing, the item name is usually filled with IBM3230, IBM-3230 or IBM 3230. So, we can use the content in the title boxes to link the CAD drawings to the items in MAPICS.

- How to get the information in the title box from a drawing and a batch of drawings.

- What kind of database should be used to save this information.

2) The developing tools of AutoCAD

AutoCAD has some built-in adapting developing tools such as VBA, AutoCAD script, ActiveX, etc.

3) The data management capacities for AutoHotkey

We have discussed this before.

4) The developing ability of the developer, myself

As Figure 5-15 shows, I'm familiar with developing tools like VB, C++, C# and AutoHotkey.

5.6.3.3 Identify the Relations between Key Environment Components

1) The relations between developing tools

As Figure 5-15 shows, AutoCAD's built-in developing tool VBA is matched with one of the developer's ability VB, so it would be easier to use this to solve certain problems.

2) Relations between engineering drawings and MAPICS

We have discussed these relations before. Here we will give some ideas about the integration of MAPICS and the existing drawings. When we receive an order, we check all the related Item-Masters and make a list. The integration application uses this list to get all the related drawings and to print them automatically.

5.6.4 Conflicts and Solutions

1) Title box information vs. retrieve methods.

The conflict can be interpreted as a problem: How to get the information in the title box from a drawing?

During the environment analysis stage, we got the idea to use VBA as the developing tool to achieve this.

Since the purpose for retrieving the information from the title box of a drawing is to share it with other applications, a database or other data-saving method must be selected first.

A. Select the right database for the drawing information and the corresponding ItemMasters.

The working flow and the function of a drawing-named database can be illustrated in Figure 5-16

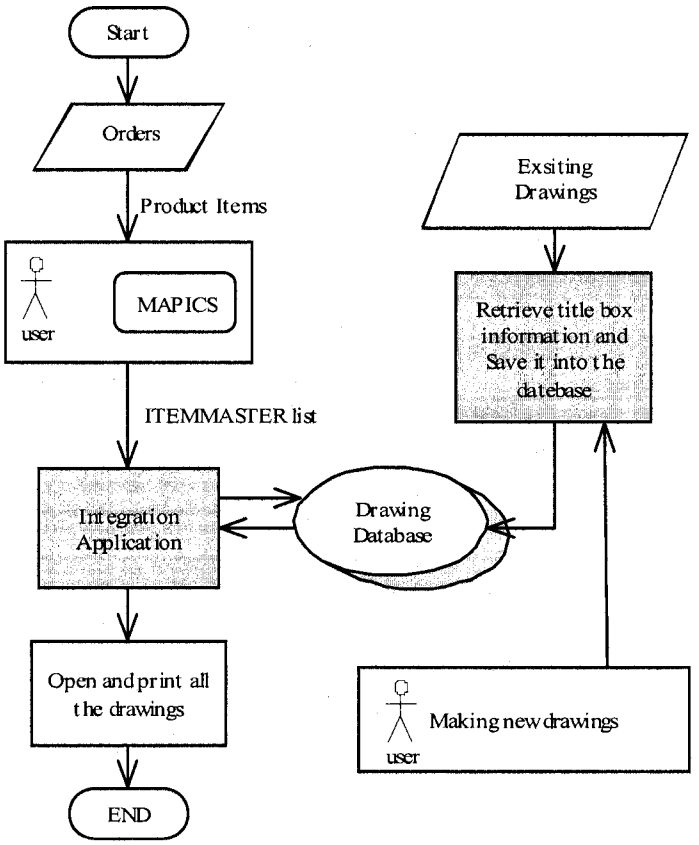


Figure 5-16 The working flow and the function of a drawing-named database

After thinking about the whole system, we made the decision to use a standard INI file, or an initialization file, as the database to save all the information retrieved from the drawings. The reasons are the following:

- Free of charge.
- Easy to handle both by AutoCAD VBA and AutoHotkey
- Powerful and easy to maintain

An INI file is a configuration file that contains data, and it is widely used to save configurations for applications.

The standard format for an INI file is illustrated in Figure 5-17:

```
[section1]

; some comment on section1
var1 = abc
var2 = 451

[section2]

; another comment
var1 = 123
var2 = dfg
```

Figure 5-17 The standard format of an INI file

B. Use a VBA application to retrieve the drawing information for a drawing opened by AutoCAD

After the VBA has been selected and the format of the database has been determined, we can start to write the program in the VBA editor built-in AutoCAD. Figure 5-18 shows the VBA programming environment built in AutoCAD. The displayed code is

the source code for retrieving the drawing data. The background of the figure is AutoCAD 2006 with an opened engineering drawing. The VBA code is saved with the name acad.dvb in the AutoCAD installation folder, so it can be loaded automatically when the AutoCAD main program is started.

