JIAN SUN

A Framework for Supporting Generative Product Design Using Genetic Algorithms

2002

PhD

1999–2020 THESIS SHOWCASE

This thesis investigates the application of generative and evolutionary techniques in computer-aided product design, aiming to create a computer model of a generative evolutionary product design process for conceptual design. This research focuses on developing a computer model of a generative and evolutionary design. A prototype system is implemented to support designers at the conceptual design stage. The study addresses several significant issues through the implementation of a generative and evolutionary system in product design, which includes: (1) the identification of two cyclically linked stages in the design process as formative construction and design development, (2) the implementation of a prototype which tests the effectiveness of the data structures and the genetic algorithms, and (3) the computational methods and representations of applying genetic algorithms to product design. The result shows the capability to generate a wide range of solutions based on initial design requirements specified by designers, and with the support of genetic algorithms and the knowledge formulated as the concepts of the rudiments and formatives.

A Framework for Supporting Generative Product Design Using Genetic Algorithms

By

Jian Sun B.Sc., M.Sc.

Sept. 2002

School of Design

The Hong Kong Polytechnic University

Abstract of thesis entitled 'A Framework for Supporting Generative Product Design Using Genetic Algorithms'

submitted by Jian Sun

for the degree of Doctor of Philosophy

at The HongKong Polytechnic University in Sept., 2001.

ABSTRACT

This thesis investigates the application of generative and evolutionary techniques in computer aided product design. It aims at creating a computer model of a generative evolutionary product design process for conceptual design. This aim is achieved by enhancing the creativity role of designers through the application of genetic algorithms in a computer-based support system.

The development of a computer model of a generative and evolutionary design, and the implementation of a prototype system to support designers at the conceptual design stage using this model are the main focuses of this research. A number of significant issues are addressed through the implementation of a generative and evolutionary system in the domain of product design.

First, two cyclically linked stages are identified in the design process as formative construction and design development. Genetic algorithms are used to support designers at these two stages for the exploration of design solutions. New data structures termed Rudiment and Formative are introduced to represent design information and knowledge that can be selected and manipulated by designers.

Second, a prototype system is implemented to test the effectiveness of the data structures and the genetic algorithms used. This system has a database of rudiments, an interface for building the formatives, and a support program employing genetic algorithms to support designers in the exploration and evaluation of a large number of design solutions in an interactive manner. The system is integrated with a commercial CAD system for 3D visualisation.

Third, computational methods and representations are worked out to apply genetic algorithms to product design in this generative and evolutionary model. This includes a hierarchical representation of functional components of products that can be associated with and manipulated by genetic algorithms. Also, multi-objective evaluation and selection are investigated to allow users to specify initial design requirements in terms of Design for Manufacturing (DFM) considerations and constraints. The integration of DFM constraints into the database and the evaluation step of the genetic algorithm provided insights on how general DFM issues can be incorporated at the initial stages of conceptual design. The purpose is to produce products configured with such components and in such a way that, when detail design occurs, easily manufacturable parts can be realised.

In this thesis, consumer product design has been selected as an example domain for the implementation and evaluation of the computational model and the system. Testing examples with mobile phones, remote controllers and handheld products are provided in the thesis. The system developed in this thesis has shown a capability in generating a wide range of solutions based on initial design requirements specified by designers, with the support of genetic algorithms and the knowledge formulated as the concepts of the rudiments and formatives.

Finally, the limitations of the approach, and the way in which the developed system can be further improved are discussed.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. John Frazer, for his continuous encouragement, support, guidance and patience. I never would have completed this without his ideas, advice and support. I would also like to thank my co-supervisor, Dr. MingXi Tang for his enthusiasm, his constructive advice, and his constant support for my study. I would also like to thank Dr. Peter Bentley, who provided me with a number of his research papers and many valuable suggestions.

Also, I would like to acknowledge the support from academic and research staff in the School of Design in the Hong Kong Polytechnic University. It is impossible to recognise all of those there who contribute in one way or another to my progress at Hong Kong Polytechnic University and to this thesis, but certainly the names of Benny Leung, Cristiano Ceccato, and Norris Wang. Thanks also to all faculty of the Design Technology Research Centre, who have been great colleagues and friends throughout my entire stay in DTRC and contribute to my progress in the graduate school and to this thesis in one way or another. Especial thanks to the colleagues, Mr. KwaiHung Chan, Mr. Patrick Janssen, Mr. ZhenYu Gu and Dr. QianPing Wang, the support and friendship of them have been a source of generosity and personal inspiration and the discussion with them often helped me a lot in my work and in difficulties.

Last, but not least, I would like to express my deepest gratitude to my family. I am indebted to my parents, sister and brother for their love and absolute confidence. My family has been always a great resource of support in my life in general and in my work in particular. I want to thank them for their love and their continuous unwavering support. Most of my successes in school over the years are due to my parents, who taught me the right priorities and always encouraged me to learn and work hard.

CONTENTS

Contents *Page*

ABBREVIATIONS

3D	Three Dimensional
AI	Artificial Intelligence
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAE	Computer Aided Engineering
DFM	Design For Manufacturability
ES	Expert System
ET	Evolutionary Techniques
GA	Genetic Algorithm
GAs	Genetic Algorithms
GUI	Graphical User Interface
KBS	Knowledge Based System
SGA	Simple Genetic Algorithm

FIGURE INDEX

NOTES ON ACCESS TO CONTENTS

Chapter 1

1. Introduction

Future design implies the use of integrated Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) systems, but with a higher level of intelligent and automatic supporting capability for the whole design process. This kind of research is now conducted at the Design Technology Research Centre (DTRC), which was established at The Hong Kong Polytechnic University in 1996. The DTRC aims at developing, demonstrating, validating, and disseminating computational design methods and for the design of technical systems. Its main objective is to promote new thinking in the research and development of advanced computer-based design tools through a theoretical and critical framework for the communication and understanding of the emerging new design philosophy and the scholarly analysis and criticism of these developments.

Studying generative and evolutionary design systems is a major research theme of the DTRC. The research work presented in this thesis relates to a sub-project of this research theme. This research is an attempt to create a computer model of a generative and evolutionary design process in the domain of product design, and then to develop a computer based tool to support designers based on this model. This research is motivated by the need for supporting designers in design concept exploration, and for the enhancement of the generative capability of any computer based design systems including commercial CAD/CAM systems.

1.1. Problem Identification

Recent advancement in evolutionary computing provides new opportunities to re-examine the issue of intelligent design support. Evolutionary Algorithms (EAs) have been widely accepted to be appropriate for exploring and evolving design problems. There are four main reasons why evolutionary techniques are used in this study.

First, evolutionary techniques have demonstrated to be good, general-purpose problem solvers for design problems. For example, genetic algorithms can be used for solving problems which are not yet fully characterised or too complex to allow full characterisation, but for which analytical evaluation is possible. That is, problems for which we do not necessarily know exactly how to find good solutions, but we can evaluate candidate solutions by some quantifiable measures.

Second, evolutionary algorithms take natural evolution as a source of inspiration, so the exploration and adaptation ability during this process is valuable and useful for design concept exploration at the early stage of the design process. "Natural evolution has been creating designs successfully for an unimaginable number of years. Even a cursory study of the myriad of extraordinary designs in nature should be sufficient to inspire awe in the power of evolution." (Bentley, 1996)

Third, evolutionary algorithms have been successfully applied to architecture design, engineering design and other optimisation problems, while its application in product design is still limited. With the introduction of evolution into the product design process, a different kind of computer-aided design systems better than the existing traditional CAD systems can be developed. In such systems, the generative capability can be added and Artificial Intelligence techniques can be integrated into the product design and development process. Such systems will enable the automatic generation of large numbers of alternative design concepts or forms, rather than a single individual as in traditional computer aided design systems, in a controlled and supervised manner based on primitive components and axioms of design.

The existing computer-aided design technology provides limited support for the evolution of design solutions in the early conceptual phases of product design process. There are three main reasons found:

First, CAD technology relies on precise geometric information to specify the model of a product. It cannot support early stage design tasks because the geometry of a product is not sufficiently definite at the early stage. The design of complex products containing many components requires that alternative solutions be explored, each being modified by making changes to key design parameters, until a satisfactory design solution is found. The generation and exploration of design concepts based on incomplete design requirements, and the handling of a multitude of variables and constraints embedded within 3D design objects are challenging tasks for designers.

Second, human design activities cannot generally be simulated or modelled using computers naturally and completely. Research in Artificial Intelligence has resulted in principles and methods for supporting conceptual design based on the techniques of Expert Systems or Knowledge Based Systems. But these principles, methods, and systems still remain experimental.

Third, it is difficult to model conceptual design process in a computer based system as the process and the information leading to the creation of an innovative design concept are still difficult to acquire, to represent and to reason about in a computer based system.

Many researches are now trying to address these issues. One of the latest and the most exciting methods is employing computers as creative problem solvers using evolution to explore new solutions with the application of evolutionary computation techniques. Evolutionary computation techniques, especially Genetic Algorithms, have been applied to engineering design optimisation, constraint satisfaction, symbolic equation solving, and manufacturing process planning etc. These adaptive and generative techniques provide more creative and intelligent support to designers than other design support systems that rely only on geometric representations and explicit deductive inference mechanisms. As stated by Bentley (Bentley 1996 and 1999), the latest researches have already shown impressive results in the creation of novel designs in architecture and art.

However, although there have been many investigations into the application of evolutionary techniques in engineering problems during last decade, the application of evolutionary techniques to product design is still limited. The difficulty arises from the fact that product design requires a wide variety of knowledge and information at the early creative stage of the design process that is difficult to formulate in a computer system. This kind of knowledge and information are related to how to determine the function, form, material, human factors, ergonomics, and environmental impact of a product. In particular, the form, styling and the configuration of a product have important implications for manufacturing processes and production cost. A product that requires high capital investment in manufacture or that requires a complex manufacturing process tends to be resistant to competition. As a result of a lack of computer aided industrial design tools, the development of an innovative product is still a lengthy process.

Furthermore, design and manufacturing have become an integrated process as a result of using advanced Computer Aided Design/ Computer Aided Manufacture/ Computer Aided Engineering (CAD/CAM/CAE) techniques. This introduced the concept of Design For Manufacturability (DFM). The DFM approach aims at improving all the processes related to the production of a product, ranging from conceptual design to manufacturing, which means that manufacturing issues are considered throughout the whole design process. These ideas have led to the development of methods and computer systems that can help designers to design products that are functionally correct and at the same time easy to manufacture. Until now, much effort has focused on the detailed manufacturing constraints solving or optimisation. But few techniques have been developed to integrate the DFM considerations into the early design concept generation and exploration stage in a generative and evolutionary product design process, and make them effective throughout the whole process. That is why it is selected as one of the focuses of this study.

How to model shape design as an evolutionary process and then use computational representations and inference mechanisms to resolve shape constraints imposed by manufacturing process are two important issues in the paradigm of generative and evolutionary design. This study focuses on modelling design process as a generative and evolutionary process and resolving shape constraints using a process based on an integration of evolutionary algorithms with the consideration of manufacturing issues.

1.2. Research Objectives

The main aim of this research is to explore the applicability of evolutionary computing techniques to product design process, which takes into account the constraints imposed by the manufacturing process. This leads to the investigation of how Genetic Algorithms can be applied to product design and the development of a computational system prototype. This system models product design process as a generative and evolutionary process, and it is capable of evolving product design with manufacturing considerations.

The objectives of this study can be highlighted as

1. Establishment of a generative and evolutionary product design process,
2. Development of a representation for 3D product data models that can be manipulated in a system using Genetic Algorithms,
3. Implementation of a generative and evolutionary design system capable of producing design solutions from abstract design specifications and design intent of the users,
4. Study of an interface in which designers can be interactively supported by a generative and evolutionary design system,
5. Identification of a way in which the exploration of design alternatives can be guided either by the selection based on evaluation rules or criteria, or the selection decided by the users from a design for manufacturing point of view.

These objectives are further explained below:

Objective 1: *Establishment of a generative and evolutionary product design process.* This is to propose a generative and evolutionary design process model upon which computer-based techniques such as Genetic Algorithms can be employed to support designers in a user-friendly and interactive manner. In this model, design is described as a process consisting mainly of two cyclically linked processes, referred to here as Formative generating process and Formative developmental process. In this model the concepts of "rudiment" and "formative" are introduced. In general, a formative is an encapsulated design schema, which defines a set of constitutive elements with relations to other such elements, as well as generative rules to be used during the design schema generating process. A rudiment is a composition part of a formative, which defines each constitutive element with specific knowledge of a domain. A more detailed description of the formative and the rudiment will be given in Chapter 4 later.

Objective 2: *Development of a representation for 3D product data models that can be manipulated in a system using Genetic Algorithms*. This is to develop a concise representation of product design based on the functional decomposition of a product using the definition of rudiment and formative. A key issue in this representation is the integration of functional description, parametric relations, 3D geometric and spatial data, evaluation and selection rules, and criteria for design for manufacturing considerations.

Objective 3: *Implementation of a generative and evolutionary design system capable of producing design solutions from abstract design specifications and design intent of the users*. This is to implement a generative and evolutionary design system based on the model, and the representation proposed in this study. In the implemented system, a user can specify design requirement or design intent through a user interface. Then the system can generate a population of design alternatives. The templates for these design requirements or design intents have been formalised and stored in advance in a database during the system development period. These can be used for the user to form initial populations of design proposals for the system to generate more alternatives generation after generation automatically. The process of evolving alternative design solutions can be controlled by a user, or by using a fitness function in the evaluation program.

Objective 4: *Study of an interface in which designers can be interactively supported by a generative and evolutionary design system.* This is to ensure that the system developed in this study is a generic design support system, which means that it is able to evolve design solutions to a range of product design problems. The system has been implemented in a way in which it can perform well in several types of design applications and is scalable to larger and more complex design problems. However, in order to obtain the evaluation results on the capability of the system in supporting designers, it has so far focused on the domain of consumer electronic product design.

Objective 5: *Identification of a way in which the exploration of design alternatives can be guided either by the selection based on evaluation rules or pre-defined criterion, or the selection decided by the users from a design for manufacturing point of view.* This is to ensure that new designs are easy for a user to specify, with the minimum requirement for additional evaluation criteria. In this system a functional product decomposition method is used so that a design problem can be specified by a user by selecting a set of existing functional product components. Then the associated evaluation software or sub-evaluation functions can be activated to guide the system in the evolution process for generating and evaluating alternative design solutions, during which the adjustment on the control parameters can be avoided. This reduced the need for frequent user intervention. The system is also able to evolve new designs guided by the selection made by the user. The selection made by user in an evolutionary design system is a useful means for dealing with ill-defined selection criteria, particularly user centred concerns. It also provides an opportunity for designers to use their own experience and intuition during design exploration.

Additional considerations for the system developed in this study included user friendliness and the potential for integration with other AI-based systems in a more general framework. Since the system developed in this study aims at providing intelligent support to designers, the prototype model of the design process built in this system must be designer centred. The system interface created is not trying to mimic the behaviour of designers, but to support the application of their design knowledge and allow them to use their own intuition. These are some essential factors which are considered essential by designers during the design process. The software system developed in this study provided a human-computer interaction interface, which can be further extended by users.

These objectives are centred around one goal, that is to produce a design system capable of evolving useful and innovative solutions to real world design problems. The solutions generated by the system developed in this study are not necessarily the final designs, but they establish essential form features and configuration parameter values of a product prototype and many alternatives of such a prototype automatically or interactively. These product prototypes can be easily visualised and evaluated, either for further refinement using other CAD tools, or used directly by designers for more detailed analysis and development.

1.3. Research Approach and Methodology

The research work during the development of the system has gone through the following steps:

- An understanding of the principles of Genetic Algorithms (GA) and development of GA programs based on simple exercises and examples,
- An analysis and synthesis on three dimensional (3D) product data models in order to choose an appropriate representation method and genetic operators which can be manipulated in a system using GA,
- An investigation of computational modelling of product design process involving the application of generative and evolutionary techniques,
- An implementation of a prototype system including database, main generative and evolutionary support tools, user interface and an integration with other CAD tools,
- An experiment on the developed system using the examples in remote controller and mobile phone design,
- Testing, visualisation, evaluation and rapid prototyping of some of the results generated by the system, and
- Further refinement of the results.

The main problems addressed in this research included:

- To identify product features associated with some general manufacturing guidelines and DFM considerations in the application domain concerned. These encompass the decomposition of product structure, the analysis of the taxonomy of product forms with related DFM issues,
- To establish a representation scheme and data structure with which a product design model can be represented as binary strings. A structured representational mapping from a 3D solid to a GA coding scheme had to be established as the existing ways of representing product shapes in both research software systems and commercial CAD/CAM systems provided no solution for this,

- To study and evaluate a number of Genetic Algorithms suitable for supporting the generation of product forms easy for manufacturing,
- To develop and implement a system prototype of the generative and evolutionary product design system, which integrates Genetic Algorithms with a design process using an interactive user interface to allow users or designers to supply requirement information, evaluation criteria or selection preference at the beginning of the design exploration stage,
- To find out the effectiveness of the model, the representation, and the system developed in this study in a concurrent engineering context,
- To integrate the developed system with a commercially available CAD system, the MicroStation/J of Bentley System Inc., so that the system developed can be used as an add-on tool to support designers in a wider spectrum.

1.4. Significance and Potential Benefits

The research presented in this thesis produced satisfactory results with original approaches, models, representations, implementations and testing cases addressing an important research problem and real world applications. Three features of the developed system highlight the significance and the main contributions of this work:

First, the computational model developed in this study models the design process as a generative and evolutionary process. This is not just to reflect what is going on inside a designer's mind, or purely modelling the design process as it is now. It tries to provide a basis for the development of a supporting environment in which a designer's ability in generating design concepts is enhanced. And the design results generated are not single solutions to a well-defined design problem but a population of design alternatives demanding less strict definitions or specifications. The proposed designs are not obtained through a deliberate attempt to produce them, but by generating a wide range of varieties and then focusing on the most promising ones, which are expected by or innovative to the designer. This means on one hand, the design results can be converged to the designer's requirements that are predefined as the evaluation function of the genetic algorithm, on the other hand, the results may diverge from the designers' expectation due to the randomly generation and selection by the system.

Second, by tackling the issue of modelling 3D forms of design at the early stage of the design process using evolutionary techniques, this study addresses an important issue that links design and manufacturing process. This involved a mapping from a 3D computer representation of a design object to a binary string manipulated by a GA program. Building taxonomy of primitive forms, considering manufacturing and other constraints in the evaluation process, and modelling the process within a computer supported environment with this mapping in the context provided knowledge and insights on how to develop more intelligent and supportive design tools.

Third, in this study the concept of the Design For Manufacturability is introduced at the initial stages of conceptual design. The goal is to produce a configuration in such a way that when it is developed further in detail, a manufacturable part can be realised. In the system developed in this study, a generative design process followed common manufacturability guidelines and the rules for an early anticipation of manufacturing problems to avoid unnecessary changes or revision to the concepts downstream in the detailed design process.

1.5. Overview of the Thesis

This thesis is divided into eight chapters. Following this chapter, a research background is provided in Chapter 2, which reviews the relevant and related work to the research work presented in this thesis.

A model for an automated generative and evolutionary product design system is proposed in Chapter 3. In this chapter, the model and its general design context are described. Two examples of generative product design using Genetic Algorithms are also presented.

Chapter 4 and 5 describe the fundamental elements of the developed system. These are the design representation and the associated elements of Genetic Algorithms for generating design solutions, which included the design representation, encoding of the design, genetic operators and evaluation and selection of solutions.

Chapter 6 presents the implemented system developed based on the model presented in Chapter 3, as a general framework of a generative and evolutionary system in the domain of product design.

Results from the implementation demonstrating the ability of the model to produce automated generative and evolutionary designs with DFM consideration are reported and analysed in Chapter 7. Several case studies are presented.

Finally, in Chapter 8 conclusions about the feasibility of the developed system and its limitations are drawn based on the results presented in Chapter 7. Considerations for future work in this direction are also discussed.

Chapter 2

2. Review of Related Work

2.1. Introduction

The research work described in this thesis is an attempt to create a computer model of a generative and evolutionary product design process. This model supports designers in design concept exploration and provides enhancement for creativity in a computer based design support environment. This work defines an emerging research area, which integrates evolutionary computing techniques with knowledge in computer aided product design area. This chapter reviews and assesses the related significant research in these areas.

The scope of this thesis covers the following areas of research:

- Design and design process,
- Computer aided product design process,
- Design representation methods,
- Design For Manufacturability,
- Genetic Algorithms,
- Evolutionary design,
- Generative evolutionary design process.

This chapter gives a general review of the work in these areas. And then it compares the differences between these researches with the proposed work. Further references to literature with regard to more detailed problems encountered during the development of the system will be made in later chapters when these problems are discussed in detail.

2.2. Design and Design Process

2.2.1. Theory Overview

The definition of design or design process is widely interpreted and argued. Christopher Jones provides a useful overview, in his book *Design Methods* (Jones, 1980):

"Design is about 'Finding the right physical components of a physical structure' (Alexander 1963). Design is 'A goal-directed problem-solving activity' (Archer, 1965). Design is about 'Decision making, in the fact of uncertainty, with high penalties for error' (Asimow, 1962). Design involves 'Simulating what we want to make or do before we make or do it as many times as may be necessary to feel confident in the final result' (Booker, 1964). Design is related to 'The condition factor for those parts of the product which come into contact with people' (Farr, 1966). 'Engineering design is the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform prespecified functions with the maximum economy and efficiency' (Fielden, 1963). Design is 'Relating product with situation to give satisfaction' (Gregory, 1966a). Design is 'The performing of a very complicated act of faith' (Jones, 1966a). Design is to find 'The optimum solution to the sum of the true needs of a particular set of circumstances' (Matchett, 1968). Design is 'The imaginative jump from present facts to future possibilities' (Page, 1966). Design is 'A creative activity – it involves bringing into being something new and useful that has not existed previously' (Reswick, 1965)".

In other definitions, design is described as a process of transforming a set of functional specifications and requirements into a complete description of a physical or tangible product or system, which meets those specifications and requirements. During the transformation process, the designer makes decisions about function, shape, material properties, manufacturing technologies etc., based on the information or knowledge provided by handbooks, standards, numeric analyses, company practices, rules of thumb and personal intuition and experience. Tomiyama et al., regard design as a theorem solving process in their extended General Design Theory (Tomiyama et al, 1987), where design is viewed as a mapping from a functional space to an attribute space. A description of the design process from a computational perspective is given by Dr. Vasant Honavar, "The design process, from a computational perspective, essentially involves searching a design space...... Design is a cognitive activity that all humans, many animals, and to a limited extent, intelligent machines, engage in. Given the ubiquitous nature of design activity, it appears highly improbable that the underlying cognitive processes involved in different domains are fundamentally different. Since computation offers one of the most useful metaphors for modelling and even engineering cognitive systems, it is only logical that exploration of design as a cognitive process should start with a computational characterisation of the activities involved. Some natural implications of this position lead us to explore design in terms of computational and information processing processes involved in search, decision-making, and distributed problem solving." (Honavar, 1997).

Although most of the definitions above viewed design process as a problem solving process, in this research project design process is described from a different point of view, as a problem finding process. This can be quite explained through these words: "Design as seen from the designer's perspective is a series of amazing imaginative jumps or creative leaps. But design as seen by the design historian is a smooth progression or evolution of ideas that seem inevitable with hindsight. It is a characteristic of great ideas that they seem self evident and inevitable after the event. But the next step is anything but obvious for the artist/creator/inventor/designer stuck at that point just before the creative leap. They know where they have come from and have a general sense of where they are going, but often do not have a precise target or goal. This is why it is misleading to talk of design as a problem solving activity – it is better defined as a problem finding activity. This has been very frustrating for those trying to assist the design process with computer based, problem-solving techniques. By the time the problem has been defined it has been solved. Indeed the solution is often the very definition of the problem." (Frazer, 2001)

2.2.2. Product Design Process

Design is vital to a manufacturing company's goal of creating successful products. Necessarily it is a systematic activity which covers a wide range of expertise, from identification of the market/user needs, to selling of the successful products to satisfy that needs - an activity that encompasses product, process, people and organisation (Helander, 1992). Although the precise definition of design is often argued, there is perhaps more agreement concerning the functional composition of design. In his book on the product design, Mike Baxter gives a comprehensive overview of design process (Baxter, 1995). The development of a relatively simple product is shown schematically in figure 2.1. The design activities involved at each step differentiate the stages of product development process. Basically, a total design process is split into five main phases as design specification, conceptual design, embodiment design, detailed design and manufacture.

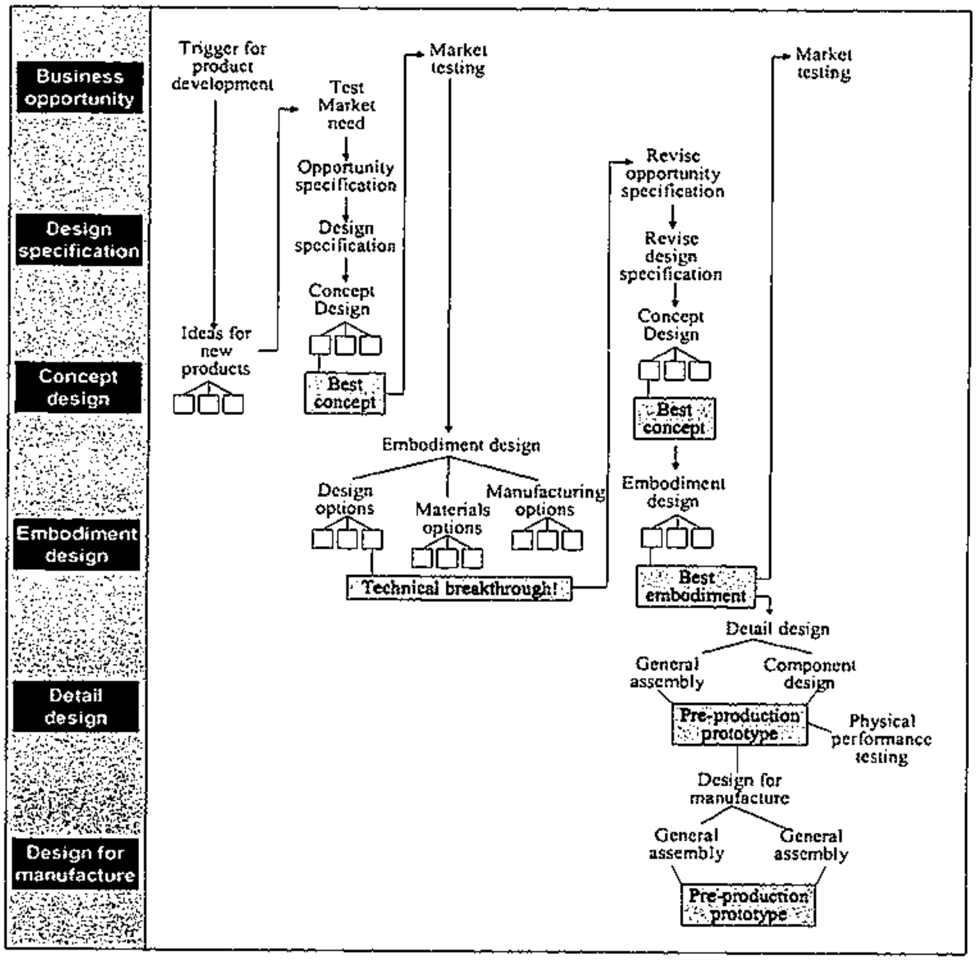


Fig 2.1. A product design process. (redrawn from Product Design by Mike Baxter)

To date, computers have been successfully used for all these stages except for the first: conceptual or creative design (Dym and Levitt, 1991, Goldberg, 1991). Goldberg stated that: "The creative processes of engineering design have long been regarded as a black art. While the engine of analysis steamrolls ever forward, our understanding of conceptual design seems locked in a timewarp of platitudes, vague design procedures, and problem-specific design rules". (Goldberg, 1991). And a definition of conceptual design given by Brown (Brown, 2000) is claimed as: conceptual design takes the statement of the design problem and generates broad solutions. In general, the conceptual design encompasses all the tasks necessary to develop a concept to proceed with detail development, prototype and analysis. This includes a first pass definition of requirements and specifications, layout different concepts, concept generation, exploration of available design solutions, information gathering efforts and concept evaluation.

2.2.3. Classification of Product Design Processes

The design process is a complex and not yet well-understood cognitive process conducted by humans, which is related to the process of actions and decisions taken during design in order to arrive at a complete product design. In order to make a more clarifying and formalised description of design process, a classification on the existing design processes is carried out here. This classification looks at recent studies of managing product design and the development based on the synthesis and analysis of the existing researches. Considering different design methods and strategies employed, design process can be formalised in different ways (Jones, 1980. Baxter, 1995). Typically:

- Linear process,
- Cyclic process,
- Branching process, and
- Parallel process.

Linear process: This approach describes the traditional sequential pattern of the product development process in which design process consists of a sequence of actions. Each action is dependent upon the output of the last but must be independent of the output of later stages, as shown in figure 2.2 below. In this approach, each department makes its contribution before passing the project 'over the wall' to the next department.

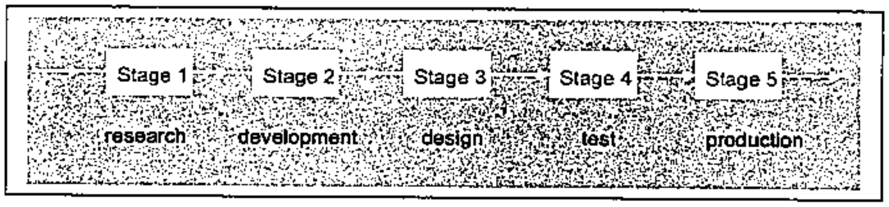


Fig 2.2. The linear design process.(based on the description in Design Methods, Christopher Jones. 1980)

Cyclic process: As shown in Figure 2.3, when an earlier stage of task has to be repeated for correction after the output of a later stage of task becomes known, then a cyclic process is adopted. In this approach, there are a series of iterative loops during which a design may be tested and modified several times before being passed to the next department.

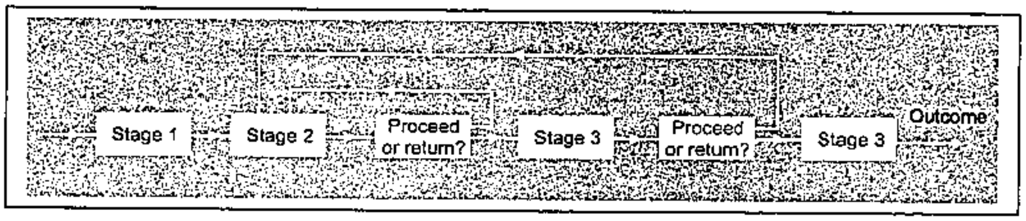


Fig 2.3. The cyclic design process. (based on the description in Design Methods, Christopher Jones. 1980)

Branching process: This is possible when design actions are wholly independent of each other. Figure 2.4 gives a schematic description on this process. In the figure, the approach can include parallel stages, which have the great advantage of increasing the number of persons working on a problem at one time, or alternative stages, which allow the adaptation of strategies according to the outcome of previous stage.

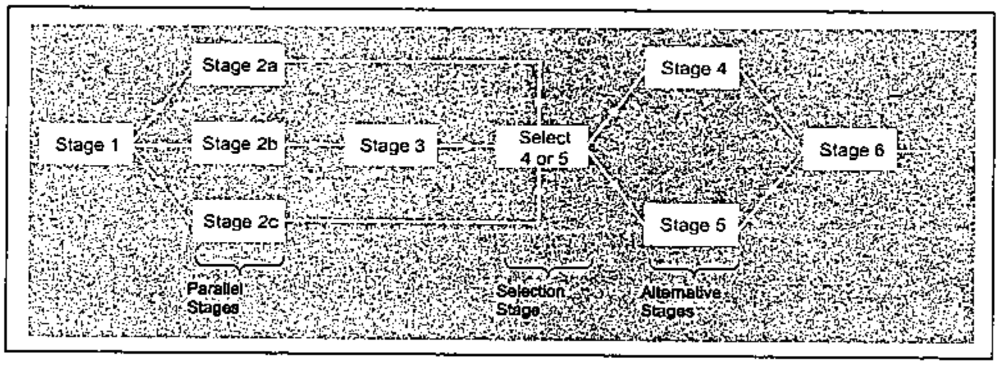


Fig 2.4. The branching design process. (base on the description in Design Methods, Christopher Jones. 1980)

Parallel process: In this approach, design is undertaken by a multidisciplinary project team working simultaneously and in regular communication with each other, as illustrated schematically in figure 2.5. And in engineering design, it is termed Concurrent Engineering. This basically refers to organisational structures that attempt to break down the functional barriers so that designers and production staff are all on the same side working towards the same goal with good communications between them.

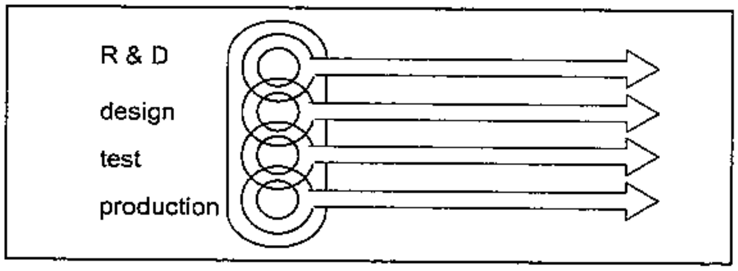


Fig 2.5. The parallel design process. (Based on the description in Design Methods, Christopher Jones. 1980).

In this study, the design process involved is based on this parallel design process concept, which integrates various stages of total design process at the start point and focuses on the conceptual design stage. As in most cases, to design a product through the process of total design is a tedious task. In order to speed up the process, add intelligence to the development, and hence to reduce time to market, it is reasonable and significant to integrate various stages in total design process at the early stage, including conceptual design, detail design and manufacture. There are still other formalised design process types based on different design methods or strategies (Hawkes, 1984, Medland, 1986 and Pugh, 1991.), but most of them can be classified into these four types outlined above. And the parallel design process is the most active one in recent summaries.

2.3. Computer Aided Product Design Process

2.3.1. A Brief Historical Overview of CAD

The use of the CAD technologies in industry has greatly improved the quality of products, raised the efficiency, and reduced the cost and lead-time of product design. Today, the CAD technology is playing an important role in all domains. Figure 2.6 shows an implementation of a typical CAD procedure in a CAD system.

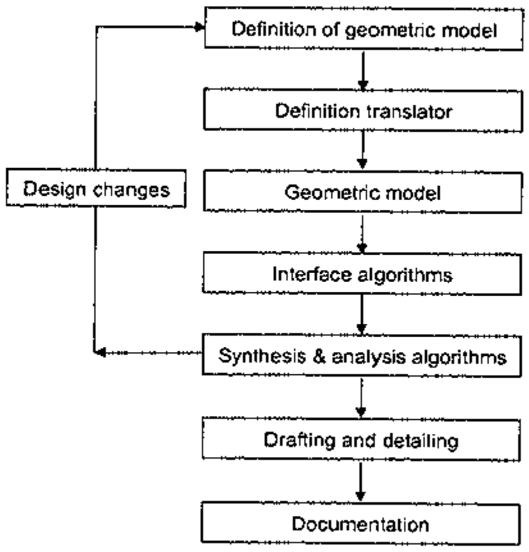


Fig 2.6. Implementation of a typical procedure in a CAD system.
(based on the proposed CAD procedure in Medland, 1986.)

The development of Computer Aided Design (CAD) technology has gone through four major phases in the past. The first phase spanned the decade of the 1950s and can be characterised as the era of introducing interactive computer graphics. The first graphic system was developed in mid 1950. Then during the decade of the 1960s, the term "Computer Aided Design" (CAD) emerged. In the decade of the 1970s, the research efforts of the 1960s in Computer Graphics had begun to be fruitful and the important potential of interactive computer graphics in improving productivity was realised by industry, government, and academics. Some automated design/drafting systems were developed in early 70s and were improved and further developed. The decade of the 1980s can be characterised as the CAD heady years of research. In this period CAD systems were expanded beyond three-dimensional geometric design and provided functions for more practical engineering applications, such as the accurate representations of sculptured surface, mechanism modelling and simulation, robotics analysis and simulation, injection moulding design and analysis, and front-end tools to automate conceptual design. Currently, some technologies, such as feature modelling, have been developed for integrated and automate design and manufacturing.

The undisputed popularity and overwhelming adoption of CAD technology by industry can be undoubtedly attributed to many of its capabilities, which offer significant advantages over the traditional design methods. The main advantages of CAD can be identified: CAD can produce graphical representations of objects with highly complex geometries. It provides designers with the ability to specify more accurately the necessary clearance and tolerance of these objects. CAD also has the ability to perform sophisticated engineering analysis by such powerful techniques as the finite element analysis method and to present the results in comprehensive graphical forms. Many commercial CAD packages can perform various degrees of design optimisation involving hundreds of design variables and constraints, such as weight, volume, space, material specification and costs. Other optimisations, such as optimisation of the methods of production, can also be tackled.

Substantial benefits to designers and engineers are provided by applying the CAD technology in industry. CAD can enhance the quality of engineering design with excellent graphical representation of product geometries and production drawings, and raise their efficiency with effective computer algorithms. Since most CAD systems have individual databases, they offer great flexibility and economy in the management of design data, which makes the modification of existing designs easy. With increasingly available computer power, a CAD system can solve complicated design problems involving hundreds of variables, which would be difficult or impossible if analysed manually.

Most of CAD systems today are restricted to the detail design phase in the whole design process. Computer based support to embodiment and conceptual design is still limited. The majority of CAD systems emphasise geometric modelling and product visualisation, but offer limited functions for users to explore a wide range of design issues. So Computer Aided Design is sometimes used synonymously for Computer Aided Drafting, indicating that CAD is nothing more than an extension of the traditional drawing board. To change this situation needs more understanding of design as an intelligent behaviour and how this behaviour can be enhanced.

Some other approaches have been developed to improve the functionality of a traditional CAD system. For example, Thornton stressed the fact that current CAD systems do not support embodiment design because of lack of sufficient ways for constraint satisfaction involved in embodiment design (Thornton, 1994). Erens stressed that today's CAD systems are incapable of keeping different views of the product model consistent (Erens, 1993). The views that Erens mentioned are those of sales and marketing, product engineering, assembly engineering, manufacturing logistics and service, and design management. Van Houten regarded it as a disadvantage of the current CAD systems, that a large number of and sometimes illogically organised command sets only show too much of the CAD system's internal structure but too little of the application domain (Houten, 1999). In (Shah, 1988) the general mismatch between contemporary CAD/CAM/CAE software and engineering tasks has been described.

In this thesis, it is proposed to consider computer enhanced design rather than computer-aided design in order to overcome the existing limitations and connotations of CAD. In this proposed new approach, the quality, value or extent of design must be intensified, increased, and further improved in an enhancing process. In this process, human intelligence and innovative ability and computational techniques are collaboratively and naturally integrated rather than simply aided with the useful but more limited meaning of help, assist or support for designers. And it is proposed to aim at going beyond mere assistance that relies merely on geometric representations of product data incapable of adapting, collaborating and learning. In this approach, design must be seen as part of a process and in a context and not just the description of some artefact in isolation

2.3.2. Computer Aided Product Design Process

The use of CAD technologies in product design process leads to the advent of computer aided product design. Figure.2.7a shows a typical traditional CAD process of a product design. Based on the design specification given by a designer, the parts and components are built as the combination of some primitives, which are basic design elements provided by the CAD system. The assembled product model consists of these components. This figure describes a step-by-step linear sequence of the product design. During this design process, if the user or designer wants to change a little bit of his design requirements or modify part of the developed design, he/she needs to restart the whole developing process from the very beginning.

The introduction of parametric design has improved this problem partially, as shown in Figure 2.7b. Here, varied components can be obtained through the adjustment of the determined parameters separately, and meanwhile, varied designs can be generated based on different chosen components. In other words, since the design is parametric, any edition or modifications made to a drawing can be reflected in the assembly and components after regeneration, and vice versa.

Fig 2.7. (a) A traditional CAD process

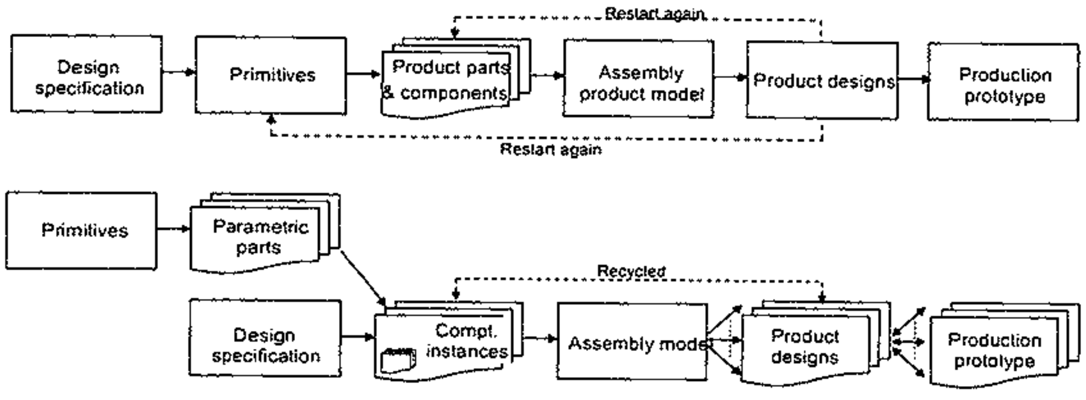


Fig 2.7. (b) A parametric CAD process

Even with this improved capability, a CAD system is still mainly used as a drafting, drawing or visualisation tool. Parametric technology still cannot provide support to concept exploration at the early stage of the design process. The generative and evolutionary design process developed in this thesis aims at overcoming this problem. The generative and evolutionary design process is based on the evolutionary development of populations of design proposals, while traditional CAD process is based on taking a single design concept for modifying and analysing as the design process progresses. The use of evolutionary computation techniques makes it possible for the computer to play a more fundamental role in the design process in which design parameters can be altered by the computer program. The computer is used in a generative and evolutionary process more than just a drafting tool or an analysing tool. The software provides more wholesome support to designers by combining, reconfiguring and modifying the key aspects of a design.

2.4. Design Representation Methods

In the last decade, the following representation methods have dominated the development of CAD systems.

- Geometric representation,
- Feature based modelling,
- Parametric technology, and
- Knowledge based representation.

2.4.1. Surface and Solid Modelling

Geometrically representing a design object and visualising it in computers for the purpose of drafting was the main aim of early CAD systems. Techniques for this purpose has been classified as wireframe modelling, surface modelling and solid modelling (Rooney, 1997). Wireframe is the simplest modelling technique and also the most commonly used technique. All the early CAD/CAM systems were wireframe-based. Surface modelling, which is more complete and less ambiguous than wireframe modelling, is an extension to wireframe modelling, with richer associated geometric information. Solid modelling, which is based on informationally complete, valid, and unambiguous representation of solid objects, is generally better than wireframe and surface modelling techniques. Solid modelling technique has been acknowledged as the technological solution to automating and integrating design and manufacturing functions.

Furthermore, there are three dominant representations in solid modelling. These are Constructive Solid Geometry (CSG), Boundary representation (Brep) and Spatial Subdivision. These three models have served as the basis for the development of several major CAD systems but their uses have been limited to drafting purposes. Since they cannot provide adequate means of dealing with functionality either at the level of representation or at the level of providing efficient mechanisms to extract and manipulate such functionality from the geometric and topological representation they use internally (Foley, 1990 and Rooney, 1997).

For product design both surface modelling and solid modelling techniques are needed in order to model and visualise the design objects generated. Many CAD systems such as ProE and MicroStation have such techniques available but combining these techniques with a representation for functional components with program controllable design variables, parameters, constraints and configuration rules is not straightforward. To develop a generative and evolutionary design system capable of reasoning about and manipulating 3D solid objects is therefore a challenge task for automating part of the product data modelling process and design concept generating process. Therefore, geometric representations, no matter how sophisticated or completed they may appear in mathematics, they are not effective in formulating the design process and supporting design tasks at a conceptual level at which detailed geometric information has not been completely defined.

2.4.2. Feature Based Modelling

In CAD systems aimed at supporting engineering design, features are considered as the smallest elements which posses explicit engineering meaning. There are numerous definitions for features in the literature. For example, Shah pointed out that "features encapsulate the engineering significance of portions of the geometry of a component or assembly, and, as such, are important in product design, product definition, and reasoning, for a variety of applications." (Shah, 1991). Wierda proposed a general definition of feature, i.e., "A feature is a partial form or a product characteristic that is considered as a unit and that has a semantic meaning in design, process planning, manufacture, cost estimation or other engineering discipline." (Wierda, 1990)

Much research has been done on the taxonomy of design features according to different design aims. A feature-based representation of a thin-walled component is shown in Figure 2.8. The set of design features for a thin-walled component can be expressed as a hierarchy in which primitives, add-ons, intersections, and macros define the top level. Primitive features are the building blocks of the component. They are connected together by intersecting features. Add-ons provide local shape modifications to the primitives. Macro features are pre-specified combinations of the other features, such as U-channels and boxes.

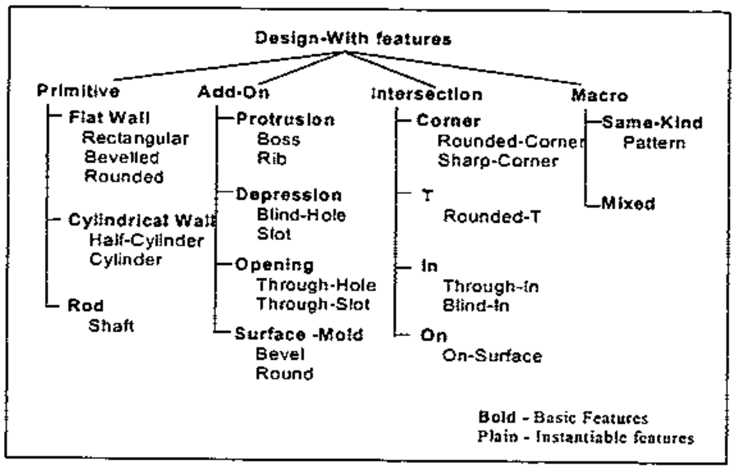


Fig 2.8. Design with features hierarchy. (redrawn from Rosen, 1993)

However in an application in a specific design domain, a strategy is needed in order to work out a feature hierarchy with which specific components can be instantiated using the feature operations available in a CAD system such as extrusion, revolving, sweeping, and a combination of these operations. In order to make use of the existing feature-based modelling techniques at a programming level for the development of a generative and evolutionary system, an additional representation is needed in order to utilise features and their operations in a hierarchical way. In this thesis, a strategy for building a feature hierarchy is to pre-specify the appropriate top levels of the hierarchy, then provide several feature classes under these levels. The detailed representation utilising features will be explained in more detail in Chapter 4.

2.4.3. Parametric Technology

In parametric design, designs are modified following the change of the values of defined parameters. Initially, parametric design referred to the use of parameters for geometric definition and it generalised its meaning gradually. Monedero presented a summary of different applications of parametric design in architecture, and he classified these applications as five types of parametric designs: Variant programming, History based constraint modellers, Variation design, Rule-based variants, and Parametric feature-based design (Monedero, 2000).

Parameters involved in the representation may not necessarily be limited to geometric variables only any more but may also include other variable features. Richard Wittenoom defines general case parametrics as "parametrics which includes but is not limited to geometrically founded parametrics and which may define abstract concepts or processes involving changes in model state and model topology"(Wittenoom, 1999). In this way, a generalised parametric representation can be obtained. And then its further improvement results in the parametric feature based design.

2.4.4. Knowledge Based Modelling

Regardless of how features are defined and how features are applied, it is important that features should capture the designer's intent and represent the engineering meaning of geometry of a component or assembly. So features are proper objects for engineering evaluation since they reflect manufacturability. Besides these, a more detailed description of a product should include information about its ergonomic and aesthetic consideration, which means that more knowledge about design data as well as design process must be involved. For this, knowledge based or rule-based representations have been developed (Smithers et al, 1990).

A representation termed design prototype was brought forward by Frazer and Gero, which brought together all the requisite knowledge appropriate to a specific design situation or a task. (Frazer, 1987 and Gero, 1990.). This approach classified design attributes into three categories and has been successfully used in many architecture design applications. These three categories are functional, behavioural, and structural. Later Maher (Maher, 1994) added an additional category, i.e., relational attribute. Functional attributes are those that represent the intended purpose of the design. Behavioural attributes represent the response of the design to its environment. Structural attributes are physical features of a design. Finally, relational attributes represent relationships among design cases, or between design cases and general concepts.

2.4.5. Limitations of the Current Representations

The above discussed ways of representing objects in CAD systems have been largely based on the need to be able to quickly translate a representation into a screen image or a drawing. They have a common limitation, that is, at present, computer modelling tends to occur after a design is substantially complete but with only minor modifications. Once the model has been loaded into the computer, there are only certain kinds of alteration that can easily be made. Despite the advantages claimed by CAD system providers, the truth is that it is generally not easy to make changes, at least not of the kind that would help to develop alternatives strategies or alternative solutions.

Furthermore, the features available in present CAD systems are usually predefined within the systems, allowing the end user only to change the parameters of the features, as in parametric feature-based design. These parametrically modifiable features are sometimes referred to as user defined features. The term user-defined features is somewhat confusing as only the feature geometry can be user defined, and not the topology and other non-geometry related characteristics of the feature. For a lot of applications it is, however, necessary to define one's own application dependent features. These features should be application specific both geometrically and topologically, including the non-shape related aspects. And now the object-oriented technology provides the possibility of programming new features, but it can also be a tedious job although the possibilities of feature definition are generally better than in most traditional CAD systems.

2.5. Design For Manufacturability

2.5.1. The Concept of DFM

It is a commonly accepted view that the design of any commercial product is a compromise between conflicting goals. The most common and important conflict is the one between the cost of meeting the customers' various performance criteria, the price that customers are willing to pay for that performance, and the price of rival products (Baxter, 1995). The two phases in the product life cycle where most of the costs are incurred are in the development phase and volume manufacture phase. Both of these must be controlled in order to reduce costs. However, it is during the design phase that the basic costs are determined, by good or bad design decisions. It is in the manufacturing process that the possible cost savings due to good design can be realised.

Traditionally, design and manufacturing activities took place sequentially rather than concurrently or simultaneously. This separation of design and manufacturing usually produces manufactured goods with higher production costs than necessary. In this approach manufacturing engineers were given detailed specification of a product and were asked to make it. They often encountered difficulties because the designers or the product engineers did not anticipate the production problems. Although a complete integration of design and manufacturing has not been achieved yet, the increasing use of computers in engineering has decreased the gap between the two.

Through the use of computers, design and manufacturing information can be accessed relatively easily applied to new designs so that when a part is designed it not only provides a set of functions but also presents as few manufacturing problems as possible. This introduced the concept of "Design for Manufacturability". In this approach, manufacturing requirements are well considered in advance at the design stage. The DFM approach aims at improving all the processes related to the production of a product, ranging from conceptual design to manufacturing. These ideas have led to the development of methods and computer systems that can help designers to design products that are functionally correct and at the same time easy to manufacture.

2.5.2. DFM at Conceptual Design Stage

At the highest level of the design process, conceptual design is defined as searching across an ill-defined space of possible solutions using approximate objective functions and value concepts of the structure of the final solution. As artificial intelligence views of conceptual design, "conceptual design takes the statement of the design problem and generates broad solutions." (Brown, 2000). Also, conceptual design is called preliminary design or functional design (Baxter, 1995.). At the conceptual design stage, designers deal not only with aesthetic issues, such as styling, but also with practical issues such as simulation and industrial design for manufacturability. Conceptual design requires processing information from diverse sources in order to define the functional requirements, operating constraints, and evaluation criteria pertinent to accomplishing a prescribed goal. Furthermore regardless of how manufacturing is to be done, designers need information that allows them to decide the optimum geometries and shapes for the processes. They may also need advice on the geometrical limitations of a process. If this information is not available at the conceptual design stage, the resulting component may be impossible or costly to make.

Theoretically, design for manufacturability involves considering manufacturing constraints throughout the design process, as shown in Figure 2.9. It starts at the conceptual design stage and continues through the embodiment and detailed design stages. Different companies have tried a variety of approaches to implement the DFM methodology. With the advent and popularity of various CAD tools, there is increasing interest in supporting DFM through intelligent CAD systems (Gupta, 1994).

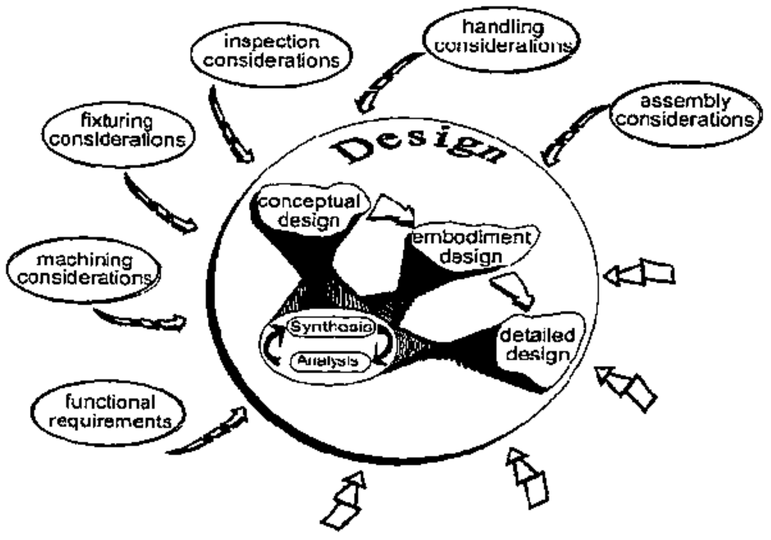


Fig 2.9. Design for Manufacturability. (from Gupta, S. K. 1994)

The DFM considerations differ dramatically in detail within different manufacturing technologies, since certain DFM considerations are composed by the specific manufacturing process involved. In this thesis, consumer product design is selected as the application domain for which injection moulding is considered one of the major manufacturing methods for the design examples used to test the developed system. DFM considerations considered in the computer model developed in this thesis include varied constraints of design and manufacturing guidelines and rules of the proposed certain manufacturing methods, and specifically the early anticipation of manufacturing problems at the conceptual stage of the design process.

2.6. Genetic Algorithms

2.6.1. Overview of Genetic Algorithms

During the last three decades there has been a growing interest in algorithms that rely on analogies to natural processes. The emergence of parallel computers with massive process power made these algorithms of practical interest. The best-known algorithms included:

- Evolutionary Programming,
- Genetic Algorithms,
- Evolutionary Strategies,
- Simulated Annealing, and
- Classifier Systems.

In the book (Michalewicz, 1992), a term "evolutionary program" is used to define a subclass of algorithms, those which are based on the principle of evolution, survival of the fittest. Correspondingly, Bentley (Bentley, 1999) classified these algorithms as evolutionary algorithms. In such algorithms a population of individuals, the potential solutions, undergoes a sequence of transformations using the mutation and crossover operators. These individuals strive for survival: a selection scheme, biased towards fitter individuals, selects the next generation. After some number of generations, the program converges guided by selection based on the predefined fitness function or diverges with random selection.

In general, Genetic Algorithms demonstrate the basic principle of these algorithms. And they are the best known and the most widely used evolutionary techniques (Holland, 1975 and Davis, 1991). They resemble natural evolution more closely than many other approaches because they are based on the mechanics of natural selection and natural genetics. Universal Darwinism suggests that natural evolution acts through selection, transmission and variation (Dawkins, 1983).

The basic concept of a Genetic Algorithm is to encode a potential solution to a problem as a series of parameters. The coded parameters are normally referred to as genes, with the values of a gene as alleles. A collection of genes in one individual population is held internally in a computer system as a string, and is often referred to as a chromosome. The entire coded parameter set of an individual, which may be anything from a single gene to a number of chromosomes, is known as the genotype, while the outcome that the coded parameters define is known as the phenotype.