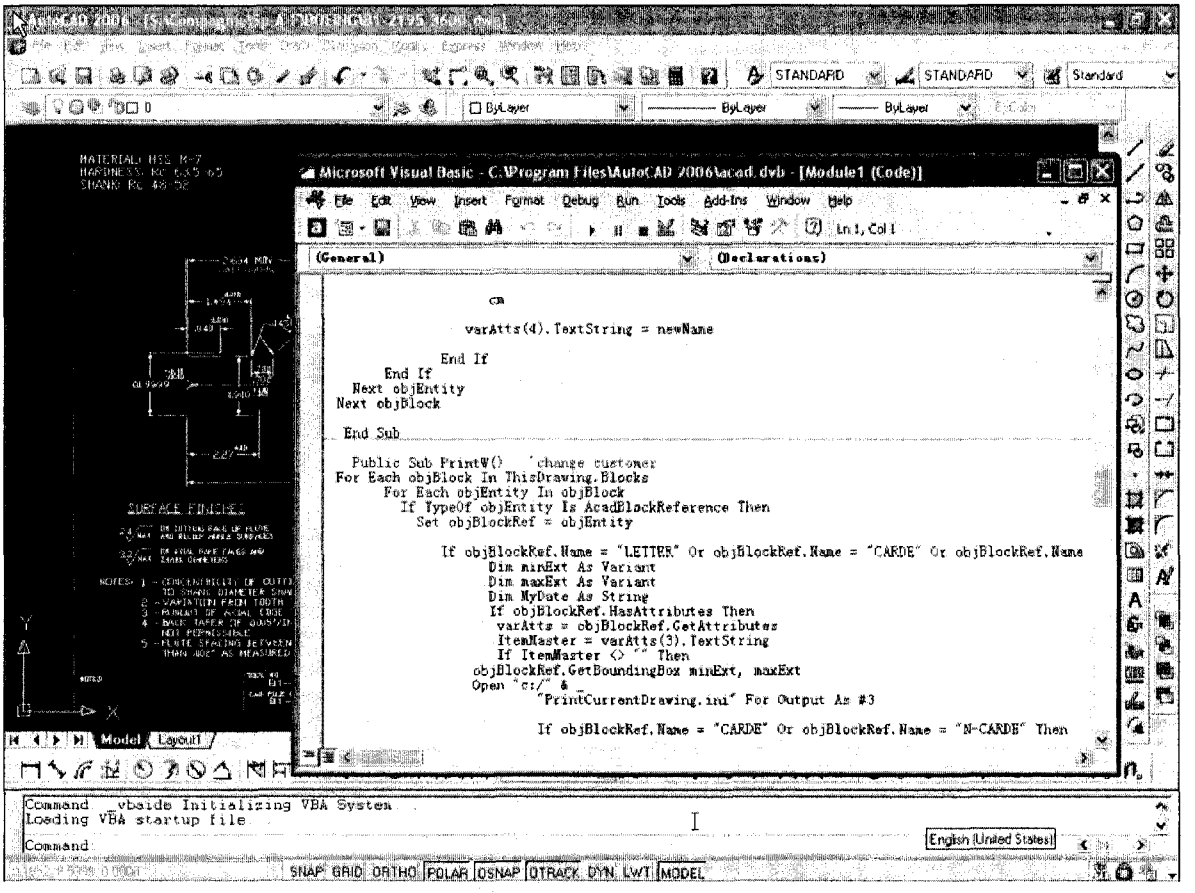


Figure 5-18 Using VBA to retrieve drawing information from an opened drawing

This VBA application has several functions. First, it can find the block name of the title box from the current opened drawing, read the needed information, and save all the information to the INI file serving as the database. Second, it can automatically print a drawing in a predefined format when the print command is given.

2) Access speed vs. INI file

This conflict is generated by the previous solutions. We know that an INI file is a kind of text file rather than a true database. Although it is very easy to use, when it contains too mach data, the access speed will be significantly reduced. In this company, they have at least 20 thousand existing drawings. If we need 1k bites for one drawing, the total size of the INI would reach 20M! It is certainly too big.

To solve this problem, instead of using one INI file, we used many INI files to save the information, using the first 2 letters of an item name as the INI file's name.

The solution to this problem is illustrated in Figure 5-19.