In the first population, candidate solutions are created initially with random parameter values. These solutions are bred with each other for several simulated generations under the principle of survival of the fittest, which means the probability, that an individual solution may pass on. Some of its parameter values to subsequent children are directly related to the fitness of individual, for example, how well that a solution is relative to the others in the population is determined by the fitness function. At this point a Genetic Algorithm can start generating new populations. Breeding takes place through the use of operators such as crossover, which simulates basic biological cross-fertilisation, and mutation that is essentially the introduction of noise.

The simple application of these operators with a reasonable selection mechanism has produced startlingly good results over a wide range of problems.

A simple Genetic Algorithm works as follows:

```
{
   Initialize population;
   Evaluate population;
   While TerminationCriteriaNotSatisfied
       {
           Select parents for reproduction;
           Perform recombination/crossover and mutation;
           Evaluate population;
       }
}
```

Genetic Algorithms differ from traditional optimisation algorithms in four ways (Goldberg, 1989):

- GAs usually work with a coding of the parameter set, not the parameters themselves,
- GAs search from a population of alternatives, not a single individual,
- GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge, and
- GAs use probabilistic transition rules, not deterministic rules.

In detail: first, Genetic algorithms manipulate decision or control variable representations at the string level to exploit similarities among high performance strings. Other methods usually deal with functions and their control variables directly. Because genetic algorithms operate at the coding level, they are difficult to fool even when the function may be difficult for traditional schemes. Second, through working from a population, the probability of reaching a false peak is reduced by GAs. In this way, GAs find safety in numbers. Third, Genetic algorithms achieve much of their expansion by ignoring information except that concerning payoff. While other methods rely heavily on such information and in problems where the necessary information is not available or difficult to obtain, such as some design problems, these other techniques break down. Genetic algorithms process similarities in the underlying coding together with information ranking the structures according to their survival capability in the current environment. By exploiting such widely available information, GAs may be applied to virtually any problem. Fourth, the transition rules of genetic algorithms are stochastic, while many other methods have deterministic transition rules. GAs use random choice to guide a highly exploitative search.

2.6.2. Application of Genetic Algorithms

One of the regions in which Genetic Algorithms perform well is optimisation and many results have been reported in the last ten or fifteen years (Holland, 1992 and Goldberg, 1994). Different problems in different areas have had solutions successfully optimised by GAs and design problem is one of common problem areas to use GAs as a way of optimisation. In his thesis, Bentley gave a detailed list of design related optimisation problems tackled by GAs (Bentley, 1996). Frazer claimed Genetic Algorithms as powerful generative and adaptive techniques applicable to computer aided architecture design (Frazer, 1995).

The application of Genetic Algorithms to product design has been limited as a result of a lack of research on how to associate 3D design problems with the generative power of Genetic Algorithms. Also a wide variety of knowledge and information required by a product design at the early creative state of the design process is difficult to formulate in a computer system. This kind of knowledge and information are related to the definition or description of the function, shape, ergonomics, environmental impact of a product, and so on. The design of a product first requires a clear understanding of the functions and the performance expected of that product. Strategically, reaching a target can be seen as a two-stage process. This is, thinking of the possible ways in which the target might be reached and selecting the best of these possibilities. Here in practice, this means that creative idea generation is followed by systematic idea selection using the design specification as the basis, and later in the design process, selecting the best embodiment design for a product involves the same process. That is, thinking of the possible ways in which the product could be made and then choosing the best from them. The same cycle repeats itself throughout product development, operating within the progressively narrower boundaries established during the previous stage of the product.

There are three essential design activities at any stage of the design process: generation, evaluation and selection. Searching techniques are involved in all these activities. Genetic Algorithms are suitable to support all these activities. To apply Genetic Algorithms to the product design process, there are two major significant issues based on the feature of the algorithm itself as described above in the difference between GA and other traditional optimisation techniques. First, the GA does not search from one single point, but from a population of points. As a result the results generated at one generation is not a single design but a population of design alternatives. The evolutionary process is, at any one time, considering the whole populations of proposals for a given design problem. Second, the GA uses stochastic reproduction instead of deterministic rules. Through the random choice, an exploitative search will be guided so that sometimes the unexpected or innovative results will emerge.

2.7. Evolutionary Design

Evolutionary design is an approach that utilises different evolutionary computation techniques in various different design domains. The strength of evolutionary design comes from the observation that controlled evolution can be formulated as a general purposed problem solver with the ability similar to human design intelligence but with a magnitude of speed and efficiency. Traditional AI methods such as rule-based reasoning have to model design intelligence explicitly in terms of knowledge both in representation and inference. These methods have serious drawbacks because the process of how human designers actually use this kind of knowledge is not necessarily fully understood.

Evolutionary design is classified into four categories by Bentley (Bentley, 1999):

- evolutionary design optimisation, which is concerned with optimising existing designs by evolving the values of suitably constrained design parameters;
- creative evolutionary design, which generates entirely new designs from little abstract knowledge to satisfy functional requirements;
- conceptual evolutionary design, which deals with the production of high level conceptual frameworks of preliminary designs; and
- generative evolutionary design, which directly produces forms of designs contributing to the emergence of implicit design concepts.

These evolutionary design approaches combine several vital aspects of design intelligence in an evolutionary process including modelling design data and information, concept formation, idea generation, optimisation, learning, and evaluation. Once a design problem is properly formulated in this evolutionary process, the computer is able to generate a large number of candidate solutions before reaching an optimum one. The candidate solutions are sometime unpredictable but the process and the final outcome are manageable by the designers.

Evolutionary design extends and combines CAD with analysis software. It borrows ideas from natural evolution. Therefore this paradigm has excellent potential for developing more intelligent design support tools. For example, Frazer used Genetic Algorithms in his evolutionary architectural design to evolve unpredicted forms of architectures and their possible interactions with the environments (Frazer 1995, 1996 and 1997). Chakrabarti developed a functional synthesis program that generates a large number of abstract design concepts from functional requirements and abstract building blocks of engineering elements (Chakrabarti et al., 1996). Thornton utilised Genetic Algorithms as constraint management tools in the process of embodiment design (Thornton, 1994).

However, the development of evolutionary design tools is at its early stage. So far, many Genetic Algorithms have been used and tested only in design problems of small scale. A theoretical understanding of evolutionary design and its applications in design process is necessary. A computational model is needed in order to formulate product evolution processes in which Genetic Algorithms can be used as general purposed problem solvers. In an automated and evolutionary computational process, the role of designers in relation to creative decision-making needs to be strengthened rather than weakened. In this sense, evolutionary design techniques as general purposed problem solvers or design support tools need to be integrated with knowledge-based design techniques in order to reflect the expertise of designers and experience in automated generative processes.

2.8. The Generative and Evolutionary Design Process

Theories and methodologies have been developed in Artificial Intelligence (AI) and Evolutionary Design for automatic generation of design concepts and intelligent search for design alternatives. These theories and methodologies are referred to as generative and evolutionary computation design techniques. They can be used by designers in the areas of architectural and engineering design to synthesise initial design concepts from functional design requirements. They can also be used for the exploration and optimisation of the initial concepts in order to generate alternative design solutions. The main advantage of using these techniques is to enable the designers to concentrate on the more creative aspects of the design while utilising massive computing power for generating and evolving candidate solutions.

Based on the theory of GAs, a generative and evolutionary process can be defined as consisting of four main phases (Frazer, 2001):

- start,
- generate,
- develop, and
- transform.

As shown in Figure 2.10. First, the problem needs some form of generic representation and then this representation needs to be described in a genetic code. Second, a population of code scripts needs be created. Third, designs are then generated from the genetic code scripts through some form of epigenetic development in an environment. These designs are evaluated and those most successful ones are selected. Fourth, the selected code scripts are transformed by reproduction operators such as crossover and mutation. The process is then repeated from step two till satisfactory results are achieved. This generative and evolutionary process forms the core mechanism of the generative and evolutionary design system developed in this thesis for which more detailed descriptions will be given in later chapters. While this generative evolutionary process is originated from the architecture design field, it can be applied to product design by adding domain and context dependent design knowledge.

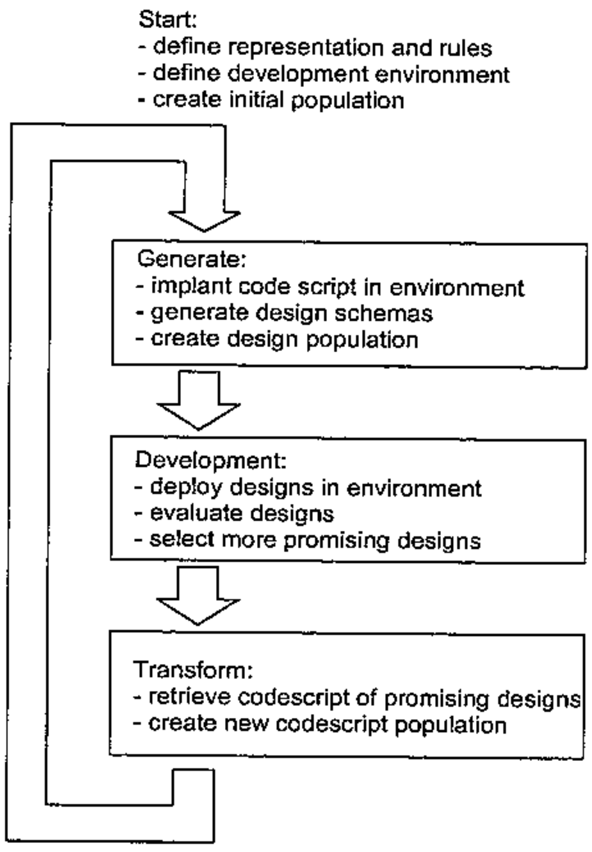

Start:
- define representation and rules
- define development environment
- create initial population
Generate:
- implant code script in environment
- generate design schemas
- create design population
Development:
- deploy designs in environment
- evaluate designs
- select more promising designs
Transform:
- retrieve codescript of promising designs
- create new codescript population

Fig 2.10. A diagram of the generative evolutionary process.

2.9. Summary

Based on the review, it can be concluded that the issues of how to model product design as an evolutionary process and the method for computational representations and inferencing mechanisms in such an evolutionary process to deal with shape constraints imposed by certain manufacturing processes must be addressed. So far a generative and evolutionary design system using Genetic Algorithms in product design process with manufacturing considerations does not exist. This provides an opportunity for the development of a system capable of generating potential product designs with DFM considerations. The remaining chapters present how such a system is developed and tested.

Chapter 3

3. A Generative and Evolutionary Product Design System

In this chapter, a computational model of an automated generative and evolutionary product design system is proposed. This model is described in this chapter in a more general design context, rather than in a specific domain of implementation. The structure, the main elements and features of the system developed in this thesis are described.

3.1. The Proposed System Model.

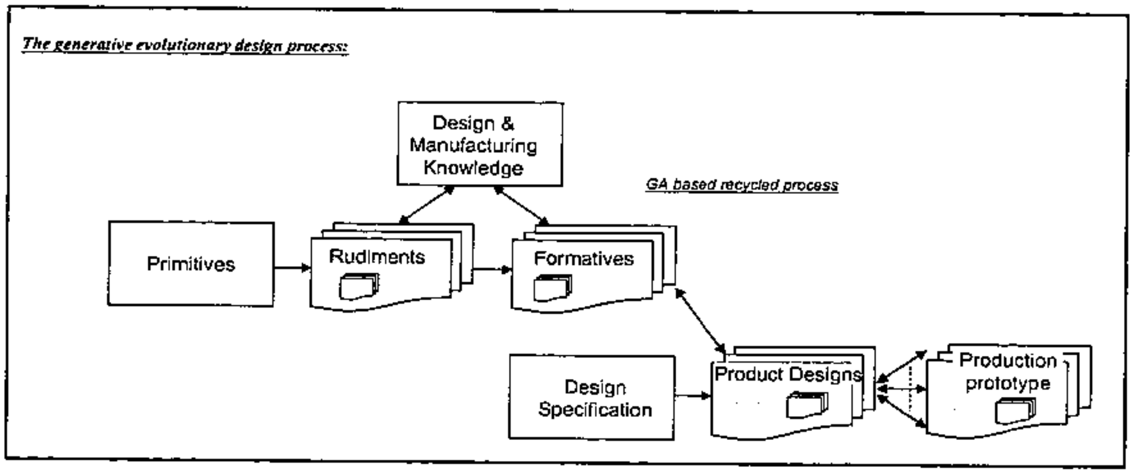

The generative evolutionary design process:
Design & Manufacturing Knowledge
GA based recycled process
Primitives
Rudiments
Formatives
Design Specification
Product Designs
Production prototype

Fig 3.1. The generative and evolutionary product design process based on CAD systems

Based on the computational model of a generative and evolutionary design process introduced in the preceding chapter, a prototype system has been developed. Figure 3.1 illustrates schematically this generative and evolutionary product design process. Comparing to the traditional and parametric CAD approaches illustrated in Figure 2.7 in Chapter 2, a significant point of this approach is that at the beginning of the design process, an abstract generic design concept is considered without having to specify many detailed parameters to define the design problem. Then during the generative and evolutionary process more specific and desired designs can be developed from this generic concept. More detailed explanation on its composition items will be given later.

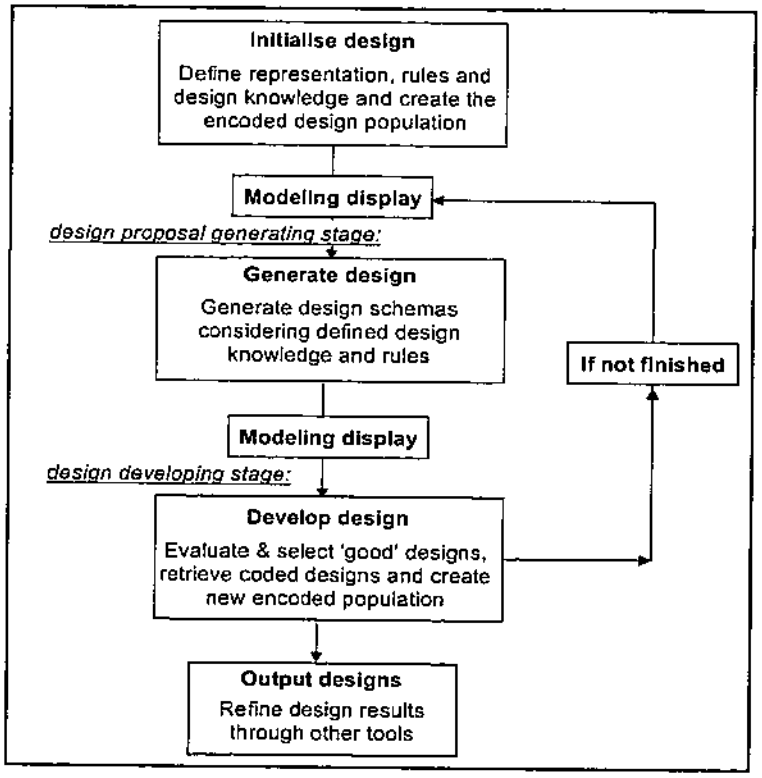

Initialise design
Define representation, rules and design knowledge and create the encoded design population
Modeling display
design proposal generating stage:
Generate design
Generate design schemas considering defined design knowledge and rules
If not finished
Modeling display
design developing stage:
Develop design
Evaluate & select 'good' designs, retrieve coded designs and create new encoded population
Output designs
Refine design results through other tools

Fig 3.2. The program structure of the generative and evolutionary process

As shown in Figure 3.2, a generative and evolutionary product design process consists of two cyclically linked stages - the design proposal generating stage and design developmental stage. At the beginning of this cycle, an initial population of an abstract design that embodies the potential for evolution of the full design must be created first. This requires an environment in which these initial populations can be specified and associated with those rules to be invoked to infer on the initial populations represented as code scripts. These code scripts are encoded strings that can be manipulated by a genetic algorithm. Usually, this initial population can be generated randomly, although code scripts can be specified in order to predispose the system to the creation of certain types of schemas. During this process, computer modelling is used to simulate the development of prototypical forms, which are then evaluated on the basis of their performance in the simulated environment within the evolutionary cycle.

During the first stage of this evolutionary cycle, the design proposal generating process generates each potential design from its associated code script in response to a simulated design scope and context in which the intended design is expected to perform. Such a proposal-generating step is crucial for the overall generative and evolutionary process to work. In this step design exploration task is supported by automatically producing large numbers of meaningful design proposals. This is achieved through the creation of rudiments and formatives as shown in Figure 3.1. As a general definition, a formative can be understood as an encapsulated design proposal, which defines a set of entities with relationships, as well as the generative rules involved during the generating process. A rudiment is a composition element of the formative, which defines the set of entities and related design knowledge. More detailed explanation on these will be given later in this chapter.

During the second stage, the design development process, the system develops those promising designs through the evaluation/selection, transformation and reproduction of the existing design populations. Evaluation/selection, transformation and reproduction are three core elements of the generative and evolutionary program employing a Genetic Algorithm. This part of the process resembles the natural evolution process of Darwinism, of which the selection, transmission and variation are three main ingredients (Dawkins, 1983). Transformed from their code scripts, the design proposals are illustrated and evaluated within a specified environment and context, and the "good" or promising design proposals will have a higher opportunity to pass their code scripts to the next generation. Through this way, the principle of "the survival of the fittest" during an evolutionary process is implemented. The evaluation of the design proposals can be made by evaluation algorithms automatically (referred to as the natural selection) or human user decision (referred to as the artificial selection). New design proposal inherits features of previous design proposals through the reproduction of code script information by crossover. Each population of design proposal contains a small amount of variations due to random mutations in the reproduction of code script information. Then at this point, based on the new design proposals generated, the evolutionary cycle starts again, which only stops when all requirements are satisfied or through the intervention by users.

To sum up, when creating a computational model of a generative and evolutionary process and applying it to any new applications, several main elements must be considered. First, the representation of design proposals with associated generative rules must be specified and devised. Second, encoded scripts of these design proposals, which can be manipulated by GA, must be defined. Third, the transformation and reproduction operators to be employed during this approach must be determined. Additionally, the fitness function must be decided to allow the evaluation of potential solutions by the GA program. DFM considerations are integrated into this process and made effective throughout in two ways, i.e., as knowledge and rules that can be built into rudiments and formatives, and as specifications interactive supplied by users as the selection and evaluation criteria.

3.2. The Design Representation

3.2.1. Formative - A Design Schema

According to the application of evolutionary program and the application of GA to any practical usage, the representation of the proposed design problem is the most fundamental and important element. The term, design schema, is first used by Frazer to describe an encapsulated design concept in evolutionary architecture design. In 1966, Frazer proposed a "Concept Seeding" model as a primitive description, which encapsulated not only robustness but also a complete design concept. It was subsequently referred to as an encapsulated "design schema" (Frazer, 1975). The example he developed next gave "seeds" for development into space frame structures but with a clear design schema and with intrinsic rules which guaranteed closure and structural integrity.

Amongst the advantages of such an approach was the rapid and semi-automatic development of the seed to a completed form analogous to the growth of an oak tree from an acorn. This opened the possibility of a cyclical approach where the seed and the rules for development could both be improved in an iterative manner. This approach was demonstrated to be particularly well suited to the application of that uses the genetic algorithm as a direct analogy with the evolutionary processes of nature. (Frazer, 1995 and Sun, 2000). Frazer recently further elaborated the definition of a design schema and he stated (Frazer 2001):

"Most designers employ a methodology highly personalised yet can often be generic when the designer's body of work is taken as a whole. It is part of their working method and hence characterises their style by which they are known……, This personalised but generic methodology can be described as a design schema in that it is an abstract conception of what is common to all designs. Inside the designer's office, these implicit design-schemas often become formalised. It is common to find sets of standard details in architects' offices that serve to economise in time, ensure details are well tested, but also to ensure a consistency of detailing and to reinforce the house style. In many offices this extends to design procedures, approaches to organisation and so forth."

In this thesis, this term of design schema is borrowed here and extended further which leads to the new term Formative used in this thesis. As stated before, a formative, by definition, is an encapsulated potential design solution, which defines a set of entities and relations, as well as the generative rules involved during the generating process. A rudiment is a composition element of the formative, which defines the set of entities and related design knowledge (Frazer, 2000). In the domain of product design, a potential product design solution corresponds to a formative and it contains the constitutional parts of a product structure, the relationship of these parts, and the configuration rules to build the product embodiment. Figure 3.3 shows the schematic rudiments and formatives developed. Based on the basic product elements, i.e., the primitives, the rudiments can be represented by the combination of them. In the existing CAD systems, the primitives are the geometrical forms and structures. For example in MicroStation, the block, sphere, cylinder etc. are 3D primitives with specified geometry. The rudiments in this approach are the composition of these primitives with associated product feature attributes. Then in the product design domain, they define functional components with related design knowledge, for example, the knowledge about manufacturability. And a formative encapsulates a set of rudiments with their relationships, as well as the product configuration rules that can be used to configure them into a more complex component or an assembly. These rules are invoked during the design generating process.

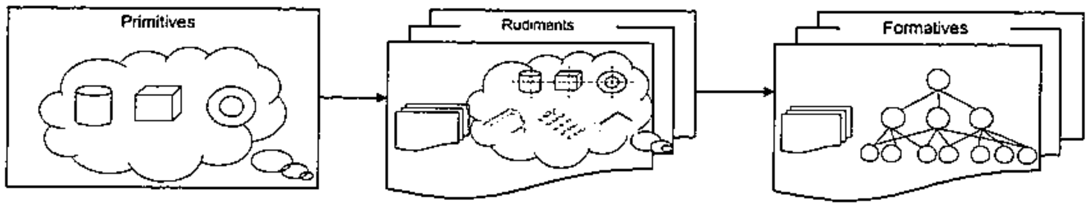


Fig 3.3. The schematic rudiments and formatives based on existing CAD systems

In the system developed, the representation of the formative consists of the functional component classes and associated product configuration rules. The representation of the functional component class is defined using constructive parametric representation. This representation combines methods from feature-based representation, parametric design and existing CAD modelling representation. In this representation, a product has a hierarchical structure derived from its functional decomposition. A functional component class consists of primitive entities, i.e., the rudiments. A rudiment is different from a primitive in that it is developed primitives and it has variables representing geometrical features as well as other associated attributes. The product configuration rules are represented by a structured grammar, which specifies how these primitive entities are to be assembled to form a formative.

For the system implementation presented in this thesis, only the representation of the functional component class is encoded as code scripts for the GA program. The configuration rules, which can also be encoded and manipulated by a GA, have not been integrated. Instead these are generated and modified manually. Through this representation method, a generic model of the representation of a product group can be built.

3.2.2. Encoded Formative and the Code Scripts

The evolutionary algorithm in this system does not directly manipulate a formative. Only the encoded formative in the form of code scripts represented in a computer program as strings, are actually modified by the genetic operators of the GA. Each code script represents a potential design solution and composes one of the chromosomes in a population of GA, and it is also named as genotype in SGA (Goldberg, 1989). In this system, every code script or a chromosome is arranged in a hierarchy consisting of multiple pieces of genes, each gene being defined by float values, as shown in Figure 3.4. This arrangement is corresponding to the structural representation used to define a formative.

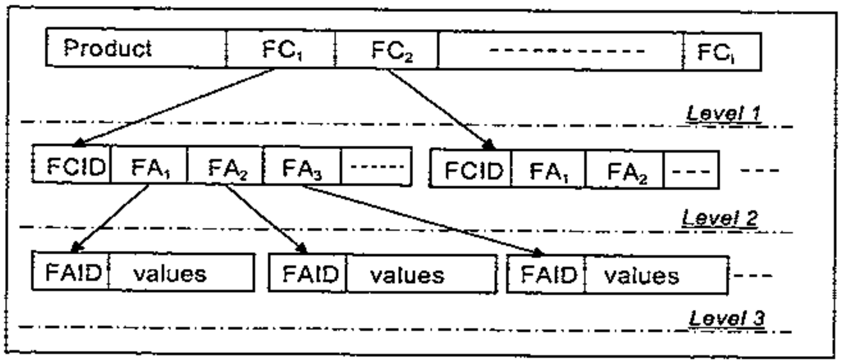


Fig 3.4. The multi-levels of the genotype representation

As shown in the figure above, a product structure has three levels. The top level is the product structure consisting of a number of Functional Components (FC_i). Each component chromosome is the combination of a selected set of Feature Attributes (FA_j), at the second level. In the figure, FCID and FAID are the abbreviation of IDentity numbers of the Functional Component and Feature Attributes, which are used here just for clear explanation but are not included in genotype representation. The value of each attribute is encoded into the genotype based on the same mechanism at the third level and a genotype will consist of a string of basic genes. Based on this hierarchical product structure, the corresponding code scripts also have a hierarchical structure, which forms the chromosome of GA. The encode mechanism and examples will be illustrated in chapter 5 in detail. The length of a chromosome in the GA program may be varied because different number or different types of functional components can be selected.

3.3. GA in the GEvoPD System

3.3.1. Genetic Algorithms

Genetic Algorithms are used as an engine mechanism in the evolutionary and generative process proposed, as shown in Figure 3.5. To build a successful GA program, appropriate data structures that correspond to the chromosome representation, should be used together with appropriate algorithms that correspond to "genetic" operators used for transforming one or more individual chromosomes.

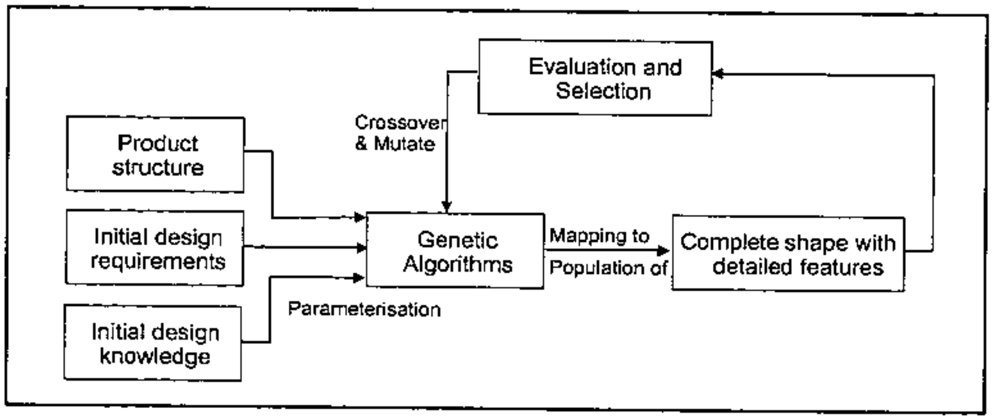


Fig 3.5. GA based evolutionary process

The genetic algorithm used in system is an improved one based on the SGA (Goldberg, 1989). The genetic operators, the crossover and mutation adopted in this approach are similar to that of SGA, but with several improvements based on the consideration of product design. These improvements are highlighted below:

- *Initialisation*: In this approach, the values of genes are specified within the variable bound. It also initialises (to zero) all fitness values for each member of the population. The initialisation procedure in the system reads upper and lower bounds of each variable from an initialisation file, which can be input in advance by users. And it then randomly generates values between these bounds for each gene of each genotype in the population. Here, a gene of the chromosome represents one design variable.
- *Keep-the-best function*: This function developed in the program keeps track of the best member of the population. And the last entry in the array of population holds a copy of the best individual of this generation.

3.3.2. Evaluation and Selection

As described in the preceding paragraph, the fitness function of the GA must be created to allow the evaluation of potential solutions of the problem. In this approach, the evaluation of designs is carried out by interpreting the generated design solution, and determining its behaviour according to a set of behavioural requirements formulated from the design requirements. The evaluation of the fitness of a design is a multiobjective problem. In this system, a multiobjective function is used to state the fitness function:

$$\mathrm{F}\,(K,f) \;=\; \sum_{i=0}^{n} K_i * f_i$$

In this equation, i is the number of an evaluated item, K_i is the weight to item i to describe its importance among all items, and f_i are the defined sub functions. Some subjective information like aesthetic, ergonomic considerations and manufacturing constraints are formulised as the sub-functions f_i at this stage and a designer can give weights to each evaluated item through the system interface (Sun, 2000a and 2000b).