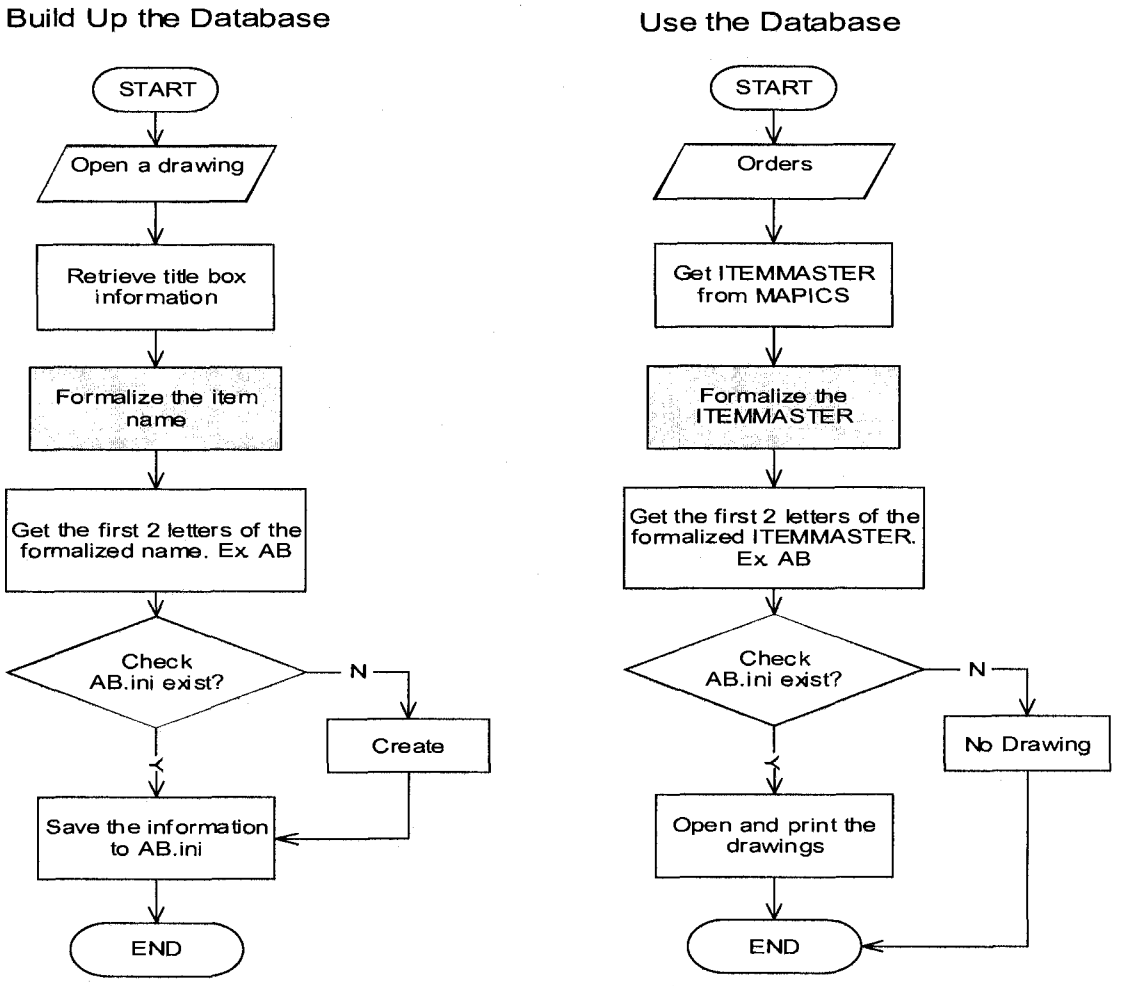


Figure 5-19 Formalizing drawing names and ItemMasters during access to the database

Figure 5-20 shows the item name formalization function in the AutoHotkey source code. In the acad.dvb, there is a similar function, too.

```
GoodName(Namestring)
{
   StringReplace, GoodDraw, Namestring, %A_SPACE%,,All
   StringReplace, GoodDraw, GoodDraw, .,, All
   StringReplace, GoodDraw, GoodDraw, RE-,, All
   StringReplace, GoodDraw, GoodDraw, \,, All
   StringReplace, GoodDraw, GoodDraw, /,, All
   StringReplace, GoodDraw, GoodDraw, RAD,R, All
   StringReplace, GoodDraw, GoodDraw, 0R,R, All
   StringReplace, GoodDraw, GoodDraw, -00,-, All
   StringReplace, GoodDraw, GoodDraw, -A,A, All
   StringReplace, GoodDraw, GoodDraw, -B,B, All

   StringLeft, FirstC, GoodDraw, 1
   if FirstC=0
   StringTrimLeft, GoodDraw, GoodDraw, 1

   MLength := StrLen(GoodDraw)
   if (MLength>7)
    StringReplace, GoodDraw, GoodDraw,-,, All
   Return GoodDraw
}
```

Figure 5-20 A formalization function

3) Building up the database from 20 thousand existing drawings. Should this be done manually or automatically?

This problem is relatively easier than the other problems when all the functions have been written. We wrote a program to create the INI data files. The program contains a main function and few error-trapping sub-functions. The main function uses a loop to automatically load all the drawings in the server one by one, to retrieve the drawing information and to save it in the data files. To make sure the program runs smoothly,

we set a 10 second delay before opening another drawing. That is, in one hour, the program can get information from around 500 drawings. We chose a weekend to run the program. After two days, the data files were built up successfully.

5.6.5 Conclusion

This pattern solved the conflicts between the drawing files and the Item-Masters' naming methods. INI files, or initialization files, are chosen to serve as the database. AutoCAD VBA was used to write a program to get the title box information and to save it into the data files. A method to avoid large data files was used. Finally, a database building program was written, and a database with hundreds of INI files for more than 20 thousand drawings was built up successfully.

5.7 Other Patterns and Related Problems

After the solutions for the main structure of the EAI system, the MPICS operator and the AutoCAD operator have been generated, and many related sub-problems have been solved in the patterns described before. Two more problems are left: First, the integration of MS Office (or other applications); second, the integration of physical devices (printers). To solve these two problems, at least two patterns are needed. Since the problem-solving processes are similar to the given patterns, the detailed problem-solving process will be omitted from this thesis.

5.8 A Brief Introduction to the Finished EAI System

1) The Main Components of the EAI System

The final EAI system for the company includes several applications. Figure 5-21 shows the interfaces of some of these applications.

- Job Issuing Management
- Drawing Management
- Closing Job Helper
- Barcode Correction Helper
- Item-Master Management
- Profit Margin Report Generator
- Batch Print for AutoCAD, Acrobat
- Spool Cleaner
- Special Calculators

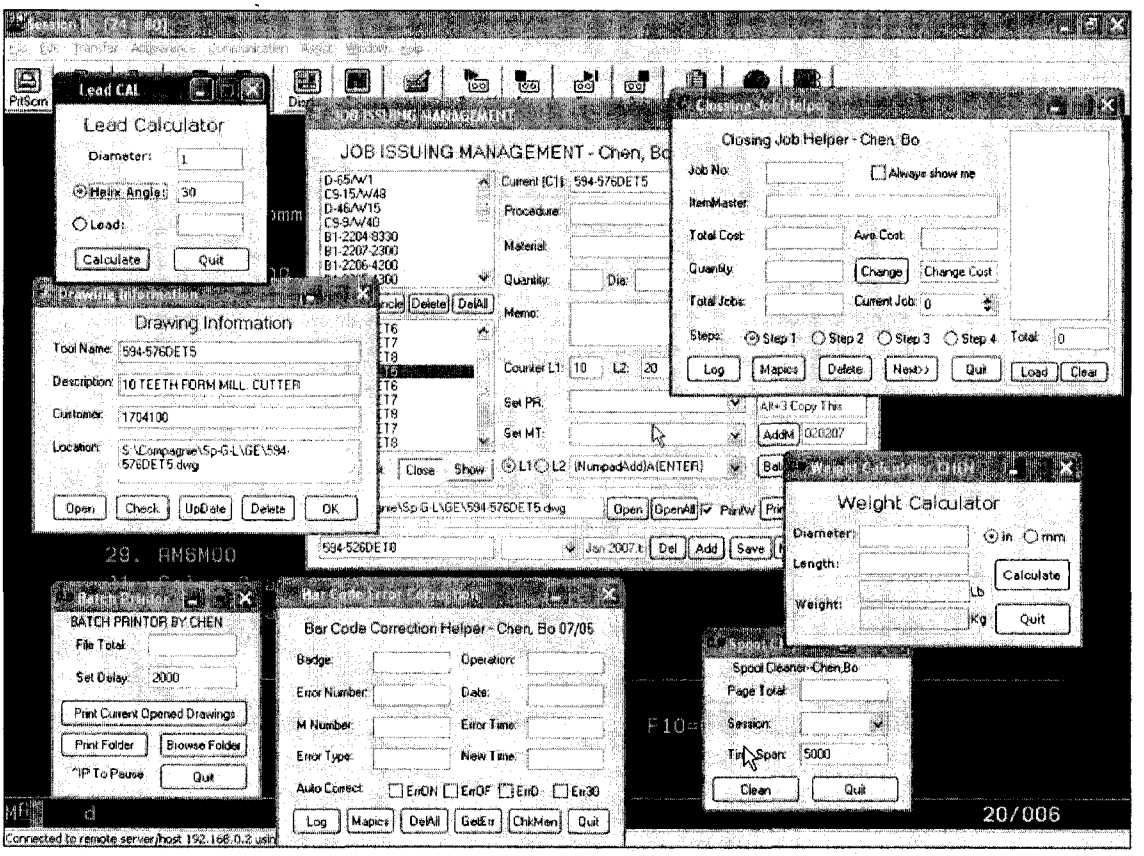


Figure 5-21 Applications for the EAI system

2) The Job-Issuing Management application

The Job-Issuing Management application is one of the most important applications for the system. It integrates MAPICS and AutoCAD through Item-Masters. In this application, we made three lists to manage all the Item-Masters. One is for waiting items; one is for prepared items; and the other one is for finished items. When an order is received, the user enters all the items into the preparing list. The system automatically checks the MAPICS using this list. If an Item-Master does not exist in the MAPICS, which means the item has never been made before, then the Item-Master is created by the Item-Master management application. Figure 5-22 shows the interface of the application Job-Issuing Management.