In design, some of these sub-problems are well defined, as in engineering or other traditional optimisation problems, while others maybe not quite well defined or even ill-defined at the early stage. Then besides the selection based on the evaluation functions, artificial selections by the user are also supported in this approach, which means that during the design development process a designer or user can credit his/her favoured designs and thus guide system to evolve the promising designs.

Artificial selection can be a useful means for dealing with ill-defined selection criteria, particularly user centred concerns. These are usually difficult to formulate in quantitative forms before any potential solutions are emerged. However during the evolutionary process, a designer or a user might be stimulated by initial results generated by the system. At any moment when a designer or a user sees something different emerging, he/she may be able to use his/her intuition to judge what kind of designs can be selected for reproduction. Therefore it provides designers with an opportunity to use their experience and intuition to jump to faster results (Frazer, 1995). So this model provides for divergent evolution for the generation of alternative ideas. This gives a matrix of four possible combinations of natural/artificial selection and convergent/divergent evolution, as in Figure 3.6.(Dawkins 1986; Sims 1991; Graham, Frazer, and Hull 1993,1995; Frazer 2001).

	Convergent evolution	Divergent evolution
Natural selection	●	●
Artificial selection	●	●

Fig 3.6. Matrix of four possible evolutionary combinations

3.4. Integration of DFM Considerations

During recent years, the application of computer aided design systems to improve manufacturability is of ongoing interest in engineering design and much progress has been made. For example, the integration of the design and manufacturing features during the design process has been advocated and practised by many researchers. (Salomons, 1993). Features as used in design can differ significantly from those used in manufacture. The problem arising from this is sometimes referred to as the problem of the multiple views of design features and manufacturing features. Figure 3.7 gives a frequently quoted example of multiple views of design features and process planning features in the case of a prismatic component. In this example, the designer would prefer to design with protrusion features, the ribs, as these are functional to him. A process planner would look at the material to be removed, the depressed feature is of the most importance: in this case the slots and the step are seen separately. So an integrated design and manufacture features are proposed to solve this problem. (Salomons, 1995). In this developed system, an example of this integration can be illustrated as below: when thinking about the button design of a keypad, on the one hand it represents the form and size requirements on button shape design. On the other hand, it represents the form, size and location of the Hole feature on the product housing shell corresponding to this button during the manufacturing process. In this way, these two are connected with each other consistently.

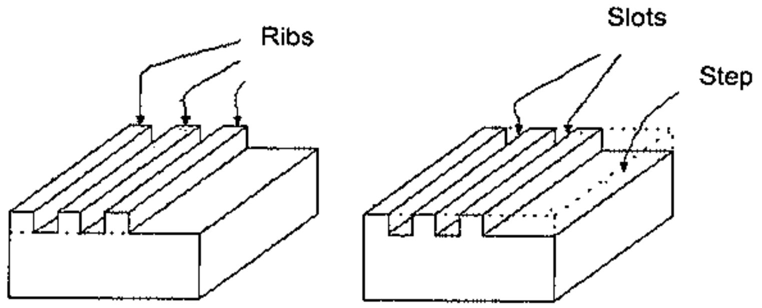


Fig 3.7. An example of multiple views of design features and manufacture features.

Furthermore, to involve the advanced computer aided design technologies into this field, the development of design for manufacture computer programs has also been investigated. A most general approach of this development is that, the designer provides a skeletal part based upon functional criteria and the computer system modifies the design for manufacturability. While in this system, manufacturing considerations and designer's experience are involved during the product design process to produce both makeable parts and guarantee optimally mouldable designs in the selected manufacturing method (in this cases injection moulding). Also in this approach as shown in Figure 3.1, design and manufacturing knowledge defines varied constraints of design and manufacturing guidelines and rules for the proposed manufacturing method and specifically the early anticipation of manufacturing problems at the conceptual stage of the design process. In this way, the DFM considerations are formulated as knowledge at the early design stage and made effects through the whole design process, which differentiate this approach with other approaches, in order to achieve more efficiency and intelligent supporting function.

During the system implementation, the consumer electronic product design was taken as an example and only injection moulding manufacturing method was considered. The generic evolutionary product design process developed in this system follows general manufacturability guidelines to avoid problems downstream in the detail design process, but it is not concerned with detailed mould design.

3.5. The Generative Systems

The system developed in this study is generative. The history of generative systems is summarised by William Mitchell (Mitchell, 1997). After tracing back the use of generative systems in architectural design, Mitchell outlined the concept of "shape grammar" or elemental combinatorial system, which forms the essence of a generative system. The term grammar in this sense was first mentioned by Chomsky (Chomsky, 1957) who applied it to natural language analysis. Now, a grammar is considered a formal device consisting of a set of production rules, a set of symbols, and an initial symbol or a symbol set. Grammar rules can deduce an initial symbol into a set of symbols, which together create a meaningful expression. Grammars exist in many forms and are classified according to their productions and the symbols they manipulate. Shape grammar and structure grammars are two of them.

Shape and structure grammars are computational formalisms for the representation of shapes and spatial structures. These related methods have been used fairly wide in architecture design for the development of formal approaches for producing designs in specific styles. In the following sections, shape grammars and structured grammars adopted in this study are discussed.

3.5.1. Shape Grammar

A shape grammar (Stiny, 1980a and 1980b) derives designs in the language it specifies by successive application of shape transformation rules to some evolving shapes, starting with an initial shape. It can be used to describe how a complex shape is built from simple entities and how a complex shape can be decomposed into simpler sub shapes.

A shape grammar is made of four different components (Stiny, 1980a). First, the grammar must have a finite set of shapes, which are the building blocks from which other shapes can be constructed. Second, a finite set of symbols is required, which in many cases can be regarded as markers or labels to indicate where shape rules are to be applied to. Third, a set of rules to transform one shape and symbol to another is included. Finally, the grammar requires an initial shape to start with. Figure 3.8 illustrates the components of a very simple shape grammar. And the generation of a shape using this shape grammar in Figure 3.8 is shown in Figure 3.9.

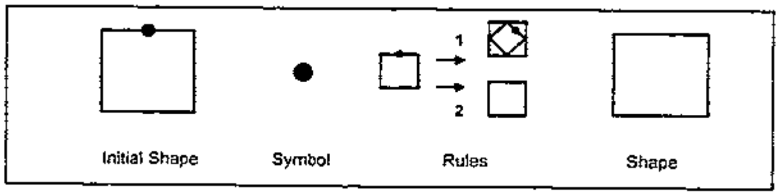


Fig 3.8. Elements of a simple shape grammar. (redrawn from Stiny,1980)

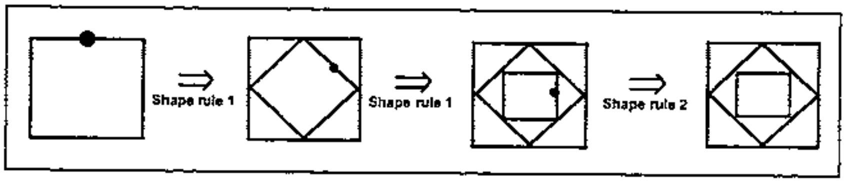


Fig 3.9. Generation of shape using the shape grammar. (redrawn from Stiny,1980)

Shape grammars have been successfully applied to generating permutations, which are then used in spatial design in the field of architecture. Some related research can be listed as, villas in the style of Palladio (Stiny and Mitchell, 1978), mughul gardens (Stiny and Mitchell, 1980), prairie houses in the style of Frank Lloyd Wright (Koning and Eizenberg, 1981), Greek meander patterns (Knight, 1980), suburban Queen Anne houses (Flemming, 1987). More detailed description on shape grammar and examples can be found in (Stiny, 1980a and 1980b). However, there has been a limited application of shape grammars to engineering design. Fitzhorn (Fitzhorn, 1990 and

1991) presented shape grammars for specifying the languages of realisable solids. Reddy and Cagan (Reddy et al, 1995) and Fenves (Fenves, 1996) presented a parametric shape grammar (an extension to shape grammar) for the design of truss structures that uses the shape annealing technique of Cagan and Mitchell (Cagan et al., 1993) to generate optimal truss structures.

Almost all these applications are limited to two-dimensional problems. Attempts to develop shape grammars for surface generation have been made too. As shown in recent literature, they have also been used in shape design of coffee machines (Agarwal, 1998) and mechanical design (Schmidt, 1996). While the method still has serious problems in the context of an automated product design system, the research on this topic is still ongoing.

3.5.2. Structure Grammar

Structure grammars are adaptations of shape grammars. There is an obvious difference between these two: shape grammar combines shapes to create new forms without apriori knowledge of the emergent outcome, while structure grammar specifies how a set of primitives is assembled to achieve a predetermined goal. (Carlson and Woodbury, 1990 and Wallace, 1991). For example, a structure grammar can be written to combine elements to form human-like configuration, as shown in Figure 3.10.

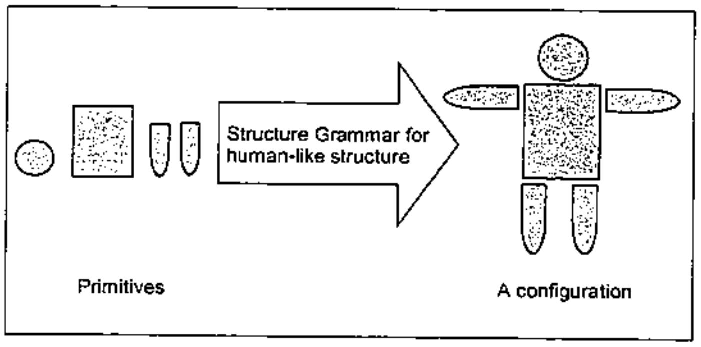


Fig 3.10. Example of a structure grammar.(redrawn from Carlson and Woodburg, 1990)

A good analogy for understanding structure grammar is to imagine a set of primitives as characters in the alphabet. Rules of spelling are used to assembly the letters into meaningful words or structures. This is a very useful way for describing the spatial positioning of product compositional components, for example, a specifying grammar for laying out a specified type of product, mobile phone. The limitation of the method is its specificity. Different grammars will be required to describe different objects.

In the example described above, which shows a grammar designed to create a human being, a different grammar would be required to make cats, even though cats and people are quite similar and both are vertebrates, with four appendages, a body, and a head. But human being and cats own the different primitives as well as different structures, so different grammars would be required. Similarly, it is necessary to construct separate grammars for televisions and calculators, even though both are electronic products with moulded housings. Structure grammar is therefore not appropriated for the proposed system developed in this study due to this problem. But based on its principle, a configuration grammar for a generic product model has been developed.

The configuration grammar used to represent the generative rules in this system is a special kind of structure grammar. The problem mentioned above is overcome through the definition of configuration rules associated with each functional component class. The configuration rules of any proposed product design are a combination of the rules of its functional component classes. This would allow the system to design televisions, calculators, and new, yet unknown, types of products using the same representation. Additionally even with the same functional component classes selected, with the same configuration rules, the design results generated through the generative and evolutionary process will still be diverse. This will be further illustrated by the concrete experiments in Chapter 7. The developed system can evolve designs only based on the supposing functional parts of the proposed product, without knowing exactly what they are. Configuration rules developed in this study are not encoded as genotype and manipulated by GA, they are built manually by the user through a system interface combined with some rules and predefined design knowledge. A sub-GA program can also be used to generate the configuration rules, but this system did not implement this sub-GA program.

3.5.3. Morphological Analogy

Analogy is a basic human reasoning process used in science, literature, art, education, and politics. Analogies are abstractions, which contain information about the situation they represent. The use of an analogy can provide a short cut in problem solving. Problems can be solved through comparison to the analogy (Barr and Feigenbaum, 1981). Morphological analysis, an idea borrowed from Biology, provides a straightforward method for the design and transferring of product surface styling and also product structure analogies. Morphology refers to the science of possible forms and morphological analysis was devised by D'arcy Thompson (Bonner, 1961) to evaluate the relateless of animal forms within a zoological class. By superimposing a grid over animal skeletons and performing various transformations he was able to assess evolutionary closeness, as illustrated in Figure 3.11. Morphological analysis separates the unique description of a configuration from the infinite variations possible through scaling or distortion (Steadman, 1983).

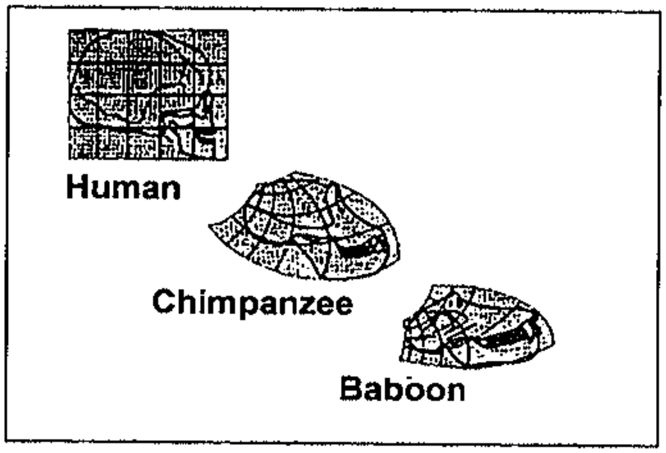


Fig 3.11. An application of morphological analogies

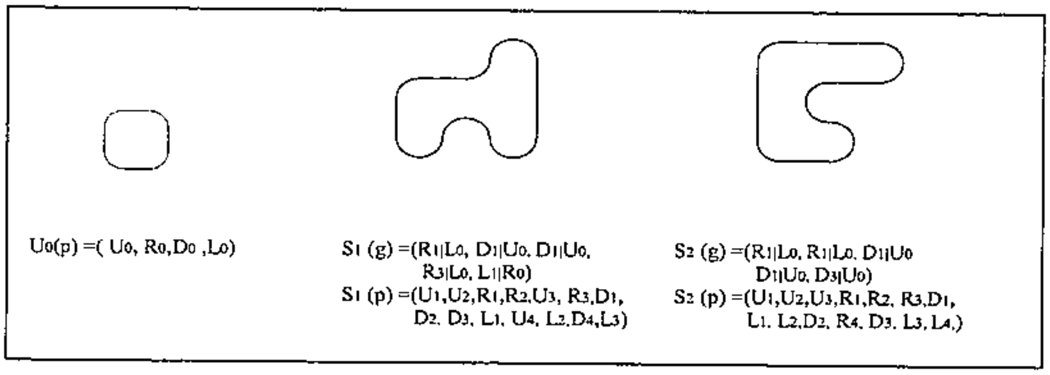
Using morphological analogy in design, we might consider two designed objects, which have essentially different forms, but similar appearances, to be analogous forms. In this system, the generated design alternatives can be further extended based on the morphological analogy. For example, a personal digital assistant like product can be also a weather clock with the similar appearance, having a display and some functional keys on its main surface, but a different design purpose for the selected functional component. This is very useful for the further refinement and exploration of the generated design results of this system.

3.6. Simple Examples of Generative Design Using GA

In this part, two examples of generative design using GA are presented. In these examples, GA is used for design concept exploration and adaptation, different representation methods are adopted according to different design problems.

3.6.1. A Generative Design of Enclosure Forms

In this example, an experiment on the form generation of a single component element, the enclosure housing, was conducted. The shape generation process starts with a basic shape unit, and new forms are obtained following the evolving generative rules. The shape unit used here is a square and represented by four edge-vectors as shown in Fig.3.12 (U0): Upgo=(1,90), Rightgo=(1,0), Downgo=(1,270), Leftgo=(1,180). When two units joined to form a new one through the conjunction of negative edge vectors, i.e., Upgo with Downgo or Rightgo with Leftgo, the conjoining of these vectors results in an internal edge which is invisible in the new shape. Shape generation grammar introduced in this example is based on this single rule about the conjunction of negative edge vectors. This is the single rule used in this example.


$U_0(p) = (U_0, R_0, D_0, L_0)$

$S_1(g) = (R_1|L_0, D_1|U_0, D_1|U_0, R_3|L_0, L_1|R_0)$
$S_1(p) = (U_1, U_2, R_1, R_2, U_3, R_3, D_1, D_2, D_3, L_1, U_4, L_2, D_4, L_3)$

$S_2(g) = (R_1|L_0, R_1|L_0, D_1|U_0, D_1|U_0, D_3|U_0)$
$S_2(p) = (U_1, U_2, U_3, R_1, R_2, R_3, D_1, L_1, L_2, D_2, R_4, D_3, L_3, L_4,)$

Fig 3.12. Symbolic version of the simplified genotype and phenotype representation.

To simplify the representation, the sequence of joined edges instead of rules is coded as the genotype, and sequence of edge vectors describing the shape is coded as phenotype in GA. Figure 3.12. shows the symbolic version of the genotype and phenotype of another two generated shapes during the evolving process. Figure 3.13 shows some of the shape generation results with profile shapes on the left and simple solids on the right using an extrusion solid modelling operation. The process of the shape generation starts with the basic shape unit, and new forms are generated by over-crossing the fitter ones of the previous generations following the rules for evolving. Aesthetic factors of the smooth surface, the numbers of the angles and the minimum perimeter of the profile are considered as the evaluation elements here. Simple extrusion of a profile or sweeping along a skeleton line is chosen during the visualisation process on the Microstation platform with Visual Basic programming language.

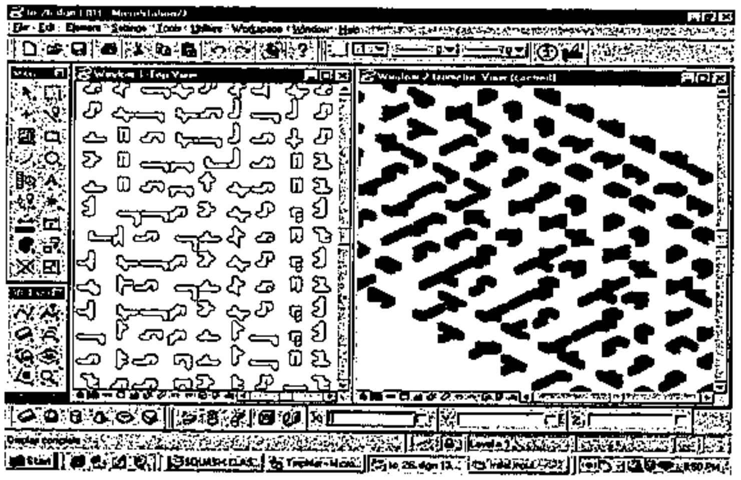


Fig 3.13. The generated shapes with profile on the left and simple solids on the right.

These results demonstrated that the algorithm works in real time for simple forms, like a housing of a controller or any similar object. Then shapes of other component elements of the product can be generated based on the same mechanism with different evaluation conditions and design requirements in the evolutionary process. A hierarchical evolutionary approach can be derived from the hierarchical product structure for the product representation. Then based on the hierarchy evolutionary approach, new primitives can be obtained through the combination of evolving processes at lower levels. In this example, the length of genotype is set to be constant to simplify the process. With the extension of genotype, which means more rules are included, more complex shapes with higher diversity can be obtained. All these potentials and capability are further developed during this system development as presented in this thesis.

3.6.2. The Generative Design of Glasses

In this example, the profile of a glass is selected as a design object. The GA manipulates the co-ordinates of the control points on the profile. The control points are introduced in the profile representation, and a control curve is described by choosing several points on it, as showed in Figure 3.14 b. Curves are used to describe a 2D-profile shape of the glass. The 3D shape is generated by revolving the profile around a central axis. The co-ordinates of points, which define the location of each point, are chosen as the characteristic parameters and encoded as the genotype. The user can evaluate each individual and assign its fitness value, which will be used by GA for selection. Some design results generated are shown in Figure 3.14 a.

This example was developed for a demonstration on the applicability of GA in product design. And software program of this example was developed jointly with another member, Mr. KwaiHung Chan, of the Design Technology Research Centre.

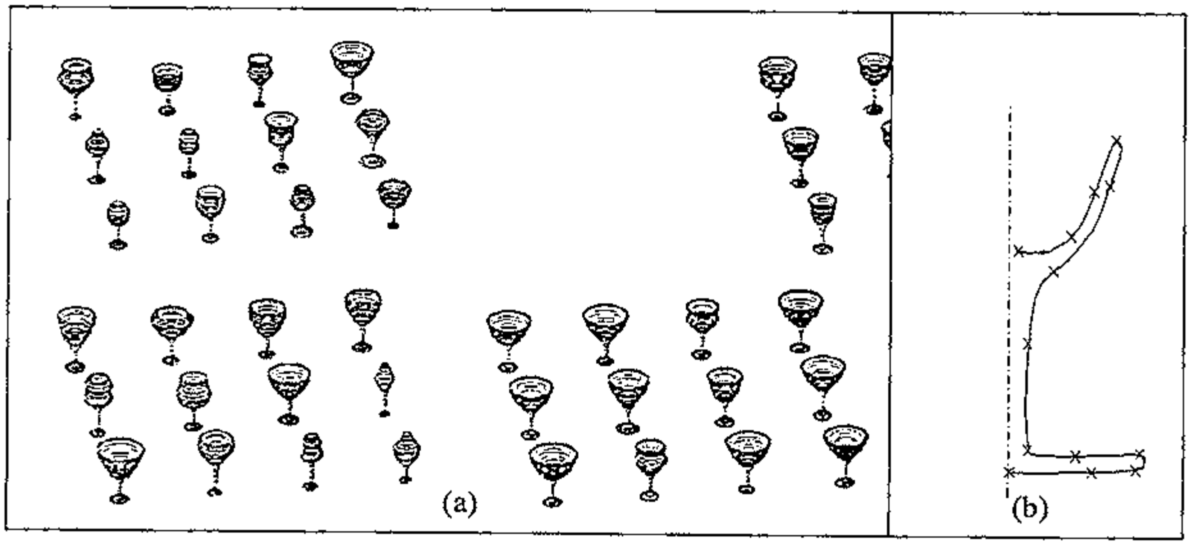


Fig 3.14. (a) Some diverse results of the generative process.
Fig 3.14. (b) The profile curve with control points.

3.7. Summary

The generative evolutionary product design process developed in this thesis consists of two cyclically linked stages, the design schema or proposal generating stage and the design developmental stage. This chapter has given an overview of the system prototype and outlined the main elements of the developed system. Each of the elements of the system was identified and summarised briefly. These elements will be fully explained in the remaining parts of this thesis when it comes to discuss the details of system implementation and evaluation.

Chapter 4

4. Product Design Representation

Design is a dynamic process, in which information is constantly reinterpreted and restructured, especially in the early phases due to ill-defined problems and poorly structured information. This makes modelling and representing design information very hard, because any representation for design must be generic and flexible to incorporate design data as well as design knowledge. Formalisation of design knowledge in the representation to support decision making throughout the design process for the exploration of design alternatives using evolutionary techniques is a key issue addressed in this thesis. This requires that the representation adopted in this approach must support not only a generative and evolutionary design process, but also the performance analysis and manufacturability evaluation. Such a representation should also be generic and elastic in order to model a wide range of product design domains.

In order to apply genetic algorithms to the product design support system developed in this thesis, a new representation scheme consisting of rudiment and formative was developed. In this representation, functional components with features and attributes as building blocks of design are defined in advance in a database as rudiments. The formative, which is a potential product design can be created, by combining functional components using their associated configuration rules. In addition, this representation includes graphical interfaces to allow 3D visualisation, simulation and integration with other 3D solid modelling systems.

This chapter introduces this representation scheme by describing how the concept of rudiment and formative were developed and what advantages it has in comparison with other feature based or geometric representation schemes that have been introduced in former chapters.

4.1. The Concept of Formative

As briefly introduced in the preceding chapters, a representation termed "design schema" was developed by Frazer (Frazer, 1995) primarily for architecture design, but the idea has not been tested in the domain of product design in which conflicting requirements on the product functionality and form must be balanced for manufacturability. The idea of design schema, though very successful in several demonstrations, has not been formalised as a representation scheme within a generative design support system framework. In this thesis, the basic idea of design schema is further developed in the domain of product design and a new and more generic representation called "Formative" representing an initial design concept is proposed (Frazer, 2000).

To establish the concept of formative for product design, a generic product data model needs to be defined for the specification and generation of design concepts reflecting design intent. Here, the word "Formative" is used to describe this generic product data model. A formative is an under-developed or immature structure of a design with the potential for complex development. It includes necessary primitives of a product and the rules for developing and evolving into a range of specific products under a given design proposal scope.

The similar idea can be seen in the definition of the universal plant archetype, the Urpflanze as shown in Figure 4.1. Goethe defined the "Urpflanze" as a universal plant archetype (Mitchell, 1990), the essence of which is to be found in every plant instance.

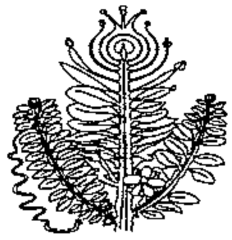


Fig 4.1. The Urpflanze.(Mitchell, 1990).

In detail, a formative encapsulates the product component classes as well as the configuration rules and their relationships. Thus the formative owns the potential for developing and evolving into a number of specific product forms under a given environment, such as a specified design requirement.

Then the term "Rudiment" is used to define the basic product component classes. Rudiments can be understood as advanced primitives, which are primitives or combination of primitives with some initial development and knowledge already encoded within.

Next, in order to apply evolutionary techniques to the generative design process, the representation and encoding scheme of formative and rudiment need to be developed first. In the following section, the representations of rudiment and formative are described.

4.2. Representation of Formative

When applying evolutionary techniques to design problems, a mechanism is needed for capturing potential design solutions and codifying them in the generative and evolutionary system. Generally, two kinds of representations of design solutions have been achieved till now. In one approach, the genes represent the transformational grammars, which have their unique interpretation within a specific problem. Results generated through this approach are most abstract and need further mapping or transformations. Frazer shows some examples of this approach:

"During the generative process, the design schemas are expressed as generative rules so that their evolution may be accelerated and tested. The rules are described in a genetic language which produces a code scripts of instructions for form generation". (Frazer, 1995).

Another view of this is that the evolved genes form the basis of a representation of a design case or cases, which can then be used to generate designs that are adaptations of the original cases. This approach is sometimes also regarded as an optimisation and adaptation approach. Bentley has done an example research on the table design using this method (Bentley, 1996).

While the representation method adopted in each approach above has its own focus, the former is focused on the representation of form generation process and the latter is more focused on the representation of the data, detailed design cases. The separation of design data and process, in which these data may be used, is also a main problem of most existing product design representations illustrated in Chapter 2. Then the representation method of formative developed in this thesis is trying to integrate these two with the combination of the existing representation methods.

The representation of the formative consists of two main parts:

- Composition components of the product model, and
- Configuration rules of the product model for development and evolution.

The configuration rules in this case can be regarded as a structure grammar that can be applied to the generative design support system as mentioned in last chapter. An investigation of the product structure decomposition and the existing representation methods was carried out before the representation of formative and rudiment was developed.

4.2.1. Functional Components and Elements

Based on the common functional decomposition of products, a constructive parametric representation method has been developed in this system to represent the rudiment and formative. This representation emphasised not only representing the geometric features of a design object but also the relationships among these features, and the related design knowledge.

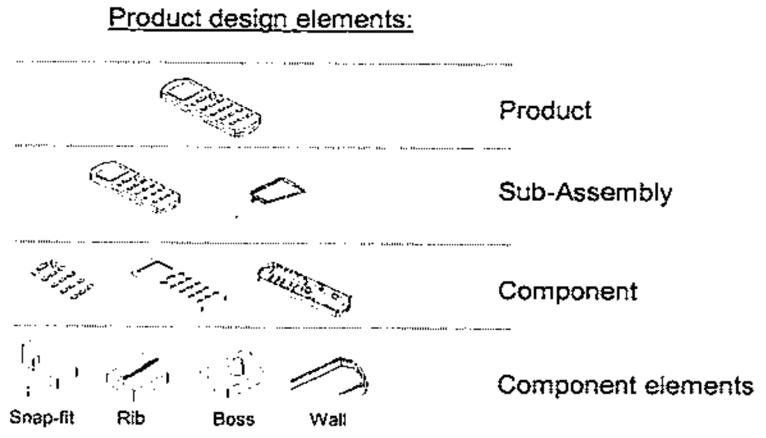


Fig 4.2. Product composed of design elements

Hierarchical decomposition is one way of reducing the complexity of design objects. An artefact can be viewed as a system that can be further decomposed into subsystems according to different criteria, such as construction, function etc. Thus observing product structures from a constructional viewpoint gives rise to a product breakdown structure. A schematic structure of a product tree is illustrated in Figure 4.2.