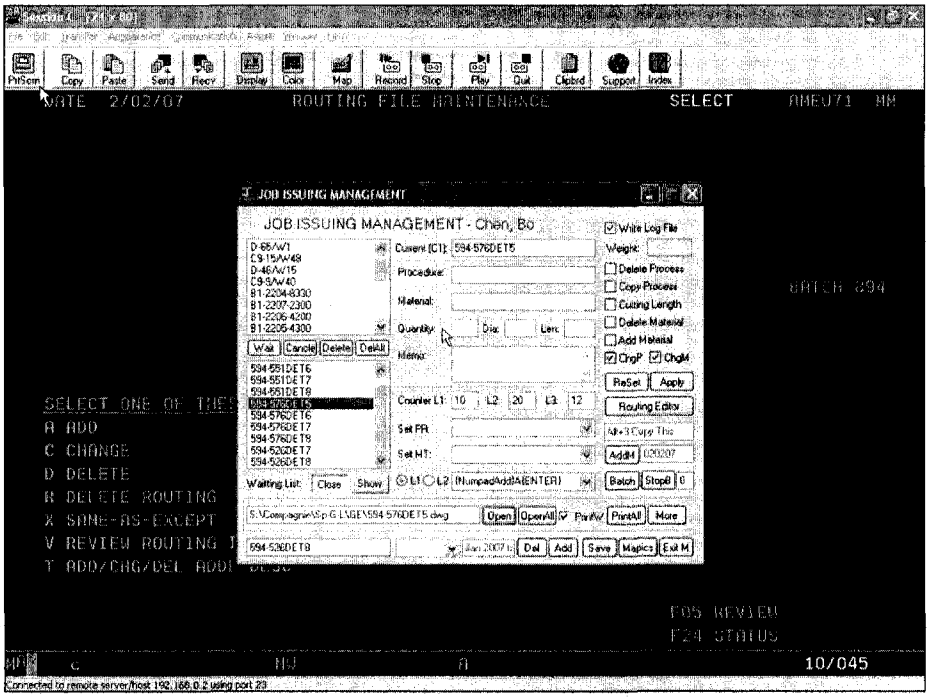


Figure 5-22 The interface of Job Issuing Management

Figure 5-23 shows the production process generator of the Job-Issuing Management application. By using this component, it becomes much easer to create or edit the production process for a product in the MAPICS.

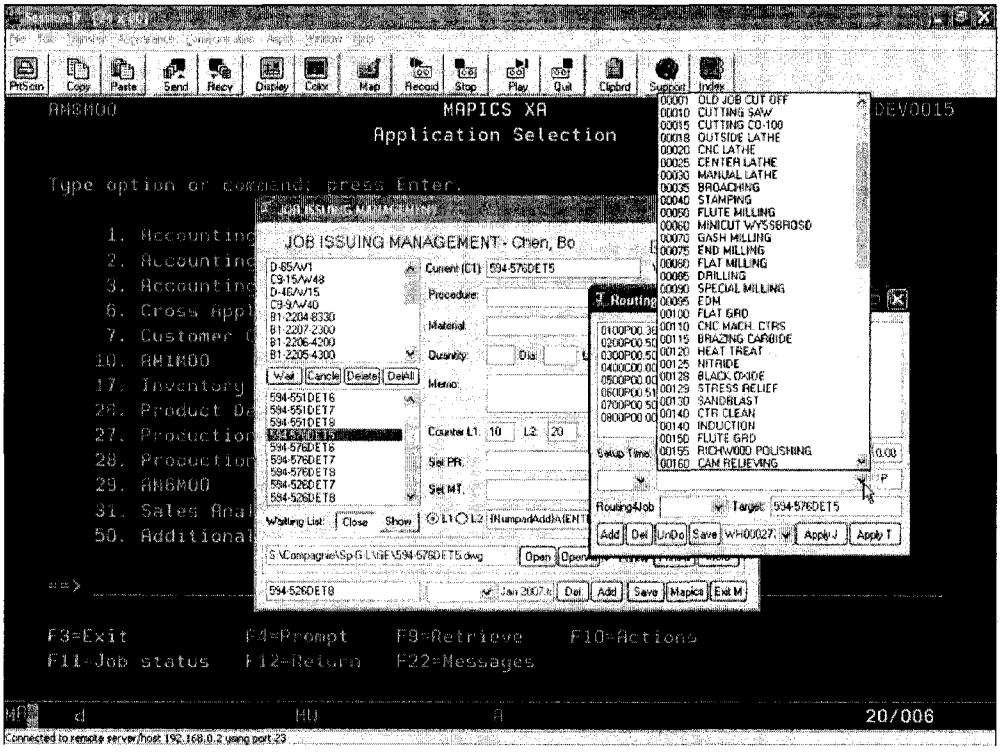


Figure 5-23 Production Process generator

Figure 5-24 shows the drawing information database management component of the Job Issuing Management application. By using this component, an item can be added or deleted manually from the database.

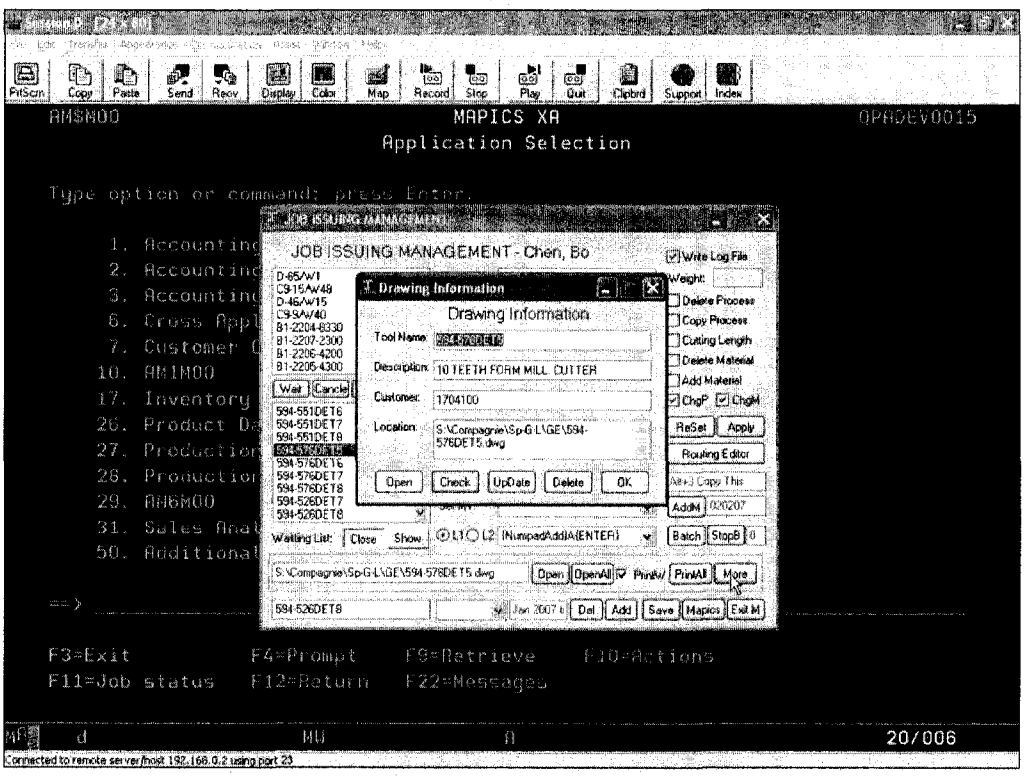


Figure 5-24 Drawing information database management

3) Reports generated by the EAI system

The EAI system can generate different reports and working logs. Figure 5-25 shows some reports generated by the system