In this tree, an artefact consists of a collection of elements. The product breakdown structure consists of different classes of product design elements:

- A product is an artefact purposely designed for a user, for example, a remote controller;
- A subassembly is a product structure consisting of a set of components, for example, the controller enclosure, which assembles several separated components, such as numeric buttons, top plastic enclosure, and bottom plastic enclosure;
- A component is a single physical object produced without any assembly operations; for example, the bottom plastic enclosure of a remote controller;
- Component elements are basic product design elements that constitute a component that can be called as feature groups. Component elements include snap-fits, and rib features etc.

As described above, an assembled product consists of various components, and a component is further comprised of different features, such as shells, holes, ribs, etc. Features are considered as the smallest data units here which posses explicit engineering meaning and each component element is the combination of feature groups. During the design process, a designer can gradually define the product breakdown structure in a constructional domain using different product elements or feature groups. So the product representation can be derived from this structure sequence in a top down manner:

- Product,
- Subassembly,

- Component, and
- Component element.

Similar to this structure, there are several methods developed to represent a product. Such as the product tree representation developed by Lucienne Blessing (Blessing, 1996), which uses a structure consisting of product, assembly, component, part, and feature. A similar representation consisting of product, assembly, parts, and features are adopted by some commercial CAD packages, such as Inventor, SolidWorks and ProE.

4.2.2. Functional Decomposition

Based on these existing product decomposition methods, simplified product structure decomposition is adopted in this approach in order to integrate the product representation with Genetic Algorithms. This decomposition method is based on functional product features and its sequence is:

- Product,
- Functional component, and
- Component features.

This decomposition sequence is used because the design of products, especially consumer products, such as mobile phone, telephone and calculator, often tends to be driven by a basic functional decomposition but the products themselves are differentiated by form. For example, in a mobile phone, a number keypad is provided for the phone dialling and varied control keys are provided for obtaining other information. A screen is used to show or edit all visible information and antenna for receiving signals etc. Most major brands of mobile phones can be decomposed this way. However, the form that the product takes, and in particular the differentiation in the form of each functional component, generate for a wide variety of mobile phone products on the market. Additional functional features, independent of the functional breakdown, further differentiate mobile products. Considering the generic representation of this domain, including consumer electronic products such as remote controller, mobile phones, calculator, personal digital assistant, and televisions etc, all of them have something in common. That is, these products all have a box-like housing or enclosure and most functional components are distributed on the outside surface, which will benefit from the extraction of the product configuration rules developed later. So based on this simple functional decomposition, many consumer electronic products in this domain can be represented as combination of some of the existing functional components.

4.2.3. Constructive Parametric Representation

Based on the decomposition structure of a product, a hierarchical approach is proposed as the framework of a generative and evolutionary design support system, as illustrated in Figure 4.3 below. The advantages of a hierarchy approach are: first, only those factors relevant to component design are considered; and second, factors relevant to the relationships between functional components are treated at their product level.

As shown in Figure 4.3 only three levels, the product level, functional component level and component feature level, are considered during the evolution process. Each higher level can be represented by the composition of one or several individuals at the lower level. At each level, a component is generated from a combination of components from the level immediately below. At each level, an initial population is generated and then evolved over a number of generations, until one satisfying population of objects at that level is obtained. Members of that population are then selected as suitable components for generating the initial population at the next level.

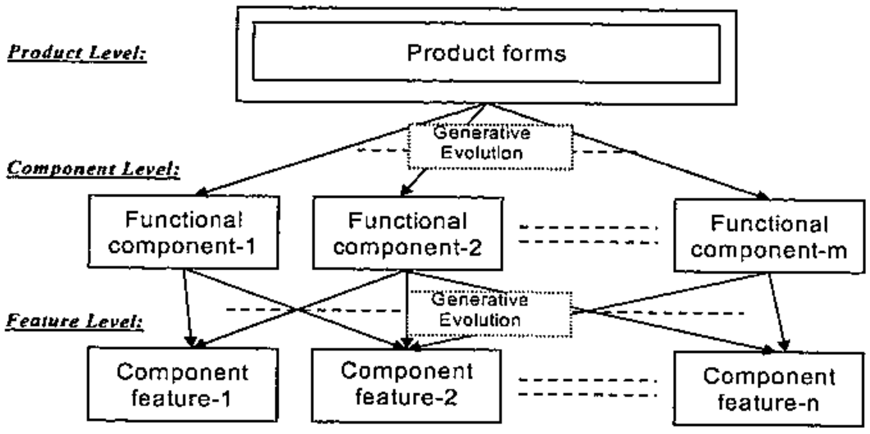


Fig 4.3. Basic structure of the proposed multi-level evolutionary process.

A suite of all functional classes of a product group with appropriate attributes is pivotal to this concept. Take the consumer electronic products as an example, main functional classes include:

- number keys pad,
- functional keys pad,
- screen/visual display,
- acoustics/speaker,
- Microphone.

The classes' decomposition is based upon a criterion, which defines that a functional component class is a group of components with similar functions. All these classes considered here are placed on the surface of the product. After using the method, the operation rules and the evaluation factors for generating new designs can be identified for a class, rather than for individuals.

As to the representation of the functional classes, in this approach a representation method that combines geometric parameters and other feature characters or attributes is used. As shown below, table 4.1 lists the schematic representation of five functional component classes of consumer product group.

Each functional component is described by selected feature attributes as well as the geometric parameters. For example, the number keypad class is represented by its three geometrical parameters, the length width and height, and two parameters about other feature attributes, the number and shape style of the keys. Table 4.2 shows the definition of the product group in the program implementation.

Table 4.1

Functional Classes	Feature Attributes
Nk_Pad	Size(L, W, H), KeysNo., Keys_shape
Fk_pad	Size(L, W, H), KeysNo., Keys_ shape
Screen	Size(L, W, H), Srn_ shape
Speaker	Size(L, W, H), speaker_ shape
Microphone	Size(L, W, H), Microph_ shape

Table 4.2

Description of the functional Classes in VB:

```
Type ClassInf
    CName As String
    CPict As String
    Alen As Integer
    Atxt (AttrNo) As String
End Type

Global Fminf (FMnoMax) As ClassInf
```

4.2.4. Rules for Product Configuration

Besides the representation of the composition components, the representation of configuration rules of the product model is another and important part of formative representation in this approach. The configuration rules in this system can be regarded as a structure grammar applied to the generative system developed. Then in this case, a product model is broken down and described by combined blocks, and the configuration rules specify how to locate the functional components into these blocks to form a complete product model.

It is proposed that a product configuration model can be derived from the proposed formative and then the generative rules for this formative can also be invoked for evolution. In this approach, the electronic consumer product is selected as design example. As described in former chapters, a main similarity among these products is that all these products have box-like housing or enclosure. Besides this, this kind of product owns some general characteristics. Ergonomic, manufacturing and aesthetic constraints must be considered as in any other product design domains. The early conceptual stage of the product development concerns primarily with the appearance and human usage. Injection moulding is assumed as the primary manufacturing process for these products. The product housing is divided into two parts along the parting plane with desired orientation, vertical or horizontal. Product usage is either handheld or desktop lying.

In addition, the shape design of the product housing or enclosure can be divided into two integrated steps:

- External visible surface of the housing, and
- Internal functional surface shape of the housing.

During the first step, the initial topology of the product is defined and then it is treated as the foundation of the second stage. Since the housing will be used and viewed by users, a surface finish of very high quality on side of the housing is often required for aesthetic reason. While on the other hand, the inner side of the housing often must be a very complex shape in order to mount all of the components and to provide structural rigidity. The task for the generative and evolutionary system here is the generation of both external and inner surface shapes of the housing. The external shape is originated from a generic model with the product component requirements as input and rules or procedures of functional component layout and aesthetic constraints as a generative environment. And then the internal housing shape is evolved through the structural construction operators, where the fundamental manufacturability constraints related to the geometry of injection mould parting and constraints on the solidification of molten plastic are embodied.

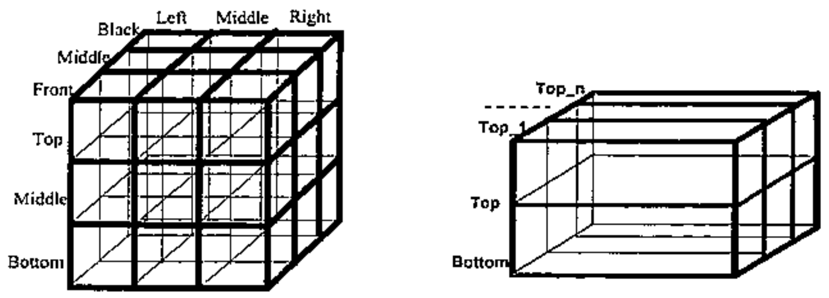


Fig 4.4. A product matrix model by Wallace (Left) and a simplified model (Right).

Based on these considerations, a product grid is introduced in the representation of product configuration. Each grid or cube of the model corresponds to a conceptual face or a region of the product. This model was also defined as Product Matrix by Wallace as sown in figure 4.4 (Wallace, 1993). Wallace gave the original description on the product matrix, which was first created in consumer product design domain. This matrix structure was created as a generic product model and Wallace believed that any kind of products could be represented this way. A much similar method was adopted by Chen to generate a range of designs, such as desk and chair etc. (Chen, 1998). In these applications, the matrix allows the program to assign components to regions without specifying physical dimensions. And the product matrix is complete when each component is assigned to an appropriate matrix element.

Thus a product matrix is built as the suitable configuration model of the system. This approach addresses the initial topology and tries to get it right at the first time.

Organisation or the spatial location of components can be described in two steps. A rule-based approach builds an intermediate structure, i.e., product grids, using ergonomic and manufacturing rules.

This approach is considered appropriate for the generative and evolutionary system developed in this thesis because manufacturing and ergonomic requirements can be readily expressed as heuristics or rules of thumb. The matrix structure establishes the product face and region that components will occupy without having to precisely locate them first. Next, the system locates the components physically, thereby defining the product's volume. In this approach, the GA can be utilised to evolve the parametric variables, which define the functional component instances. A procedural approach was used in the system developed in this thesis in the positioning stages because aesthetic concerns are difficult to articulate as rules.

Using the product grids, a model can be built which describes the spatial positions of product components or the component spatial layout. The external surface of one product design will be different from another based on the different layouts. The product configuration rules involved in this approach specify how those functional component classes are to be assembled to form a complete product. The attributes of each component class determine which set of rules to be applied. That is, each functional component class has a responding configuration rule set. For example, for a screen part of a controller, it should be convenient for the user to see, so it should occupy a Front and Top grid but not the Black and Bottom one. All these are predefined in a database as common sense knowledge in the system. A simplified product matrix model as shown in figure 4.4 right, is developed in this system implementation and the detailed explanation will be given in chapter 6.

A set of functional components, with variable parameters of size and attributes, is combined in a certain order following a set of rules to form a product model. The system developed is based on this constructive parametric representation, using the exiting modelling methods provided by the CAD system platform to realise the shape modelling and visualisation.

4.3. Summary

When applied to product design domain, the generative and evolutionary design system relies on a suitable generic product representation to allow the definition of a wide range of designs, and to enumerate the design space in a manner suitable for evolutionary search. The concept of Formative was introduced as an encapsulated design schema in this generative and evolutionary design system. The representation of formatives leads to the representation of a generic product model. The representation of formative consists of two main parts. One is the representation of the composition functional components, referred as rudiment in this approach, and another is the representation of product configuration. The representation of the rudiment consists of the parametric variables of the features with associated design knowledge. The representation of the product configuration is formalised as the construction rules in this approach based on a product grid model. Although, the representation method is derived from a certain selected example design domain, the product model built is generic for this design domain and it can be extended to other new application domains.

Chapter 5

5. Genetic Algorithm for Generating Design Solutions

As indicated in the second chapter, the algorithm developed in this system relies on the analogies to natural evolutionary process and is based on the principle of GA. And the evolutionary process described in this thesis puts more emphasis on the process involved rather than the algorithm itself. As indicated before in this thesis, Genetic Algorithm refers to a class of algorithms that are based on the same principle but with different features in different applications. Besides the data representation described in the former chapter, the algorithm developed contains all other associated elements of genetic algorithms. The algorithm developed in this system implementation is based on a Standard Genetic Algorithm (Goldberg, 1989) with specific representation and evaluation mechanism adopted.

This chapter describes the genetic encoding of designs based on the representation adopted and the associated genetic operators used in the evolutionary process. This section also defines the methodology adopted for the evaluation and selection of potential solutions.

5.1. Introduction

As illustrated in Chapter 2, the usual, also named as standard form of genetic algorithm was described by Goldberg (Goldberg, 1989). Genetic algorithms are stochastic techniques based on the mechanism of natural selection and natural genetics. Genetic algorithms, different from conventional searching techniques, start with an initial set of random solutions called *population*. Each individual in the population is called a chromosome, representing a potential solution to the problem at hand. A chromosome is a string of symbols; it is usually, but not necessarily, a binary bit string. The chromosome evolves through successive iterations, called generations. During each generation, the chromosomes are evaluated, using some measure of fitness (Gen, 1997). To create the next generation, new chromosomes, called offspring, are formed by either (a) merging two chromosomes from current generation using crossover operator or (b) modifying a chromosome using a mutation operator. A new generation is formed by (a) selecting, according to the fitness values, some of the parents and offspring and (b) rejecting others so as to keep the population size constant. Fitter chromosomes have higher probabilities of being selected. The general structure of genetic algorithms is described as shown in Figure 5.1.

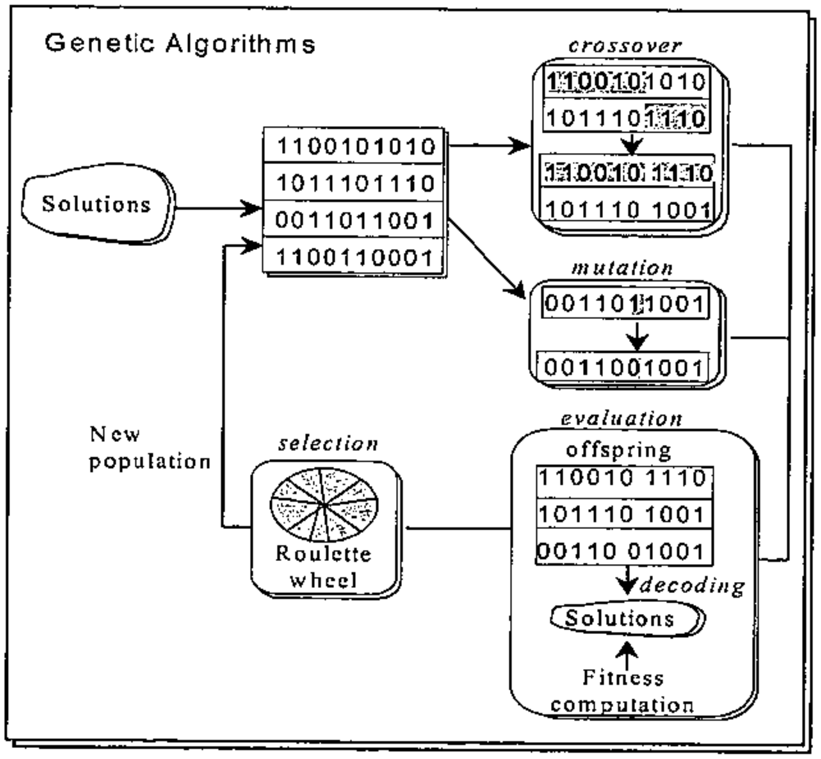


Fig 5.1. The general structure of genetic algorithms.(redrawn from Gen.,1997)

The system developed in this study mainly benefits from the adaptation and exploration of design results through the application of genetic algorithms. Several aspects of the algorithm used are based on a SGA and are described through the several aspects illustrated below. They include the hierarchical structure of the chromosome, specified encoding, evaluation, and selection scheme.

5.2. Hierarchical Chromosome of the GA

As indicated in the introduction of the GA theory given in previous chapters, the basic data structure a GA is a string, which corresponds to naturally occurring chromosome. And this artificial chromosome may be mapped to a potential solution to the problem being investigated or as the design solution being developed in the case of this research. Based on the hierarchical design representation presented in Chapter 4, the design proposal or the phenotype is represented as the combination of functional component classes and their associated product model configuration rules. Since every functional component class is defined by its associated feature attribute parameters, each chromosome consists of a list of multiples of attribute parameters. A chromosome structure developed in this system is shown in Figure 5.2.

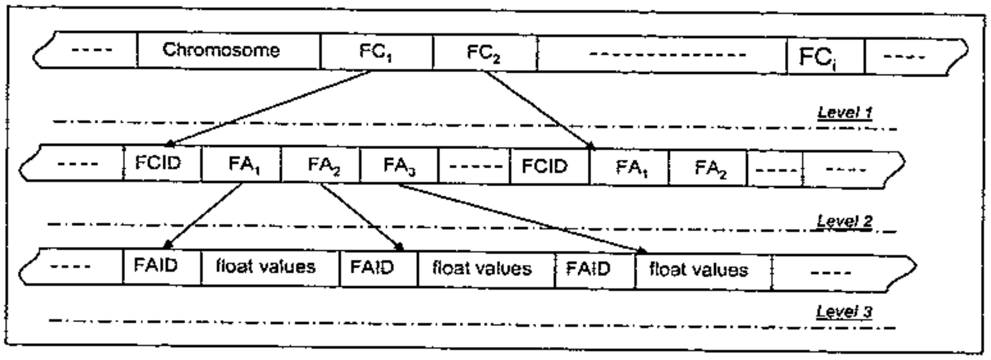


Fig 5.2. The multi-levels of the genotype representation

A chromosome is composed of three levels. The top level is the product chromosome that consists of a number of Functional Component Classes' chromosome section, FC_i. And each component class chromosome section is the combination of the selected Feature Attributes chromosome section, FA_j, at the second level. In the above figure, FCID and FAID are the identity numbers of the functional component classes and class attributes, which are used here just for clear explanation but not included in genotype encoding. Then each attribute is encoded into genotype based on the same mechanism at the third level and a genotype consists of a string of basic genes. In this system, the parameter values of each feature attribute are encoded into genes in the genotype representation.

The length of the chromosome of GA is varied when different number or different types of functional component classes are selected during an evolutionary process. Then based on this chromosome representation, the encoding method of the represented elements should be defined next.

5.3. Encoding of Design Solutions

In nature, the genetic information is encoded in a string of DNA using a four letter alphabet of the nucleotides, i.e., Adenine, Cytosine, Guanine and Thiamine - arranged in triplets that may then be decoded to produce one of twenty amino acids. Sequences of these triplets are used to encode the information necessary to produce the protein and enzymes required constructing, maintaining and regulating a living organism. In another words, the natural evolution process operates on the coded instructions for how organisms should be grown, i.e., the DNA, and it does not operate directly on the organism themselves (Goldberg, 1989). Likewise, the genetic algorithm operators on coded design parameters are not directly on solutions. This can be simply explained schematically by shape generation process in nature and CAD in Figure 5.3.

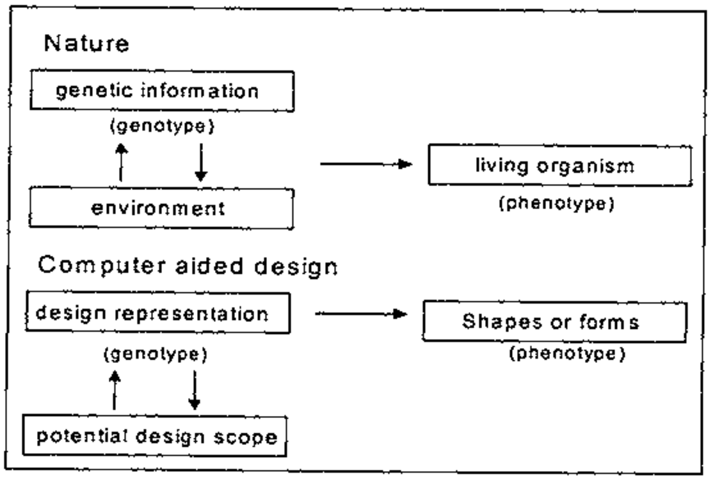


Fig 5.3. Shape generation in Nature and CAD

5.3.1. Encoding Methods

Based on different representation and encoding schemes, the genotypes can be binary strings, real value strings or other possible forms (Gen, 1997, Goldberg, 1989). Binary encoding has in the past been favoured by researchers because of its simplicity and functionality. As any numerical parameter set can be mapped onto a binary string, it is often preferred as a good general, non-domain specific representation that is "robust" across a wide range of problem areas. The use of bit strings, as well as being computationally convenient, simplifies the design and implementation of genetic operators such as crossover and mutation. As believed, binary string based encoding for a GA may be robust (Davis, 1991). In practice, it has been found to be not always as efficient as it could be within a specific domain or in a real-world application, where it may be outperformed by a GA employing a more natural representation. Another problem with binary representations is that they can present the programmer or system designer with certain conceptual difficulties in mapping a real-world problem onto a bit string. For this reason, other domain specific structures have been used, for example the real or integer lists.

A conceptually familiar coding system may be preferable to binary strings with the use of higher level programming languages, which make the design and coding of GAs possible and more accessible to non-computer specialists. Existing representation schemes may be exploited and mapped directly to strings. Real number arrays may be the system of choice for engineering and mathematical applications where the array may represent the parameters of an equation or the dimensions and settings of some structures or systems. Integer array, on the other hand tends to be favoured in ordering or sequencing tasks such as job scheduling - e.g. the travelling salesman problem (Oliver, Smith and Holland, 1987)

Besides these, an interesting development in encoding schemes is the use of mathematical functions, rather than just the parameters of them to represent the genetic string. Even the entire programs, in the form of LISP S-expressions, have been used as artificial chromosomes. This technique was pioneered by John Koza and described in his book "Genetic Programming" (Koza, 1992). He and others have used GAs to evolve programs to implement game playing strategies and to simulate the behaviour of artificial insects. In the area of Computer Graphics, Karl Sims has used a similar technique to evaluate natural looking textures in two-dimensional cellular automata (Sims, 1991).

During the early stage of this system implementation, encoding schemes of both binary string and real/integer lists are tested. While the later one is proved to be more suitable for this case. So the encoding scheme based on the real/integer lists adopted in this approach is described in detail next.

5.3.2. Encoding of Data Representation.

At this stage, only the functional component representation is encoded as the code scripts manipulated by the GA. Every functional component class is defined by its associated feature attribute parameters. And in order to allow a genetic algorithm to manipulate the values of these parameters, they must be coded in a way to form code scripts, i.e., the genotype. So based on the hierarchy chromosome structure shown above in Fig 5.2, the genotype consists of multiple functional component classes, and each functional component class consists of encoded parameter values of its associated feature attributes. The genotype consists of multiple groups of coded parameters corresponding to the related feature attributes. The feature attributes of a functional component class include the geometric features as well as other characteristic attributes of this component class, which is predefined in a database.

For example, a schematic phenotype representation of the Numberkey_Pad class in this system is provided in Figure 5.4. Not all of these data are chosen to be encoded into the appropriated data type, i.e., integer or real values, which can be manipulated by GA. For this component class, only those parameters shown in italics of the phenotype representation in this figure is encoded into genotype. Figure 5.5 shows the involved feature attributes of several functional component classes for the genotype representation in this system. This representation is flexible and allows a genetic operator to be applied to one sub- string independently of others.

Data Type	Item's Name
String	Component class name
Integer	Identity number
Integer	Number of attributes
String	List of attributes
Integer	*Keypad shape type index*
Integer	*Number of buttons*
Integer	*Button shape Index*
Float	x,y,z-centre locations of the keypad
Float	*L,w,h-size parameters of the keypad*

Fig 5.4. An example of schematic description of the phenotype

Functional Component Class	Feature Attributes
No.key_Pad	Geometric(L, W, H), Keys_number, Keys_shape...
Fun. Key_Pad	Geometric(L, W, H), Keys_number, Keys_shape, array style...
Screen/Visual Display	Geometric(L, W, H), Screen_shape
Acoustic/Loudspeaker	Geometric(L, W, H), shape style
Microphone	Geometric(L, W, H), shape style

Fig 5.5. An example of schematic description of the genotype

For its implementation in this system, the genotype consists of the string of integers, upon which the reproduction operators of GA manipulate, and the phenotype contains the string real variables. Before the initialisation, the range of integer and the upper and lower bounds of each real variable, which represents a corresponding attribute parameter, are read from an initialisation file or are provided by a user as requirements. During the initialisation, each gene of the genotype in GA is randomly generated as an integer, within the range. All fitness values for each member of the population are also initialised (to zero). Then during the mapping from genotype to phenotype, this integer is mapped into a real value within the predefined variable bound. As shown in Figure 5.6, a geometric item is illustrated as an example, that is the length parameter of a product main body. And the integer range is defined between 0~100. According to the input design requirement, the product is proposed as handheld type, so the length should be limited in the range (40,60). Here, the number is just used to represent the unit size according to the selected grid unit of the CAD system, such as mm. So when the generated number is 75 in the genotype, then the parametric value of this item is 40+ (60-40) *75 / 100. To formalise this, for a defined integer range and proposed Max-Min bound of the parameter, the mapped real value of a generated integer of a parameter should be:

$$\text{Min} + \text{Integer} *(\text{Max-Min})/ \text{Range}.$$

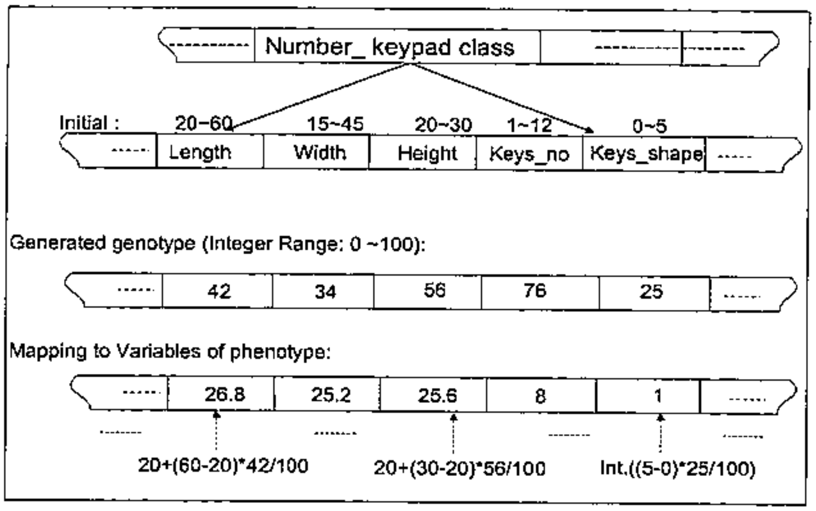


Fig 5.6. An example of encoding of the parameter values.

During the evolutionary process, the GA manipulates the integers, i.e., the encoded parameter values of the genotype. The phenotype is based on the transformation of the genotype as well as other parameters, such as the centre point co-ordinates (x, y, z) of each functional component instances when these are visualised in the CAD modelling. In the system developed in this thesis, these related parameters are defined during the generating process of the configuration rules of the product model.

5.4. Fitness Function for Evaluation and Selection

For a genetic algorithm, a single fitness score is all that is required by the evaluation function. But actually, there is always more than one evaluation criterion involved in each design solution. In this case, a composition fitness or evaluation function involving all these evaluation criteria should be developed. In some cases however, the criteria may be so different that it is impossible to combine the individual scores into a single score to guide the selection. This brings the multiobjective problem discussed below.

5.4.1. Multiobjective Optimisation Problem

A Genetic Algorithm cannot cope with more than one fitness value for one phenotype (Goldberg, 1989), so it requires numbered (scalar) fitness information to work. This means that when approaching multi-criteria problems, it is necessary to perform a scalarisation of the objective vectors. One problem is that it is not always possible to derive a global criterion based on the formulation of the problem. Without sufficient information, objectives tend to be given equivalent importance, and when the understanding of the problem is improved, then the objectives may be combined according to the information available, probably assigning more importance to some objectives. Optimisation for a combination of objectives has the advantage of producing a single compromising solution, requiring further interaction with the decision-maker (Fonseca, 1995). But if an optimal solution cannot be accepted, either because the function used excludes aspects of the problem which were unknown prior to optimisation or because an inappropriate setting of the coefficients of the combining function is chosen, then after adjusting these items additional runs maybe required until a suitable solution is found.