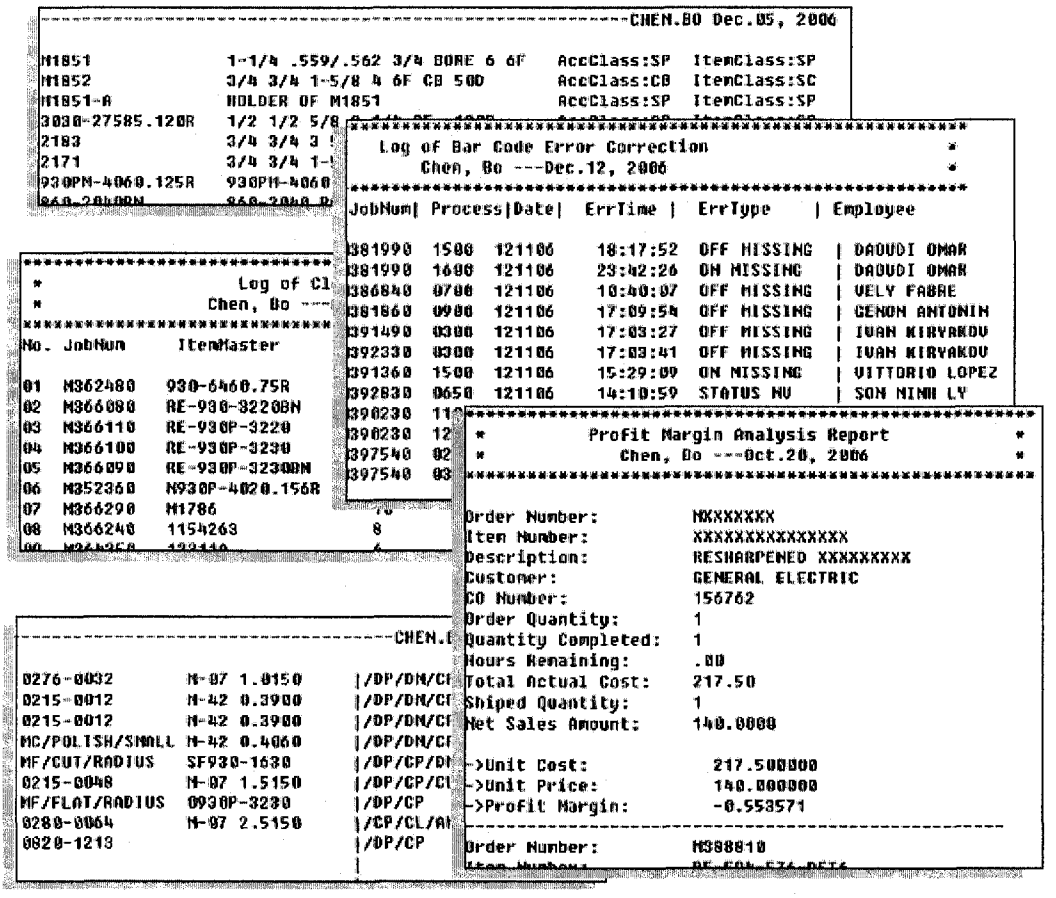


Figure 5-25 Reports generated by the EAI system

5.9 Conclusion of The Chapter

This chapter applies the EBD-EAI approach to solving a real EAI problem for a metal-cutting tool company. Three enterprise applications are integrated. These three applications are ERP (IBM MAPICS), MS Office, and AutoCAD. The Environment-Based Design method was used through the whole problem-solving process. Since many sub-problems were involved during the process, the structure of the problem solving was complex. To manage and organize the process, a pattern language was used. At the end of this chapter, a brief introduction of the finished EAI application was given.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

During the last few decades, Enterprise Application Integration (EAI) has become a very important issue for many organizations, enterprises, and companies. Many EAI methods and technologies have been developed or are under development. In my thesis, we used a different approach: Using Environment-Based Design (EBD) theory to solve an EAI problem. With the advantages of EDB theory, it becomes very easy to find the solutions for an EAI problem. On the other hand, most defectiveness solutions can be identified in the very early stage through the environment analysis process. Through the real EAI problem solving in Chapter 5, we can come to the following conclusions:

- EBD is a very effective and powerful design method, especially for conceptual design. Different from other design methods such as the Function-Based Design, when you are using the EBD method to design, while thinking about "what do I want to do?" you will always be reminded "what can do?" during the environment analysis process. So, most imperfect and defective solutions can be found and eliminated in the very early stage.

- The EBD approach of EAI problem solving can help the EAI developer to develop a successful EAI application both in function, cost, and time according to the particular environment of the EAI problem.

6.2 Future Work

Based pm the results of this thesis, some useful research still needs to be done as follows:

1) An environment analysis process management system. From the development process of the EAI application, we can find that the environment analysis is very important for every stage, such as understanding the design problem, identifying the key conflicts, and generating the concepts. Therefore, an environment analysis process management system would be very useful for both the EBD method and the EBD approach to EAI development. This management system should have the following functions:

- Environment brainstorming helper: Help the designer or designers to find out the important environment components in a systematic way. Instead of finding an environment component randomly, the system would help the designer or designers to find them from the built environment, the natural environment, the human environment, or the close environment and remote environment.

- Relations finder: One of the important stages for the EBD design process is to find out the relations between the environment components and the relations inside of each environment component. This system should help the designers to find out all the important relations by questioning or other methods. At the same time, the system will manage all the relations methodically for the next stage: identifying the key conflicts.

- Conflicts finder: Like the relations finder, this component helps the designer find and manage the key conflicts between all the environment components.

- Concept and solution management: For each conflict, we may find many solutions. It is common that these solutions have various relations with solutions to other key conflicts. This component should help the designers to mange and evaluate all the sub-solutions and sub-concepts, and finally to find the best combination (or combinations) of these solutions to form the final design concept (concepts) for the whole design problem.

2) For the EBD approach to EAI, there are important and useful future works that need to be done:

- EBD-EAI framework: It has been proved that the EBD approach to EAI can find a solution for a particular EAI problem using a shorter time cycle, lower cost, and effective integration. However, we still need to spend a great deal of time on coding. For instance, for the EAI problem at the metal-cutting tool company, we wrote almost 10000 lines of code. Therefore, if we have an EBD-EAI framework with many reusable components, such as data managing components, application control components, the development would become much easier.

- Combine OCR (Optical character recognition) technology with the application control component. This combination would improve the performance and reliability of an EAI system, and would also make the EAI system development more efficient.