Many researches have been done to extend the GA to deal with multiple objectives. Schaffer used the Pareto optimality technique to cope with such non-commensurable criteria (Schaffer, 1985). This technique results in a "set" of optimal individuals rather than a single super individual. In his thesis (Bentley, 1996), Bentley presented a review of existing application and development of GA for multiple objectives, such as Pareto-optimality, Aggregating approaches, and other Pareto-based methods etc. Besides these, two new approaches to multiobjective optimisation using GA have been provided by Carlos A Coello (Coello, 1997). In his approach, in order to extend GA to deal with multiple objectives, the structure of the GA used has been modified to handle a vector fitness function. As presented in his paper the Pareto solutions produced by those methods are guaranteed to be feasible, as opposed to other GA-based methods, in which there may be convergence towards a non-feasible solution.

In this system, the method adopted to solve the multiobjective problem of GA is based on the principle of an aggregation approach. In this approach, separate fitness values are simply weighted and summed to form a single overall fitness value for each phenotype. The value for each weight requires complex and careful adjustment before successful design results can be evolved. To avoid this problem at this stage, the weight of each evaluation criteria is assigned by the user through the system interface, which then guides the design evolving direction.

5.4.2. Definition of Evaluation and Selection Criteria

As indicated above, the evaluation of product design in this system is a multi-objective optimisation problem. It is carried out by interpreting the generated design solutions represented by the phenotypes, to determine their behaviour according to a set of behavioural requirements formulated from the design requirements. Each design requirement or consideration can be treated as an evaluation criterion to be formalised as a sub-evaluation function of the evaluation software. The evaluation software performs an analysis of how well the evolved designs are or how fit they are, and then determines the transformed fitness value for GA selection. In this way, the evaluation software guides the evolution of design and directs it to the designs that perform the desired function. In his thesis, Bentley developed evaluation modules for his generic evolutionary design system. Based on the hypothesis that it is possible to break the function of any design into a number of desired characteristics, a number of separate modules of evaluation software were used in his system. Each module analyses a design for a single characteristic and these modules were treated as the sub-function of a multiobjective optimisation problem.

The key to successful multiobjective optimisation is the way in which the principle of "the survival of the fittest" is implemented. This can be done by ranking the solutions in the current population according to their dominance and then selecting the top ranked ones to be parents of new offspring. One solution dominates another if it is better on all counts. It is also beneficial to select parents from the archive, the record of the best solutions found during search, as well as from the current population.

In this approach, the weight of each sub-function is used to describe its dominance to other sub functions. And the weights of sub functions are distributed in advance by the program and also can be reset by the user through system interface during the evolution process. Evaluation criteria as sub functions involved in this system will be illustrated in detail in the system implementation later in Chapter 6. Also when "artificial selection" is considered in the multiobjective evaluation function, it can also be treated as a special case when the weights of other sub functions are assigned to zero by the user through the system interface.

5.5. Genetic Operators

During the evolutionary process, the new offspring are created by combining the features (control variables) of two parents through an operation known as crossover and by introducing small random changes through an operation known as mutation. This process is defined as reproduction, which involves these two operators.

5.5.1. The Crossover

As mentioned before, the crossover operator is what distinguishes the GA from other evolutionary techniques. Several versions of the crossover operator have been developed, the simplest of which is the single point crossover. The "traditional" form of crossover is one-point crossover. It is applied to two parent strings with a certain fixed probability known as the crossover probability. It works by selecting a point between two string positions, breaking the parent strings at that point and exchanging the broken-off section. In this way, two new strings or two offspring are formed. As shown in Figure 5.7 (a), the two-parent chromosomes are cut at matching points, which are randomly selected, and each point is also the head of the first joined to the tail of the other.

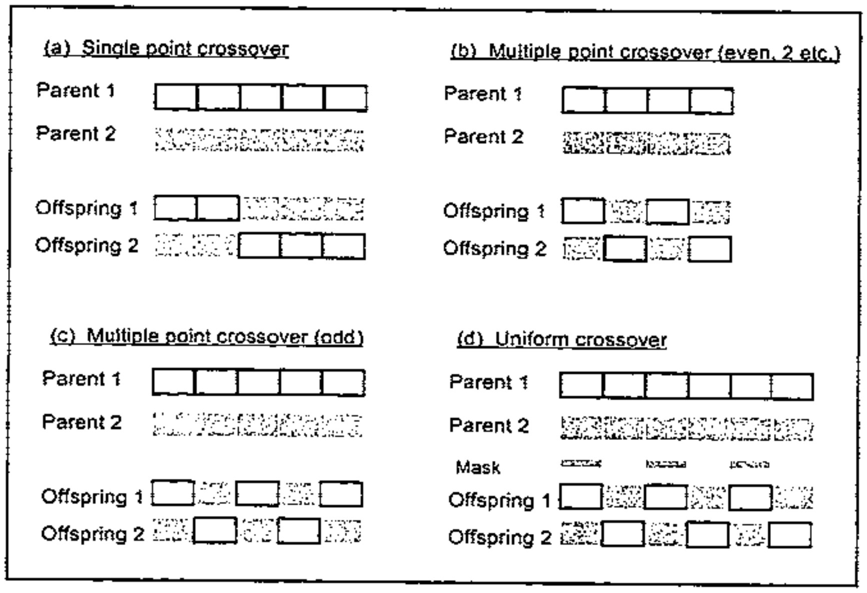


Fig 5.7. Crossover types.

Single point crossover has a major problem that it is unable to combine sections of genetic code at opposite ends of one parent in the offspring produced by crossing it with another string. This is partially solved by using multiple-point crossover as shown in Figure 5.7 b-c, where several points are randomly selected. A two-point crossover operator was found more effective than one point crossover, and is applied to each pair of selected parents with a probability. It creates a single offspring from two parents. The new sub string formed is then checked to replace any duplicated digits by others randomly determined. However as may be seen in the diagram, when an old number of points are selected, the problem mentioned above remains. Another variant is uniform crossover. It works by exchanging matching bits from each parent. This exchange may be random for each location or presented by applying a mask, as shown in Figure 5.7-d. Syswerda has indicated in his research that this method may be more efficient than the two types described above (Syswerda, 1989). Also some new crossover methods have been developed based on different specific representation methods. For example Bentley presents a structural crossover method based on the structural GA chromosome representation in his thesis (Bentley, 1996).

Based on the hierarchical chromosome representation used in this system, the crossover operator adopted in this program is also a structural one to enable the offspring to be valid.

5.5.2. The Mutation

Among the reproduction steps, mutation is the introduction of small, random changes to the genetic code. Its implementation is closely linked to the representation scheme used. It should be stressed that mutation is best used sparingly, for as in nature, the effect of mutation is more likely to be detrimental than beneficial. However, used at a low level it can introduce diversity into a population and reconstruct genetic information that has been lost in previous selections.

In binary coded implementations, mutation consists of simple bit switching "1"to "0" and "0" to "1". Integer mutation consists of incrementing or decrementing an integer by some random value or among a defined range. Real number mutation is described by Davis (Davis, 1991) as real number creep, and it is similar to integer mutation except that the value is generally changed by some proportion of the original value. On the other hand, mutation is much trickier when applied to function/equation or program based representation. Koza (Koza, 1992) omitted the mutation operator altogether partly on the grounds that he believed that it plays such an insignificant part in nature.

Goldberg (Goldberg, 1989) quoted a typical mutation rate of one bit per thousand as being sufficient, while, Schaffer (Schaffer, 1989), following a study of control parameters on the GA performance, recommended a rate between one per one hundred and one per two hundred. Mutation can also help to prevent premature convergence, and this may justify its use at higher frequency in GAs using small populations.

In this system, the range of mutation value is defined when the bounds of the integer are initialised and a default value of mutation probability is given at initialisation. And the mutation adopted is the real number mutation described above. Also the user can modify the probability of crossover and mutation through the system interface for its performance testing.

5.5.3. Converge and Diverge

During the reproduction process, another problem is the convergence and divergence of the generated results. It is the ability of the selection and crossover operators that identify and isolate desired genetic material at the same time as "weeding out" unfit material that makes genetic algorithms so effective as search techniques. Provided that the selection criteria or the environmental factors remain constant, convergence is almost inevitable. As the fittest individuals are favoured, their genes will tend to dominate, and eventually define the population. This tendency is observed as a move from an initial, diverse population towards an almost uniform population as the search narrows. Eventually the only genetic diversity present will be introduced by mutation. The direction of the search is dictated by the selection criteria, however the outcome of the search is also determined by the genetic material available at any given point in the search. If the population convergence is too fast, potentially resulting in premature loss of genetic diversity, the efficiency of the search may be reduced. Although premature convergence can produce an optimum solution, it is not necessarily the "most optimum" solution possible. By rapidly narrowing the search area, genetic material that could produce fitter populations if recombined with high scoring material may be lost. Techniques have been devised to ensure that genetic algorithms do not converge prematurely. For example random-biased based selection techniques may be preferred to elitist selection strategy (De Jong, 1975). The random element can improve the chances of potentially useful but low to average scoring code being retained in a population until it can be exploited (Schaffer, 1989).

As most implementations of genetic algorithms run automatically, the selection criteria are predetermined and remain fixed throughout a run of the GA. Differences in the outcome of separate runs of a GA using the same selection criteria will be due to the random effects introduced in selection and mutation, as well as the range of possibilities contained in the initial population. However, the solutions produced if given consistent and identical selection criteria or environment pressures, should be similar. In an interactive system, however, the criteria can be changed during the course of a run. Under these conditions, as mentioned earlier, there is no single evolutionary target upon which the population may converge.

5.6. Generating New Population

The genetic algorithm applied works with a fixed size of populations. Initially, it seeds the population with randomly generated strings. The strings are evaluated, or mapped into solutions to which the fitness function can be applied, and then selection occurs. Selection picks pairs of strings from populations. The probability of a string being picked is proportional to its fitness. The pair of strings undergoes reproduction, a process consisting of two operations: crossover and mutation, and the offspring produced are placed in a new population. The process of selection and reproduction is repeated until the new population is full. The cycle of producing a new population from an old one by selection and reproduction is called a generation. The GA is run until a certain termination condition is met. For example, the GA might be run for a fixed number of generations or until a solution fitter than a threshold value is found.

In the approach used in this study, the new population replaces the old ones after each generation and two more operators are adopted during the generating of new population in this algorithm. *Keep-the-best* function keeps track of the best member of the population, and ensures that the last entry in the array *Population* holds a copy of the best individual. *Elitist* function stores the best member of the previous generation as the last in the array. If the best member of the current generation is worse than the best member of the previous generation, the latter one would replace the worst member of the current population. If the best individual from the new population is better than the best individual from the previous population, then the best from the new population is copied over. Otherwise the worst individual from the current population is replaced with the best one from the previous generation.

To build a successful GA program, appropriate data structures that correspond to the chromosome representation, should be used together with appropriate algorithms that correspond to "genetic" operators used for transforming one or more individual chromosome. In the experiment conducted in this thesis, an improved SGA (Goldberg, 1989) is developed (Sun, 1999, 2000a and 2000b). Table 5.1 listed out some parameters of the genetic algorithm used in this system during the evolutionary process: the maximum and minimum number of Population size (PopMax, PopMin), the range of running generations (GenMax, GenMin), probability of crossover (XoverMax, XoverMin), probability of mutation (MutMax, MutMin).

Parameters	Value
PopMax	200
PopMin	9
GenMax	3000
GenMin	10
XoverMax	0.9
XoverMin	0.1
MutMax	0.2
MutMin	0.1

Table 5.1. Parameters used for the GA setting: The maximum and minimum number of Population size (PopMax, PopMin), The range of running generations (GenMax, GenMin), probability of crossover (XoverMax, XoverMin), probability of mutation (MutMax, MutMin).

5.7. Summary

When creating a computational model of the generative and evolutionary process and applying it to any new applications, these major elements must be considered:

- representation of design schemas with associated generative rules;
- encoded scripts of these design schemas, which can be manipulated by GA;
- transformation and reproduction operators to be employed;
- fitness function to allow the evaluation of potential solutions of the problem for GA, and;
- finally in this study, the DFM considerations involved during this evolutionary process.

Then based on the representation described in last chapter, this chapter described the other elements of the organisation of the system.

A hierarchical parameter representation of functional classes of a product is developed in this thesis, which allows a range of products with similar functions to be appropriately defined and easily manipulated by a GA. Variable-length of chromosomes in GAs is addressed to allow the number of selected function classes that define a product design to be variable. Also, multiobjective optimisation, evaluation/selection are investigated to allow users to define design problems with a number of considerations or constraints. And a suite of modular evaluation software is also introduced to allow a user to define a new design problem quickly and easily by picking combinations of modules to guide the evolution of design.

Chapter 6

6. Implementation of a Product Design Support System

This chapter presents the implementation of a product design support system that is used to demonstrate the application of the proposed generative and evolutionary process in the domain of product design. The first section gives a general description of the system architecture and provides an explanation of the program steps. Then two important factors of system implementation are described. These are the system interface and the initial design settings. After that, the detailed implementation of sub-processes, the generating module, the developmental module and graphical visualisation of the generated results are presented. The last section describes the DFM considerations as well as some general ergonomic and aesthetic considerations integrated in this system.

6.1. The Implemented System

As indicated in Chapter 3, a generative and evolutionary design process consists of two cyclically linked stages

- Design proposal generating stage, and
- Design development stage.

Based on the specified representation and evolutionary mechanism adopted, the process in this system consists of

- Formative generating or construction process, and
- Design developmental process.

Then the software system implementation in this thesis is based on these two stages.

6.1.1. The System Architecture

Based on the program structure of a generative and evolutionary design system shown in Figure 3.2. The main structure of the software implementation system can be divided into five parts:

- Formative generating,
- Design development,
- Evaluation and selection of designs,
- Genetic Algorithm, and
- Modelling visualisation of designs.

Figure 6.1 shows a simple diagram of this system architecture. In this selected design domain, the rudiments are predefined as functional component classes of this product group. The rudiments are built in a database of system in advance. For each component class, corresponding configuration rules are used in the formative construction. Some related design knowledge including ergonomic, aesthetic and DFM considerations is also included in the class definitions. Then the formatives are constructed with the combination of the rudiments and the associated product configuration rules. This generates an initial product model and it is further developed under the given environment. In this system, the environment is externalised as the specified design requirements and design knowledge predefined or input by user through the system interface, which are part of the evaluation and selection criteria of the system.

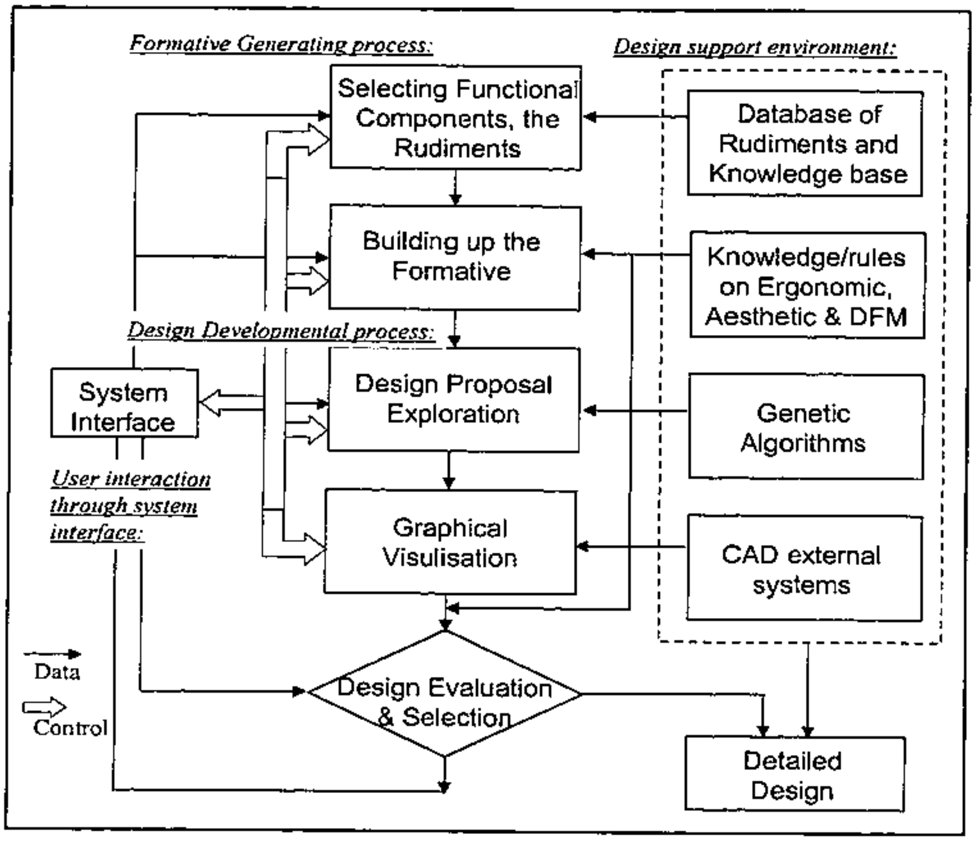

Formative Generating process:
Design support environment:
Selecting Functional Components, the Rudiments
Database of Rudiments and Knowledge base
Building up the Formative
Knowledge/rules on Ergonomic, Aesthetic & DFM
Design Developmental process:
System Interface
Design Proposal Exploration
Genetic Algorithms
User interaction through system interface:
Graphical Visulisation
CAD external systems
Data
Control
Design Evaluation & Selection
Detailed Design

Fig 6.1. A simple diagram of the system architecture.

Based on these five portions, the program of this system is developed in several modules:

- Generating module that builds formatives based on the rudiments selected,
- Design development module that implements the design proposal exploration using GA and the evaluation and selection of design alternatives, and
- Visualisation of generated results integrated with CAD system MicroStation.

The first module works interactively whilst the second and the third modules work automatically.

Genetic algorithms manipulate on the genotype through crossover, mutation and selection operators, while the mapping from genotype to phenotype and evaluation are handled in the development module. The evaluation results or fitness values obtained during the development stage are then transferred to the GA for selection. And GA is used as the engine mechanism in design development process. 3D visualisation of design results is through integration with the MicroStation CAD system.

6.1.2. Program Steps

Based on the program structure shown above, an overall functional diagram of the software system is shown in Figure 6.2. The whole process can be divided into nine steps listed out. Step eight is not outlined which means that it repeats from step four to step seven, until satisfied design solutions generated.

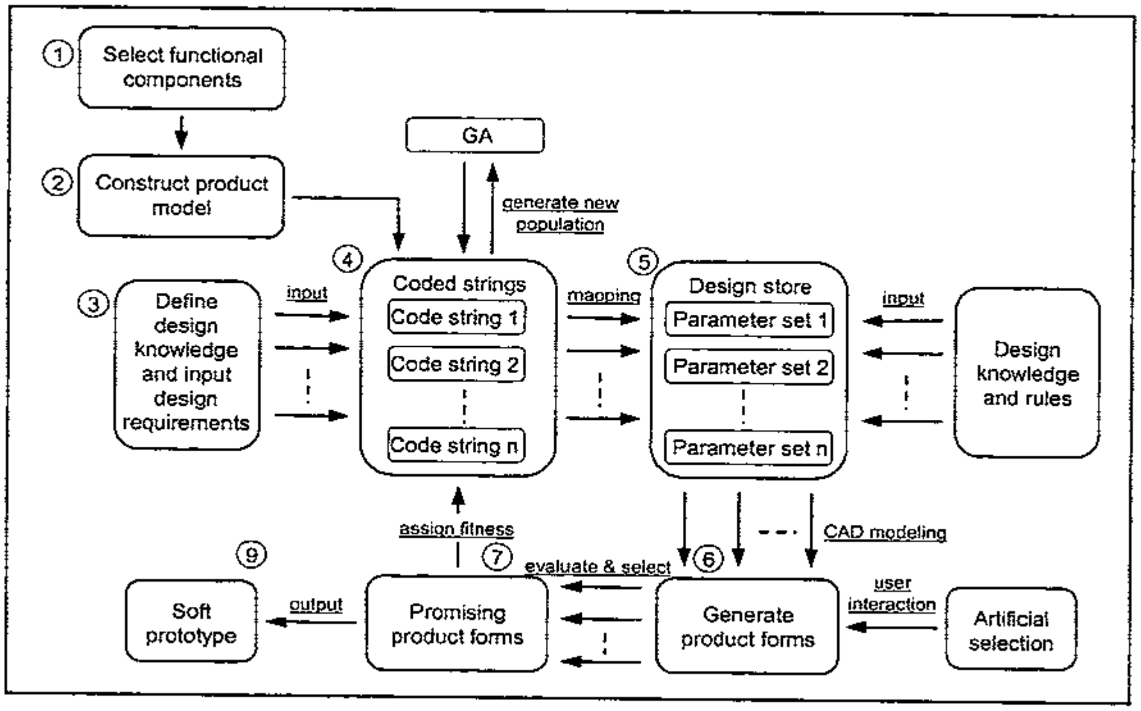


Fig 6.2. Diagram of the software system.

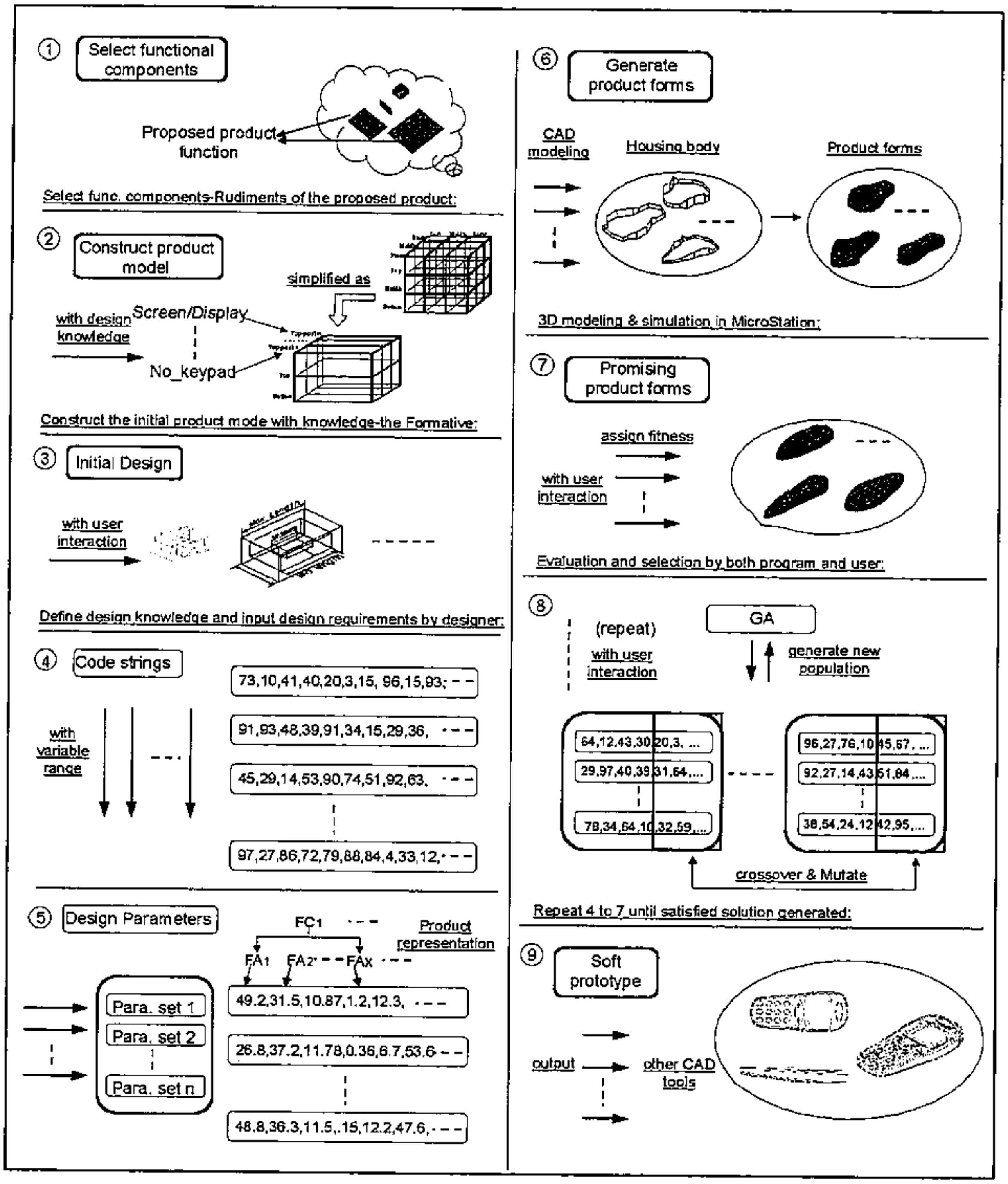


Fig 6.3 Further explanation of the program steps.

Steps outlined in the diagram describe the sequence in which a designer can be supported by the developed system. First at step 1, selecting possible functional components of a proposed product design from a functional component classes library is supported by the system. Corresponding to each selected component class related configuration rules are provided by the system, which form part of the database built in advance.

Next based on the configuration rules, arranging the selected functional components to build the product model is carried out. Here the configuration rules describe the spatial layout of the selected functional components. The configuration rules can be automatically developed by the GA process or generated with ruled-based knowledge and handled by the user manually through the system interface. The later approach is adopted in the later experiments illustrated in next chapter.

At step 3, design requirements and constraints are input by the user through the system interface, which can be further formulated as initialisations and evaluation criteria to be used later by the system. These requirements include the proposed product usage, the maximal and minimal of the product size, etc. In this system, some attributes of the functional components are also provided for re-setting by the user through the system interface. This not only expands the interaction between the system and user, but also provides a more visual understanding of how the algorithm is manipulating the design. For example, main-body styles of a proposed product, the key layout style of the functional keys, etc. can all be changed by the user. More detailed implementation of this step will be illustrated with experiments in the next chapter.

By now the initial product model, i.e., the formative with selected functional components has been defined. Then based on the quantity and the type of the functional components selected, the parameters contained in the genotype of GA are fixed at step 4. The property parameters of GA, such as population size, crossover probability, are set at default values during the initialisation, which can be modified later by the user through the system interface provided.

At step 5, the encoded phenotypes are derived from genotypes. The phenotypes are then transformed and visualised in the form of an actual 3D-product model in a CAD system.

At step 7, the fitness values of each product model of step 6 are assigned and transferred to the GA for selection. At this point the process will repeat at step 8 until the requirements are achieved or stopped by user.

At the end, the design results generated by this system can be directly transferred to other CAD tools for other detail refinement. Also the product models can be sent for manufacturing to make the physical models, which can be sent back to designer again for further testing, evaluation and modification. A more detailed explanation of these steps is shown in Figure 6.3.

6.1.3. The System Interface

The graphic user interface was developed mainly for the control of the system parameters and for the designs to be displayed during the evolution. As shown in Figure 6.4, the graphic user interface of the whole system consists of three main windows:

- Control window,
- Program information window, and
- 3D graphical display window.

These three windows correspond to three main sub programs:

- Control program,
- Evolutionary program, and
- 3D graphical display program.

The engine mechanism of GA was developed in C++ as a generic kernel of an evolutionary program. The evaluation function and the mapping between genotype and phenotype are not included in this GA kernel, because these two are domain specific. Then another main part of the system, the control program, was implemented in Visual basic and was integrated with the GA and Microstation/J CAD system platform. This part provides support for the construction of formative, evaluation of results generated by the system, and a graphic user interface for inputting control parameters. And the 3D graphical display program supports 3D visualisation of the results generated based on the CAD system platform. All programs can run concurrently under Windows 95 on the Microstation/J platform.