Bibliography

Alexander, C., S. Ishikawa, et al. (1977). A Pattern Language: Towns, Buildings, Construction. Oxford, Oxford University Press.

Altshuller, G. S. (1986). To Find an Idea. Novosibirsk, Nauka.

Asimow, M. (1962). Introduction to design. Englewood Cliffs, N.J., Prentice-Hall.

Autodesk Developer Center - ObjectARX.

Chen, Z. (2006). Formalization and Classification of Product Requirements Using Axiomatic Theory of Design Modeling. The Department of Electrical and Computer Engineering. Montreal, Concordia University.

Coombs, C. H. and G. S. Avrunin (1988). The structure of conflict. New Jersey, Lawrence Erlbaum Associates, Inc.

Feilden, G. B. R. (1963). Engineering Design. London, Report of Royal Commission - HMSO.

Gero, J. S. (1998). Towards a model of designing which includes its situatedness. Universal Design Theory. H. Grabowski, S. Rude and G. Grein. Aachen, Germany, Shaker Verlag: 47-56.

Goel, V. (1995). Sketches of thought. Cambridge, MA, The MIT Press.

Hohpe, G. and B. Woof (2003). Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions.

Kalakota, R. S. and A. B. Whinston (1993). "The future if information systems: Leadership through enterprise integration. ." Journal of Information Systems.

Lawson, B. (1997). How Designers Think, Architectural Press.

Lee, J., K. Siau, et al. (2003). "Enterprise Integration with ERP and EAI." Communications of the ACM **Vol. 46**(No.2): 54-60.

Linthicum, D. S. (1999). Enterprise Application Integration, Addison Wesley.

Minneman, S. L. (1991). The Social Construction of a Technical Reality: empirical studies of group engineering design practice. Department of Mechanical Engineering, Stanford University.

Pahl, G. and W. Beitz (1988). Engineering design: A systematic approach. London, Design Council/Springer-Verlag.

Reiersgaard, N., H. Salvesen, et al. (2005). EAI Implementation Project and Shakedown: An Exploratory Case Study. 38th Hawaii International Conference on System Sciences, Hawaii.

Salmela, H., A. L. Lederer, et al. (2000). "Information systems planning in turbulent environment." Journal of Information Systems: 3-15.

Savransky, S. (2000). Engineering of Creativity. Boca Raton, Florida, CRC Press.

Simon, H. (1969). The sciences of the artificial. Cambridge, MA, MIT Press.

Sutherland, J. and J. W. Heuvel (2002). "Enterprise application integration and complex adaptive systems." Communications of the ACM **45**(10): 59-64.

Tovey, M. (1997). "Styling and design: intuition and analysis in industrial design." Design Studies **18**(1): 5-31.

Ullman, D. G. (2003). The Mechanical Design Process, Elizabeth A. Jones.

Whiteside, R. A., E. J. Friedman-Hilll, et al. (1998). "PRE. A framework for enterprise integration." DOE Scientific and Technical Information.

Yang, K. and H. Zhang (2002). A comparison of TRIZ and Axiomatic design. First International Conference on Axiomatic Design, Cambridge, MA.

Zeng, Y. (2001). An Axiomatic Approach to the Modeling of Conceptual Product Design Using Set Theory. Department of Mechanical and Manufacturing Engineering. Calgary, University of Calgary.

Zeng, Y. (2002). "Axiomatic theory of design modeling." Transaction of SDPS: Journal of Integrated Design and Process Science **6**(3): 1-28.

Zeng, Y. (2004a). "Environment-based formulation of design problem." Transaction of SDPS: Journal of Integrated Design and Process Science **8**(4): 45-63.

Zeng, Y. (2004b). Environment-based design: process model. Montreal, Concordia Institute for Information Systems Engineering, Concordia University
40.

Zeng, Y. and B. Chen (2005). Component-Based Conceptual Design Through Environment Decomposition. Integrated Design and Process Technology, Beijing, China.

Zeng, Y. and G. Cheng (1991). "On the logic of design." Design Studies **12**(3): 137-141.

Zeng, Y. and P. Gu (1999). General Design Governing Equation. NSFC Grantee's Conference on Design and Manufacturing Engineering, Vancouver, BC, Canada.

Zeng, Y. and P. Gu (1999a). "A science-based approach to product design theory Part I: Formulation and formalization of design process." Robotics and Computer-Integrated Manufacturing **15**(4): 331-339.

Zhang, J., F. T. S. Chan, et al. (2003). "Agent- and CORBA-Based Application Integration Platform for an Agile Manufacturing Environment." Journal of Advanced Manufacturing Technology **21**(6): 460-468.

Zhang, K. Y. a. H. (Jun. 2002). A comparison of TRIA and Axiomatic design. First International Conference on Axiomatic Design, Cambridge, MA.