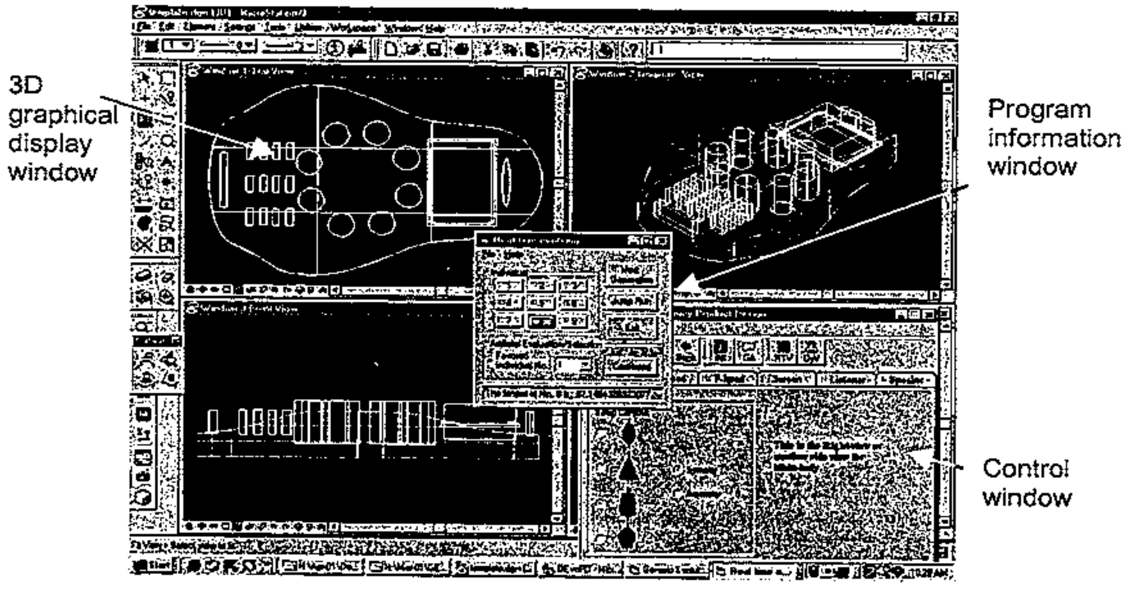


Fig 6.4. The graphic user interface of the system

6.2. Design Initialisation

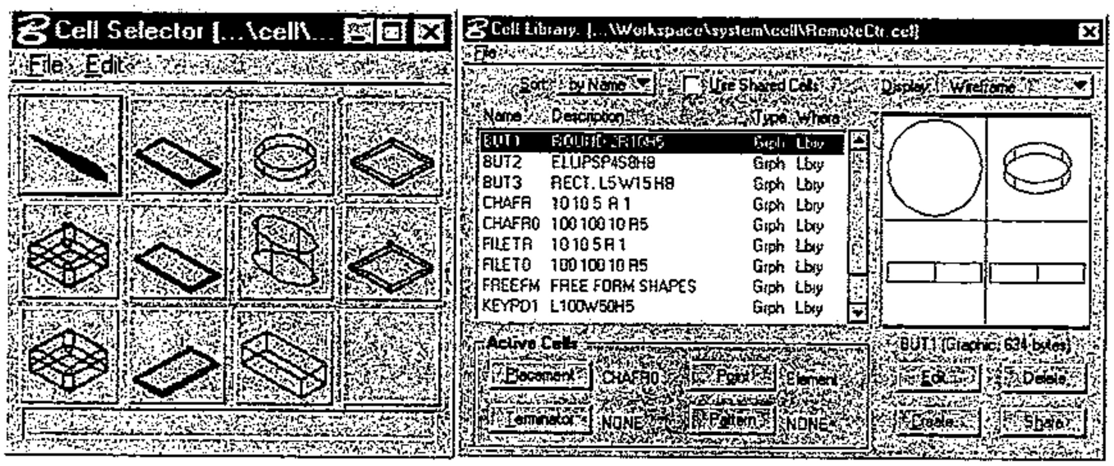
As illustrated earlier, an initialisation of the system must be completed firstly for the generative and evolutionary system to work. Design initialisation task is domain dependent. For example, in the current implementation a database of functional component classes is provided for initialisation. The purpose of design initialisation in this system is to define design requirements and constraints. The system provides a graphic interface for the requirement input and initial settings by user and all these information are transformed and kept in an initialisation file. The system program first reads and parses the initialisation data, which include the definition of the bounds of parameters and some pre-formulated evaluation criteria. Then this is followed by the formative generating process and the automatic evolutionary process.

6.3. The Generating Module

The generating module constructs the formative based on the selected rudiments by user in this system. As discussed in previous chapter, the rudiment defines a set of the functional components and the related design knowledge, and the formative encapsulates the rudiment and relationships, as well as the product configuration rules involved during the formative generating process. Next the implementation of these two are described in detail separately.

6.3.1. Implementation of Rudiment

In this implementation, rudiments are implemented as a set of functional component classes. The program implementation of this in the system is achieved by the class definition using an object-oriented programming. Each class is used to define a group of components with their characteristic factors and attributes, while the instance of each component is created during the evolving process. The attributes include geometric parameters of the functional components as well as parameters related to other feature attributes as described in the former chapters.



Name	Description	Type	Where
BUT1	ROUND 2R10H5	Grph	Lbry
BUT2	ELUPSP4S8H8	Grph	Lbry
BUT3	RECT. L5W15H8	Grph	Lbry
CHAFR	10 10 5 R 1	Grph	Lbry
CHAFR0	100 100 10 R5	Grph	Lbry
FILETR	10 10 5 R 1	Grph	Lbry
FILET0	100 100 10 R5	Grph	Lbry
FREEFM	FREE FORM SHAPES	Grph	Lbry
KEYPD1	L100W50H5	Grph	Lbry

Fig 6.5. The graphic representation and implementation of rudiments

In this system, the graphical representation of the rudiments has been built as cell units in Microstation as a database to support new design. And these cell units are then used as the basic building blocks of the generated product model during the later graphical modelling stage. Figure 6.5 above (left) shows several existing cell-objects and their geometric definitions (right) for the rudiment set in this product group. Some initial DFM considerations are already involved in these cells, so any cell is the combination of manufacturable primitives. When cells are combined to form a complex structure, the rules defined as part of the formative will make sure that they do not violate any manufacturing constraints. Currently this database is small but it is expandable.

6.3.2. Implementation of Formative

The formative is constructed based on the rudiments and predefined configuration rules. Each functional component corresponds to a sub-configuration rule and when several functional component classes are selected, the configuration rules are the combination of the related sub-rules. Based on the product configuration model described in Chapter 3, an initial box-like housing is defined as a main-body of the proposed product model and six surfaces are introduced, as shown on the left of Figure 6.6, i.e., the Front, Back, Left, Right, Top, and Down. A more detailed description about its generic model has been illustrated in Chapter 4.

Selected functional components are placed on these surfaces guided by the sub-configuration rules. A schematic example of component layout on the topside, with proposed four functional components, is shown in on the right of Figure 6.6. FC_i here represents the selected functional components of the proposed product design by a user. Then based on the initial arrangement of the configuration model and initial design requirement, the size of the product is estimated and the bounding hull of the product is determined by the system. The bounding box in the figure shows the maximum size of the product enclosure box based on an initial product size requirement. The values of design parameters representing component features and the exact positions of these features are to be evolved by the genetic algorithm. And the 3D model of any results generated by the system is visualised at a later design modelling stage.

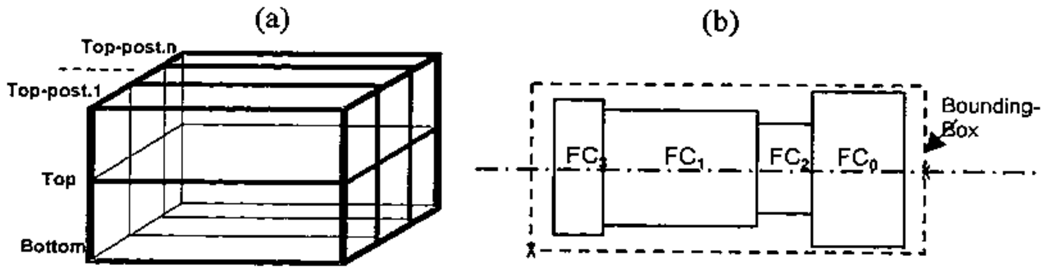


Fig 6.6. (a) The simplified product configuration model

Fig 6.6. (b) The topological layout based on functional components

6.4. The Developmental Module

The design development process develops promising designs through the evaluation/selection, transformation and reproduction of the existing design population, where a GA program is involved as an engine mechanism. Transformed from their code scripts, the design solutions are illustrated and evaluated within the specified environment and context. The evaluation of the design schemas can be made by evaluation algorithms automatically or human user decision. New design solutions inherit features of previous design solutions through reproduction of code script information by crossover. Each population of design solutions contains a small amount of variations due to random mutations in the reproduction of code script information. Then at this point, based on the new design solutions generated, the evolutionary cycle starts again, which will stop when all satisfactions are achieved or through intervention by the user.

6.4.1. Evaluation and Selection

The evaluation is carried out by interpreting the generated design solutions, and by determining their behaviour according to a set of behavioural requirements formulated from the design requirements. Both the selection based on predefined evaluation function, and selection by user interactively are supported by the implemented system. The former is also referred as the natural selection and the later as the artificial selection (Frazer, 2000).

The term natural selection is used in this process to define the evaluation and selection by the fitness function, which is analogous to natural selection during natural evolution. As described earlier, the evaluation of a design based on various evaluation criteria in this approach is a multiobjective problem. The evaluation criteria involved at this stage are mainly derived from the considerations on aesthetic, ergonomic and manufacturing factors. The detailed formulisation of these considerations will be presented in chapter 7 with examples.

Besides the natural selection described above, the artificial selection by the user is also supported in this approach, which means that during the design development process the designer or user can credit their favoured designs and thus guide the system to evolve promising designs. In this system, this is achieved through the grading of each individual by the user, which is then treated as the fitness value for GA. The user decides which one among the individuals is the better or promising one.

6.4.2. Genetic Algorithms

The genetic algorithm in this system is developed as a generic kernel of an extendable evolutionary program. This algorithm is manipulated on the genotype through the crossover, mutation and selection operators. The mapping from genotype to phenotype and evaluation is handled by the program of development module. And the evaluation results are then transferred to fitness values of GA for selection. A genetic algorithm in the system is composed of four main operators: selection, crossover, mutation and crowding.

In this system, the fitness value, derived from the evaluation, of each individual is treated as its strength. Summing up the strength over all individuals, the operator has a complete strength. Using this, the percentage of individual total strength is calculated to select individuals. For example, when an individual strength is 20.5 percent of the total, this individual has a 20.5 percent possibility of being selected at each selection.

After selection, the crossover comes to be the next step. A pair of strings undergoes crossover as following: an integer position k along the string is selected uniformly at random between 1 and the string length less 1 $[1, l-1]$. L here represents the length of genotype string. Two new strings are created by swapping all characters between position $k+1$ and l inclusively. A detailed example about this operator has been described in last chapter.

To keep the algorithm from getting stuck at a significantly less-than-optimisation answer, perhaps because of an unfortunate choice of starting configurations, the genetic algorithm used includes a mutation operator. Used sparingly, this operation randomly changes a digit in the string of digits that make up genetic information of individuals. In this system, one mutation per five hundred bits is set as default. In integer coded genetic information, this simply means changing the integer value within the given range. However, the mutation operation plays a minor role compared with the role-played by the crossover operation, but it contributes to the diversity of design alternatives.

After the crossover operation with mutation, two individuals are born, this means that two individuals have to be selected to replace them. In (De Jong, 1975) crowding, an overlapping population is used where individuals replace existing strings according to their similarity. An individual is compared to a random sub-population of crowding factor members. The early individual with the highest similarity, on the basis of bit by bit similarity count, is replaced by the new individual. The replacement of individuals by similar individuals tends to maintain diversity within the population and reserve place for two or more "species". In this way, first, three populations are picked at random from the full population set, and the worst strength individual among those selected is chosen. This process is repeated for three times. Thus there are three or less (when the same individuals are selected twice) individuals for the crowding operation where the highest similarity individuals can be replaced by the new individuals.

6.5. Graphical Modelling Visualisation

The system implementation was carried out using the Microstation/J CAD system as a platform, and with Visual Basic and C++ as development languages. It was proposed that the implementation of the prototype system should utilise the existing software as much as possible, thereby allowing maximum time to be spent on the new areas of research. Earlier testing of this system implementation has been done on the AutoCAD system using Visual Lisp language (Sun, 1999). And after reviewing several CAD systems, MicroStation was chosen for the following advantages highlighted below:

Bentley Systems have been moving their focus from task-oriented programs to a more holistic process-orientated system. MicroStation provides advanced computer-aided design on PC. Microstation/J is good for the surface modelling and solid modelling. For surface modelling in Microstation/J, complex 3D models can be created using the enhanced Parasolid-based SmartSurface modelling creation and modification tools. When applying surface intersections, MicroStation/J creates exact geometry such as true B-splines, circles and arcs, facilitating the dimensioning of intersection geometry. For solid Modelling, MicroStation/J now supports basic solid modelling operations using the Parasolid modelling kernel. Users can quickly design complex models using 3D Boolean operations such as unite, intersect and subtract. This new SmartSolid technology enables users to modify solid models by editing or removing faces. So the 3D solid modelling tools in MicroStation/J are ideal for conceptual modelling and visualisation.

MicroStation provides user-level programmability with MicroStation BASIC, C and Java. MicroStation BASIC is a very simple way to improve your productivity by letting the user automate often-used sequences of operations. The user doesn't need high-level language development skills. Among other things, tools and view controls can be selected, key-ins can be sent, dialog boxes can be manipulated and objects can be modified. The obvious advantage of writing a macro to perform a task that could otherwise be done manually is to automate mechanical and repetitive tasks. With the source codes generated by the Microstation Basic macro, the user can build their own program earlier and compiled this into their own systems.

As mentioned before, the evolutionary algorithm, the GA kernel, is developed by C++ and other sub programs are in Visual Basic. And they are complied into a whole system when executing. The GA program is combined into a software sub-system treated as an object in VB. Some simple program lines are shown in Table6.1. The graphic modelling and visualisation program created as part of this work continuously shows the three dimensional images of the designs.

Table 6.1 program implementation lines of the sub program combination.

```
Public GAApp As Object
Set GAApp = CreateObject("GA.GA_INT")

; Initialise the population
geno = GAApp.GARCInit(Popsize, XoverPro, MutPro, Range, Lgen, Null)

; Get the new population
geno = GAApp.GARCEvolve(fitness, Popsize)
```

As for data exchange, MicroStation offers ultimate flexibility by enabling exchange of virtually all industry standard file formats such as DGN, DXF, DWG R14, IGES and STEP. MicroStation/J can also publish engineering data in standard Internet file formats such as CGM, SVF, VRML, JPEG and HTML. So files generated in this system can be easily transformed to other CAD systems for further refinement, for example, the ProE.

6.6. DFM Considerations

In this approach, design and manufacturing knowledge defines the varied constraints of design and manufacturing guidelines and rules of the proposed certain manufacturing methods. Especially the early anticipation of manufacturing problem at the conceptual stage of the design process is considered important in the whole process. Based on this knowledge, the idea is to build or construct the manufacturable rudiments and then the formatives in the first place. Through this way, the DFM considerations are involved at the early design stage and made effective throughout the whole design process.

DFM issues are different when different manufacturing processes are involved. At this stage, the consumer electronic product design has been taken as an example. So only injection moulding manufacturing method has been considered. The generative and evolutionary product design process developed in this system followed the general manufacturability guidelines to avoid problems downstream in the detail design process. It did not concern with detailed mould design. A list of relevant injection moulding manufacturing issues at the initial concept design stages is given below. This is only a subset of guidelines developed from Bown (Bown, 1979) and Boothroyd and Dewhurst (Boothroyd et al., 1983 and 1994). The list is used as a basis from which specific product configuration rules are formed.

- Aim for simple mouldings (avoid high precision surpass the requirement and complex structure),
- Design generous draft angles to aid extraction,
- Design for a flat parting plane,
- Design parting lines to avoid unnecessary tolerance,
- Design generous corner radius to avoid flow problem,
- Avoid very large flat surface to reduce warpage,
- Design generous draw or cavity to avoid warpage,
- Avoid depressed or raised features on curved surfaces,
- Avoid side holes to reduce mould complexity,
- Avoid closely spaced oblique holes to limit freeze marks,
- Design for easy flash removal (tumbling), and

- Avoid undercuts.

Among these guidelines, some really lead to conflict between the designer and manufacturer, because of the separation of design and manufacturing as illustrated in Chapter 2. For example, sometimes the designer draws out beautiful and novel product forms with elegant and changeable curves. But these curves may be seen as very complex surface curves by engineers from the viewpoint of manufacturing. When the design is transferred to the engineers, many changes or even total refusal may happen. The engineers always desire for simple and reasonable manufacturable shapes according to their experience, which always disappoint the designers much. Then in this case, the designers cannot get their design manufactured no matter how this design is claimed to be wonderful.

As mention above, this system is intended for the DFM considerations to be formulated at a high level, so called common sense, to try to achieve a better compromise between the designer and manufacturer. For example, this system only considered the housings that can be made in a straight draw two-part injection mould with no side holes. And so all housings will have a straight parting line. A simplified sketch of this type of mould and a typical part from such a mould is provided in Figure 6.7 below.

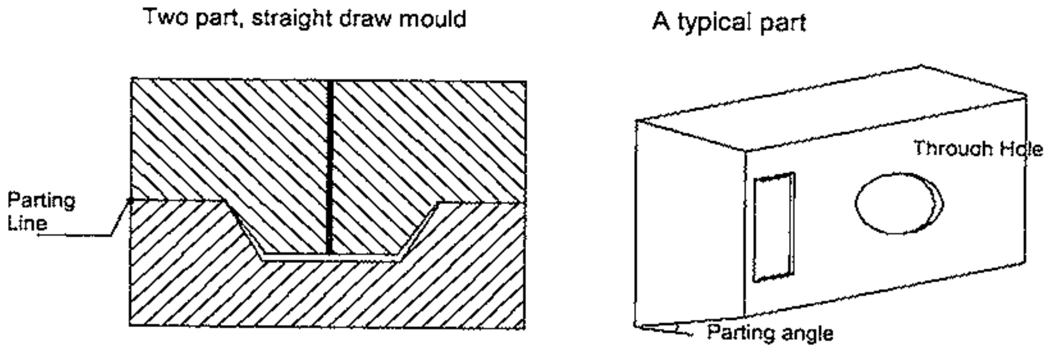


Fig 6.7. A typical part from a straight draw mould with no side holes.

As shown in the figure, the parting line, parting plane and the drafting angle are some basic and very important factors for DFM consideration. Other factors like the curvature of the profile surface, the hole spacing on keyboard are also needed to be involved, and so on. Since DFM issues are treated as different problems at the different design stage of the whole process, the DFM considerations and guideline rules in this generic evolutionary design process developed are separated into two main aspects of the component layout and evaluation criteria during the design generating and developmental stages.

6.6.1. DFM in Generating Module.

During the formative construction stage, after the functional component classes of the product are selected, they are then put on a box-like housing surface. And some of the DFM guidelines are applied to this process as constructional rules. For each functional component class, a rule base is predefined by the system and the program selects and applies these rules until the current component is fully positioned.

Taking a hand held product with a horizontal-parting plane as an example. A visual display panel should face the top of the product both for ergonomics (so it can be viewed) and for manufacturability (to avoid slide holes in the mould), and similarly the speakers are restricted to the top of the product. The definition of hand held products is limited to those used while being held in one hand so a horizontal orientation is preferred to a vertical parting plane for hand held products. Here a simplified rule base of the component class layout on the outward faces of the proposed product is adopted. It is assumed that all the select component classes are put on one main side of the six surfaces of the box. So the topside is the main surface in a hand held product and front-side is the main surface in a desktop product. Examples of the rules are illustrated below. A typical position rule in the acoustic (loudspeaker) rule base and a rule about the manufacturing constraints is constructed as follows:

Table 6.2 An example of position rules with DFM consideration.

```
If  (acoustic component) and (product parting line horizontal) and
(product use hand held)
Then
    Component main surface faces the top of the product (product main face)
End if

If  (component main surface is parallel to the parting plane)
Then
    Constrain the component to the parting line
End if
```

Here the ergonomic and manufacturability considerations are integrated at the same time, since in many instances deciding a rule classification seems to be a little arbitrary. Therefore, only the component class defines rule bases. This approach helps to avoid redundant rules. When new product group and new functional component classes are added, new rules corresponding to each new component class will be added.

6.6.2. DFM in Developmental Module.

The DFM issues are also incorporated in the fitness function and evaluation criteria for the improved GA in this developed system. The DFM issues can be formulated into different design requirements prospected by the user and evaluations criteria through system interface.

Two examples are used here for illustration. If a product is supposed to be hand held usage, then the maximum and minimum size range of the housing will be defined for evaluation during the generative process and also the user can input their favoured range through the system interface. Also, the surface and shape curvature control is a very important element for the injection moulding products, which is derivable from the general DFM considerations listed above, such as design generous corner radius to avoid flow problem and avoid depressed or raised features on curved surfaces. A simple example, where the curvature of the profile surface is treated as the evaluation function of the GA, is shown in Chapter 3. Embodying this in the product housing shape description, the system provides four main styles of profiles for each side view of the proposed product shape, which are manufacturable. And then the system generates varied shapes automatically or generates the promising shapes according to the profile styles selected by the user through the system interface.

6.7. Summary

This chapter presented the implementation of a product design support system based on the generative and evolutionary design process proposed in Chapter 3. In the implementation of the system, product design is selected as the example domain and injection moulding is treated as the proposed manufacturing method. And these two things defined the context in which design issues including DFM issues are addressed by the implemented system.

This chapter has described the detailed implementation of the system developed in this thesis. The main structure of the software implementation system consists of five parts:

- Formative generating,
- Design development,
- Evaluation and selection of designs,
- Genetic Algorithm, and
- Modelling visualisation of designs.

Further detailed description of each part in terms of application and the experiments will be provided in the next chapter.

Chapter 7

7. Applications and Experiments

In previous chapters, much has been talked about the implementation detail of the computer model of the proposed generative and evolutionary product design system. In this chapter the results of the experiments conducted using the developed system are reported. These experiments were carried out to demonstrate the capability of the developed generative product and evolutionary design support system. The applications and experiments focus on the domain of consumer electronic products.

7.1. Introduction

The general goal of the experiments described in this chapter was to demonstrate how the generative and evolutionary deign process proposed in this thesis works in a real design situation. It is necessary to demonstrate that the system is capable of generating diverse alternative design solutions for design concept exploration, as well as integrating the DFM considerations and other design knowledge at the early design stage. In addition, it is necessary to evaluate the benefits of generating design solutions by artificial selection through user interaction. Therefore the main objectives of these experiments are:

- To demonstrate that the system is effective and capable of producing diverse design solutions during its runtime using the concept of rudiment and formative,
- To illustrate the exploration and adaptation ability of the genetic algorithms in generating potential design results, and

- To establish how to use different representations for the DFM considerations and design knowledge and where this knowledge comes in during the evolving process.

In order to achieve each of these goals several experiments were carried out. The different experiments are resulted from using different initialisations, design requirements, and evaluation criteria settings during the design process.

7.2. Use the Implemented System

As indicated above, the consumer product design was selected as a design domain and a basic description on the system architecture and program steps have been given in last chapter. Before the illustration of using the implemented system, some pre-stage work built in advance needs to be introduced first.

The system has been primarily implemented for the application of generative and evolutionary techniques in the domain of the consumer electronic product design. For generality, the implemented system can be used to support the design based on a generic model of the consumer electronic product, including remote controller, mobile phone, calculator, radio and television etc. Geometrically, all these products have box-like housing or enclosures, and almost all the interface components are distributed on the outside surfaces. Figure 7.1 schematically shows a hierarchical representation structure of this product group based on the functional components in this domain, which can be classified as:

- Number keypad,
- Functional keypad,
- Screen/visual display,

- Acoustics/loudspeaker,
- Microphone, and
- Battery box, PCB etc.

The classification of the component classes is based on the product functional decomposition as described in the design representation section before. And the functional component class set is limited to the six component classes listed above in the implemented system. The component classes are represented by their related geometric and feature attributes, which will then be encoded and manipulated by a GA program.

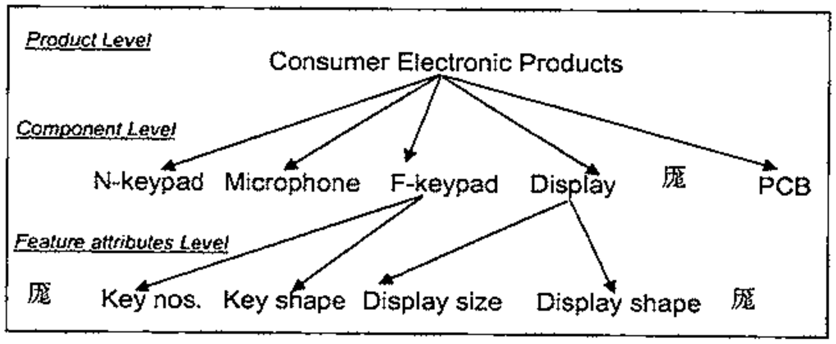


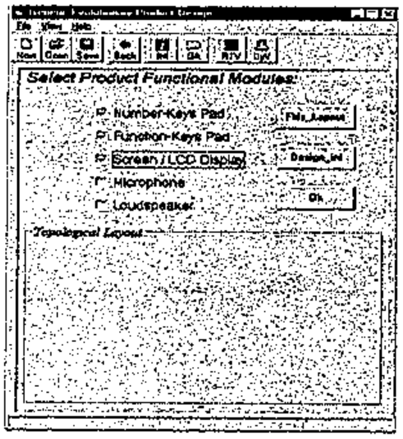
Fig 7.1. A product decomposition based on functional components

In this implementation, in order to enhance the system's supportability for product design process and the interaction with user, some main design attributes of the proposed functional component classes are also provided by the system interface for the user. The user can guide and adjust the design-evolving tendency, which is automatically directed by system as default, so that the user can participate to provide effective support. The design evolving process is described briefly by introducing the system's graphic user-interface next.

The first step is to select possible functional classes of a proposed product from a functional class library provided by the system. This step is carried out through the interface, Figure 7.2a. The user can click on the check-box to decide which functional component class to use to define the proposed product. The next step is to arrange the selected functional components on the product surface using ruled-based knowledge provided by the system. This step is carried out by clicking the checkbox of position displayed in the other half of interface as shown in Figure 7.2a-b. One can click on the crossed (selected) checkbox again to change one's former settings before one starts the next step.

Other initial requirement input by the user is supported by a sub-interface, that is, the *Design_ini* panel provided by interface, shown in Figure 7.2c, such as maximum and minimum of the product size, style selection and other constrains which form part of the evaluation criteria. Additionally, different main-body styles and other related attributes of each functional component class are pre-set and selected by the user. Through this way, some attributes of the functional components, which are manipulated by GA during the evolving process, can be controlled partially by the user, so that more interactions with user can be involved during the design evolving performed by the system. This step is carried out in an interface as shown in Figure 7.2d and Figure 7.2f.

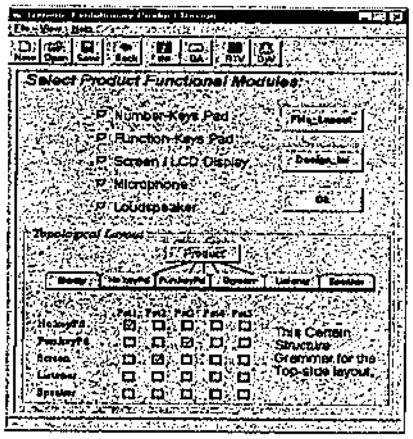
Till now the initial layout of the proposed product with selected functional components is defined. Based on the quantity and the type of the functional components selected, the parameters contained in the genotype of GA are fixed. Each functional component is represented by different feature attributes groups as illustrated above. Then graphical product models are automatically generated and evaluation and selection based on these results are achieved during the operation of the system program illustrated in last chapter. The promising or fitter results are selected and transformed into the new population manipulated by GA. From this point the process will be repeated until the end condition is satisfied or the process is stopped by the user. Next, detailed explanation on these steps will be provided with experiments testing.



Select Product Functional Modules:

- [✓] Number-Keys Pad
- [✓] Function-Keys Pad
- [✓] Screen / LCD Display
- [ ] Microphone
- [ ] Loudspeaker

(a)

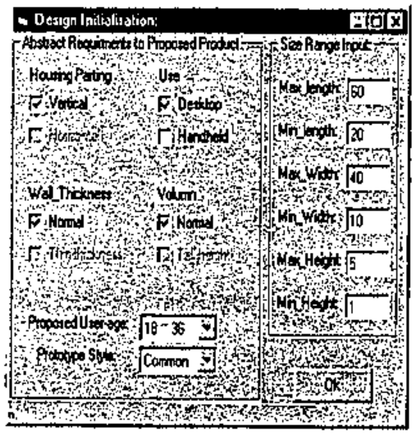


Select Product Functional Modules:

- [✓] Number-Keys Pad
- [✓] Function-Keys Pad
- [✓] Screen / LCD Display
- [✓] Microphone
- [✓] Loudspeaker

This Certain Structure Grammar for the Top-side layout.

(b)



Design Initialisation:

Abstract Requirements to Proposed Product

Housing Parting	Use
[✓] Vertical	[✓] Desktop
[ ] Horizontal	[ ] Handheld
Wall_Thickness	Volumn
[✓] Normal	[✓] Normal
[ ] Thinthickness	[ ] Tallness

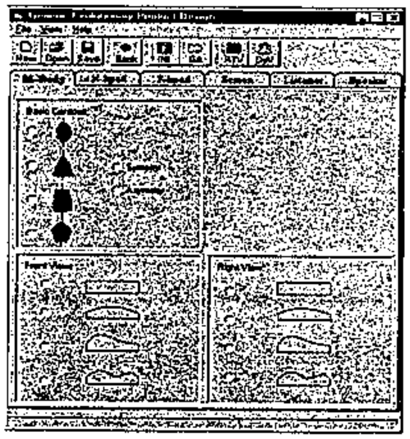
Proposed User-age: 18 ~ 36

Prototype Style: Common

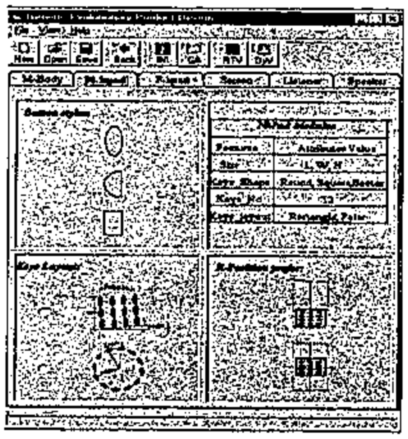
Size Range Input:

Max_length:	60
Min_length:	20
Max_Width:	40
Min_Width:	10
Max_Height:	5
Min_Height:	1

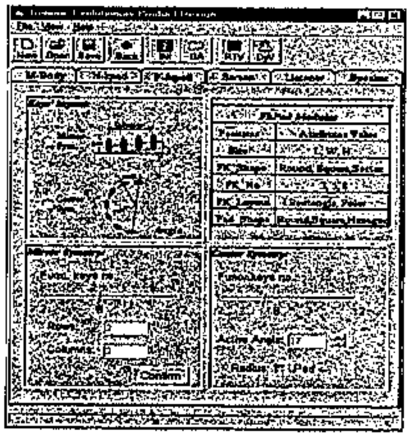
(c)



(d)



(e)



(f)

Fig 7.2. System graphic user interface.

7.3. Experiments on Mobile Phone Design

As described earlier, product forms can be different dramatically even with the same functional parts. With differences in product sizes, proportions and configurations of the product model, the same design concept may result in diverse design configurations. The problem of evolving a mobile phone was the first design task presented to the prototype system. This experiment used a simple and real coded GA with multiobjective capabilities. For the different evaluation functions involved, the user gives the each element its weight, which defines how important this evaluation criterion is in the total fitness function through the system interface. Then the different weight distribution guides the system's evolving direction and leads to different design solution area.

The GA used in this experiment had a population size of nine individuals, a probability of 0.2 for crossover, and a probability of 0.01 with which a gene in a chromosome can be mutated. Each design was represented using the selected six functional components. The GA was allowed to evolve designs for up to two thousand generations, which is pre-set as the maximum generation here. As illustrated in Chapter 5, some parameters of GA properties are provided for user setting by the system.

7.3.1. Formative Generating with Initial Settings

As described earlier, the formative definition in this system consists of the definitions of its composite components (the rudiments) and the associated configuration rules for a complete product. And the configuration rules are described as the spatial layout considering the relationship of the product functional components. These spatial layouts can also be generated and evolved by GA.

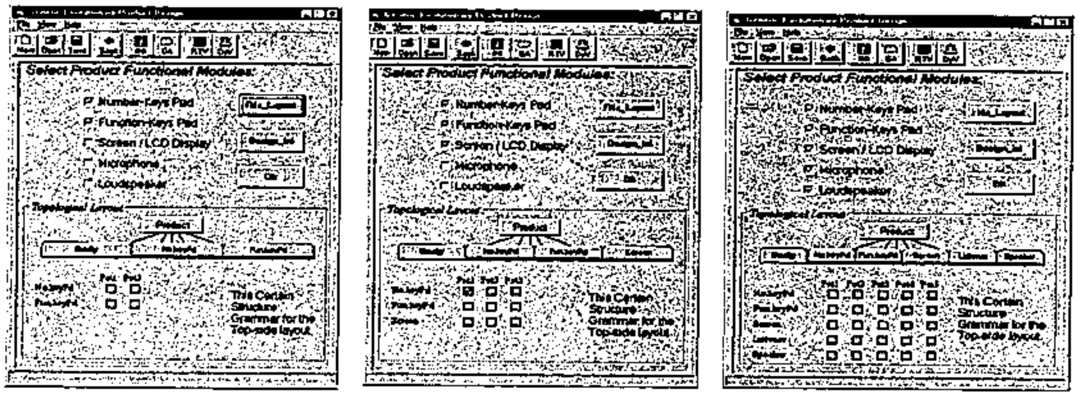


Fig 7.3. The interface of product topology setting provided by the system.

In this example, five functional component classes are selected for constructing the formatives in this application. They are number keypad class, functional keypad class, screen/visual display class, acoustics/loudspeaker class, and microphone/listener class. The configuration rules involved are simplified as the topological layout of these selected functional components on the main surface of the product. Since all the functional components are proposed to be organised on the parting plane of the proposed injection-moulding model of product and their main upside is put on the top surface of product for ergonomic considerations. As illustrated earlier, in this part the product model is simplified as a box-like model and is represented by its six outside faces. Then the selected functional component classes are distributed on the main outside of the box shape, and then all possible layouts are generated by the system. As shown in Figure 7.3 different topological layouts can be developed by the users manually using the graphic system interface provided at this stage.

When selected functional component classes are proposed to be put on the main outward side, then this box-side is divided into parts equal to the selected component number P_i ($_i$ is the number of the selected functional component) and the layout sequence or position of the components is built again. For example, NkPad and FkPad are selected and both are put on the Topside. Then the top surface of the bounding box is divided into two parts (P_0, P_1), so the NkPad is positioned either at, P_0 or P_1, and the FkPad is put on the rest position. The more the functional component classes are selected, the more possibilities of the layout on the topside will have. The displacement of the listener and speaker is decided by the rules predefined by the system. The displacement of other component classes is manually handled by user through the system interface. Then with this position setting, the modelling visualisation can be achieved with the evolved values of component parameters.

Then at the design initialisation step, different initial or design requirement input will lead to the introduction of different evaluation function. At this step, some initial requirements are specified by the user through the system interface, such as the maximum and minimum of the product size, style selection and other factors related to manufacturability consideration. These initialisations incorporate some early general considerations of aesthetic, ergonomic and DFM. They form part of evaluation criteria later during the evolving process. In this example, the evaluation modules considered are:

- Size (user input the range),
- Style (the main-body shape type selected through the interface),
- Manufacturability (such as keys spacing range for the NkPad and FkPad), and
- The grades assigned by the user on the design alternatives (use for artificial selection).

To make the system more effective for supporting the design process, an interface of design requirement setting is also provided for the initialisation. The design initial interface provided by this system is shown in Figure 7.1c. Since all these products have box-like housing or enclosure and there are several major general characters of these products to be considered:

1. Injection moulding is assumed as the primary manufacturing process for the product where parting–plane is the most important design and manufacturing feature;
2. The product housing is divided into two parts along the parting plane with desired orientation, i.e., vertical or horizontal, and all the product components can be seen to be arranged on this plane; and
3. Product usage is either Handheld or Desktop.

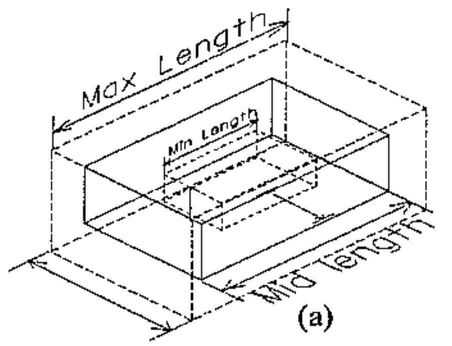
So the requirement input according to these factors are represented as six items:

- Housing parting –Vertical or Horizontal;
- Usage -Desktop or Handheld;
- Wall thickness-Normal or thin-thickness;
- Volume-Normal or Tall-Height;
- Proposed user age- Main three ranges; and
- Proposed product size.

7.3.2. Evolving Designs Using the System

It is proposed that each evaluation function involved is a combination of the aesthetic, ergonomic and design for manufacturability considerations. In the experiments carried out, DFM evaluation criteria were not separated from other evaluation criteria.

An important characteristic of the design of any consumer product is its size, in general for handheld or desktop usage. So the first criterion for the evaluation of the software to be used was the ergonomically desirable size with both aesthetic and ergonomic considerations. The users specify the size range, the maxim and minim volume extents, of the desired product during the design requirements input. With the size range input by user, the system can define the reasonable product size range as shown in Figure 7.4a and it is then used for product size initialisation of the program, as seen in Figure 7.4b. Running the system with this single evaluation criterion generated the designs with reasonable size. Figure7.4c shows the generated results under one desired product size input by the designer.



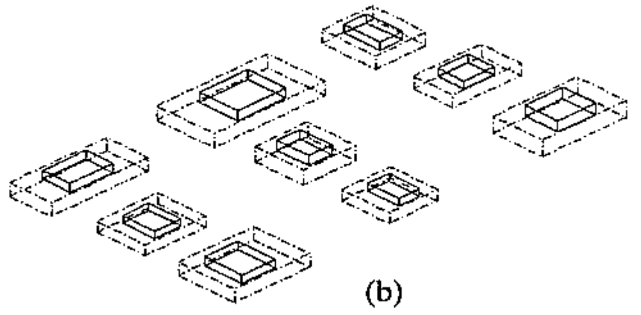


Fig 7.4. (a) Representation of the reasonable size.

Fig 7.4. (b) Initial product size generated by system.

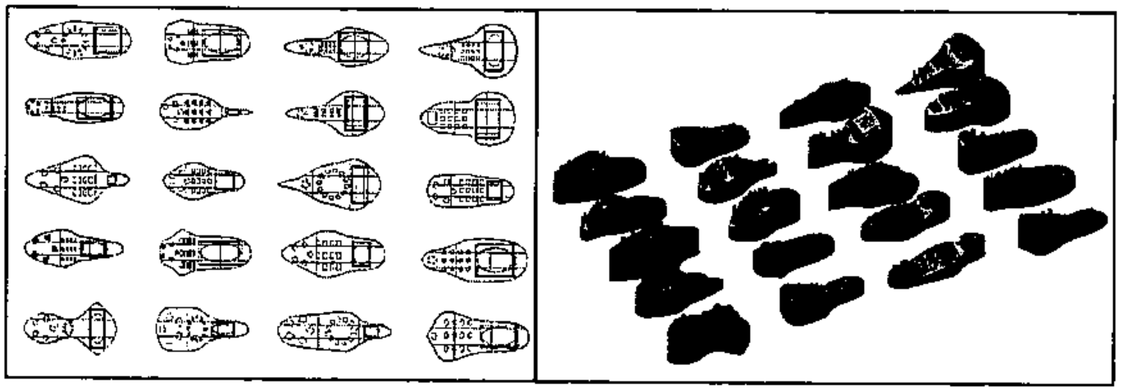


Fig 7.4. (c) Design results of a desired product size.

Another important characteristic is the main body style of the product shape, especially for the consumer products. The appearance of a product is the consequence of the designer's choice of structure, form, material, dimension, and surface treatment. And as described earlier, even with the same functional components, the designer could still design dramatically different product shapes. Through the study of aesthetics, it is possible to identify general common characteristics of attractive products. Hence, the second criterion of evaluation software to be utilised was the 'favoured style' criterion.

Based on the classification of some existing basic product forms or shapes, a new approach to describe a proposed product form was adopted in this implemented system. Any 3D form can be described by 3 projection shapes, in x-y, x-z and y-z plane. In this way, any product forms can be derived from the projection shapes defined in this system. As shown in Figure 7.2d, there are four types for main projection shapes and four types for side projection shapes. The four types in two-side planes are the same. Figures 7.5 shows some examples to each type of the main-body style. Also the aesthetic, ergonomic and DFM considerations are involved in this definition. Then the selection of the main-body style leads to the other sub-evaluation function, which are formulated by the system.

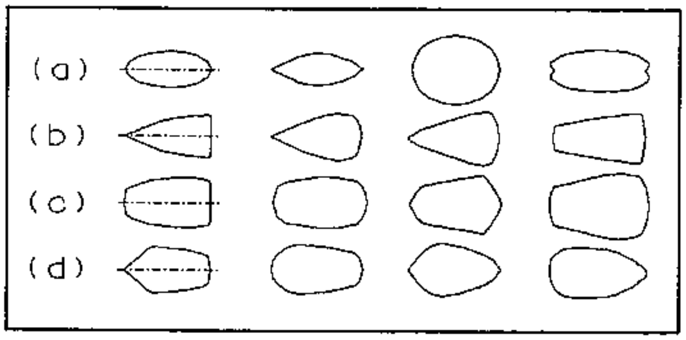


Fig 7.5. Main-body styles based on simple classification

The designs produced by the system with these two criteria were always the desired size and desired main body style. Some design results generated by the system with different design intent on the product size and main body style are shown in Figure 7.6. And running the system with this evaluation criterion, the design results with each main body style are shown in Figure 7.7, and the corresponding rendering results of these designs are shown in Appendix-B (Figure 2-5).

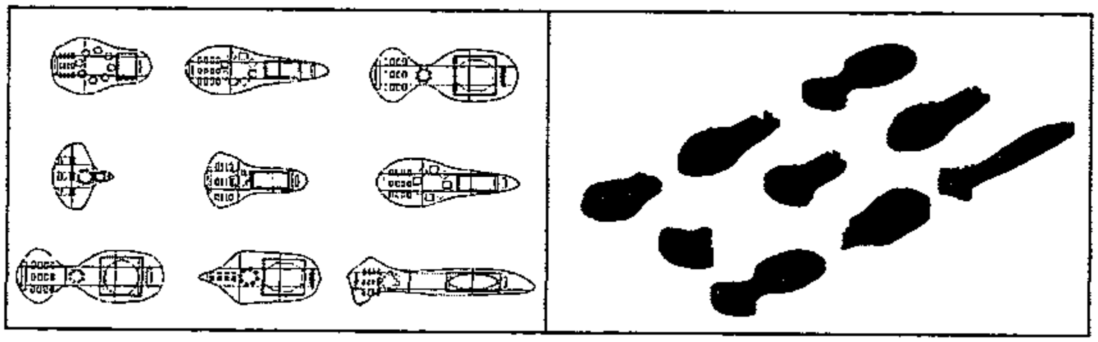


Fig 7.6. Design results of different Main-body styles. example-(1)

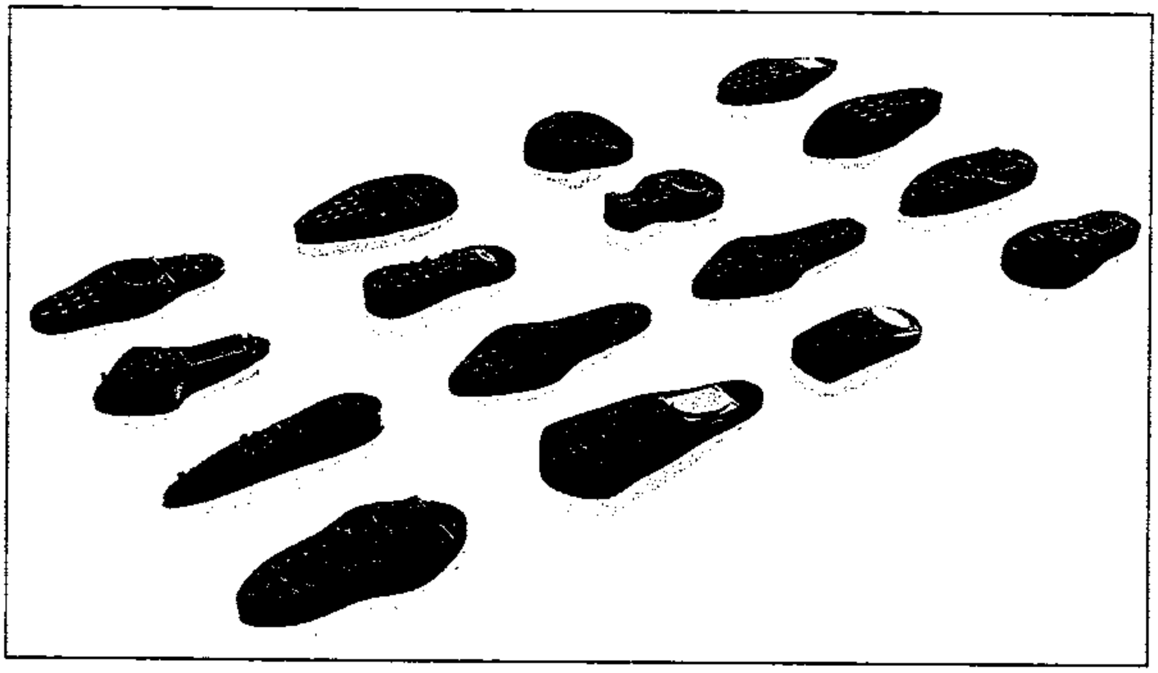


Fig 7.6. Design results of different Main-body styles. example-(2)

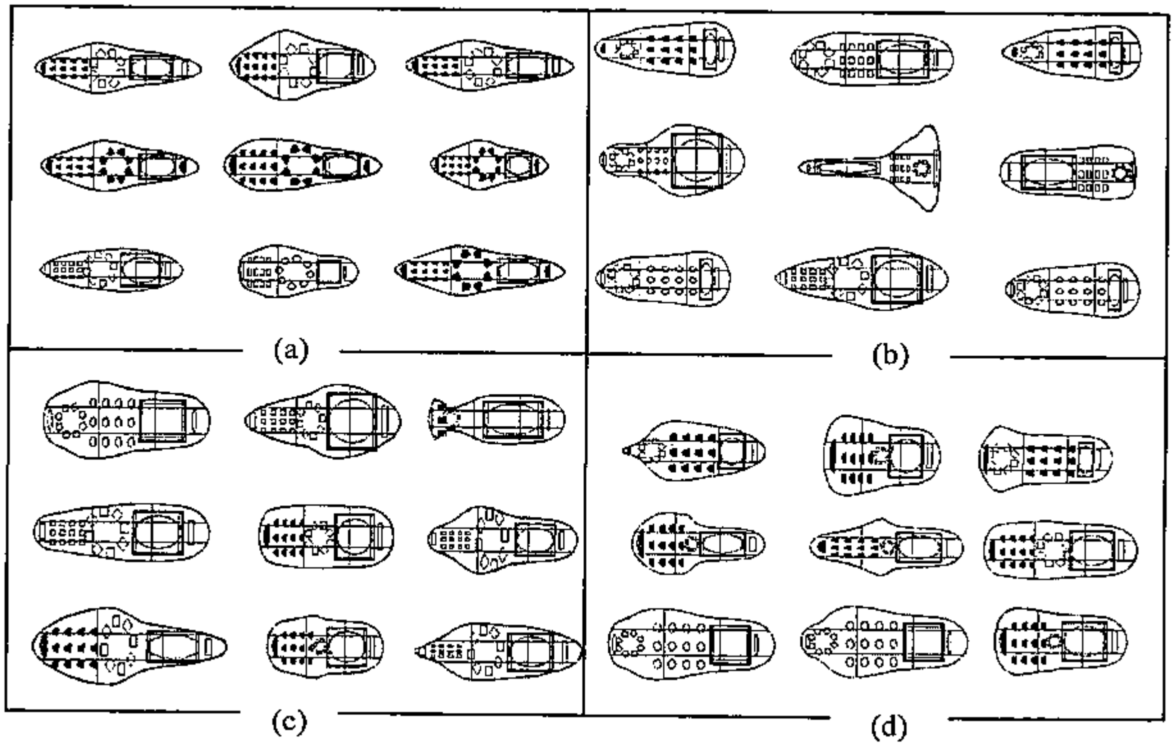


Fig 7.7. Design results of each Main-body style.

However, as described in the system implementation section, the desire for the solutions of real world design problem will be involved in this system. Since the consumer product design is selected as the example domain, one of the important considerations at the very early design stage for this product is the objective of the design, or the target user of the design. For different users the desired designs are different. For example, in a simple market survey on consumer for the commercial mobile phones, normally the young people, aged from 18 to 35, like fashionable or novel, such as Motorola 3688 and Nokia 8088. For the same product sets, the old people aged above 60, like to choose Motorola 978 etc. So the user age is treated as the third element of evaluation software to be used for defining the "desired specialisation". In this system, this is formalised as an evaluation function, which determines the ranges of the number key size and screen size, and describes the different design intent tendency.

Also during the design evolving process, more interactions between the system and user are provided. For each functional component class, some attributes of the class and some vital attribute characters of the design are provided for the user. For example, the system provided different key layouts for the number keypad and functional keypad as shown in Figure 7.2 e-f. Besides the system can automatically evolve designs, and the user can also set these items to guide the design evolving.

7.3.3. Results and Further Refinement on Designs

As illustrated in the system implementation part, MicroStation/J CAD system offers ultimate flexibility by enabling exchange of virtually all industry standard file formats such as DGN, DXF, DWG R14, IGES and STEP. So files generated in this system can be easily transformed to other CAD systems for further refinement, for example ProE, Inventor etc. Figure 7.8 shows the graphical user interface provided by the system program for picture file save. Based on the Microstation platform, the result file of this system can be saved as DGN format as default, then through "Export As" command, the result file can be further transformed into other standard file formats, which are also provided by the Microstation platform. Figure 7.9 shows a refined generated product design using ProE. Its rendering picture is shown in figure 7.10 and more rendering pictures of the design results are shown in Appendix-B (Figure 6-9). And these rendering files can be directly sent to manufacturing machines for making physical models.

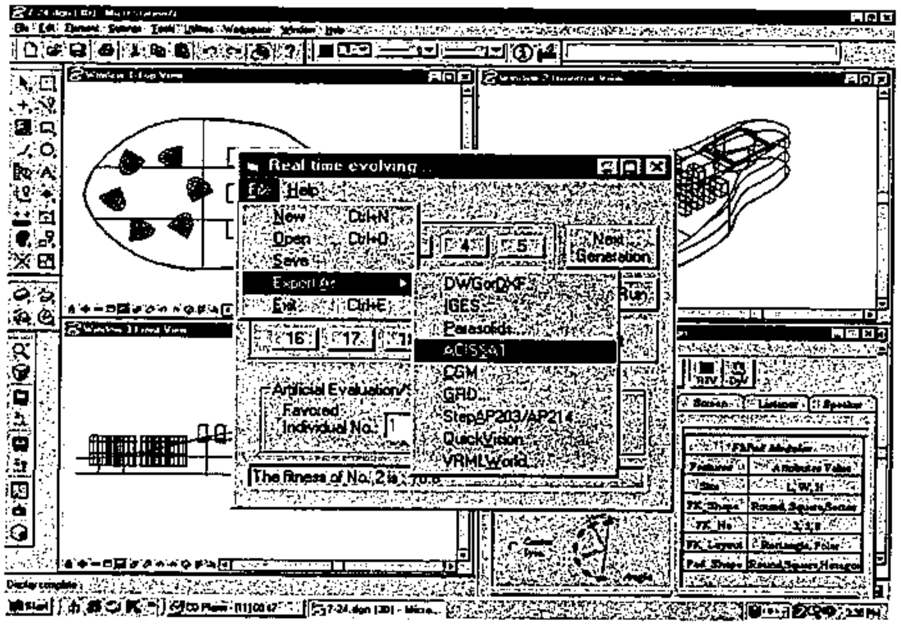


Fig 7.8. Graphical user interface of system for file formats transfer.

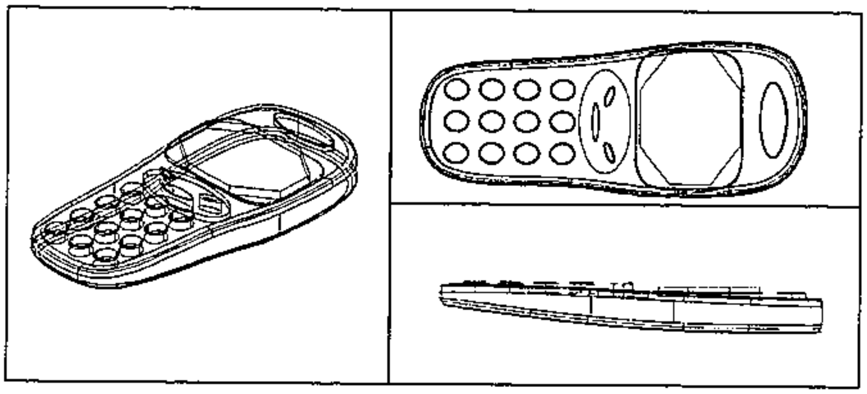


Fig 7.9. A refined design result by Pro-E

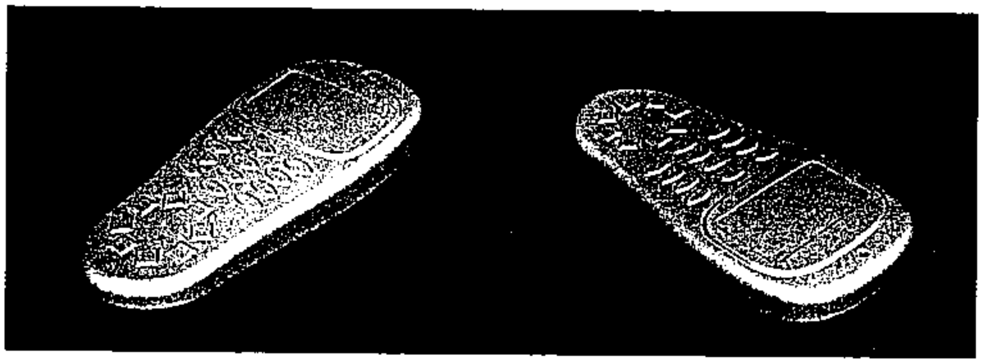


Fig 7.10. A rendering design result.

7.4. Other Examples

For the different formative construction and design initialisation, diverse design solutions can be generated by the system, which means that different initialisation settings and design requirements input will lead to different design formative construction and evaluation function, thus generating various design results. Next several experiments are tested out under different initial settings and evaluation criteria.

7.4.1. Evolving Controller-like Product

The most obvious difference of this product design from the above one is that two functional components are involved in the formative construction. They are number keypad, functional keypad. Then based on different setting on initialisation requirements and attribute parameters, diverse design alternatives are generated by this system. For example, the built layout of the functional components is NkPad (P_0), FkPad (P_1). And the Number-keys is selected to be array-layout style. Function keys' number is set to 8 and to be in a centre-symmetry layout style. Then for different main-body styles that the user selected, the results generated are different. Figure 7.11 shows some diverse results based on the same attributes described above.

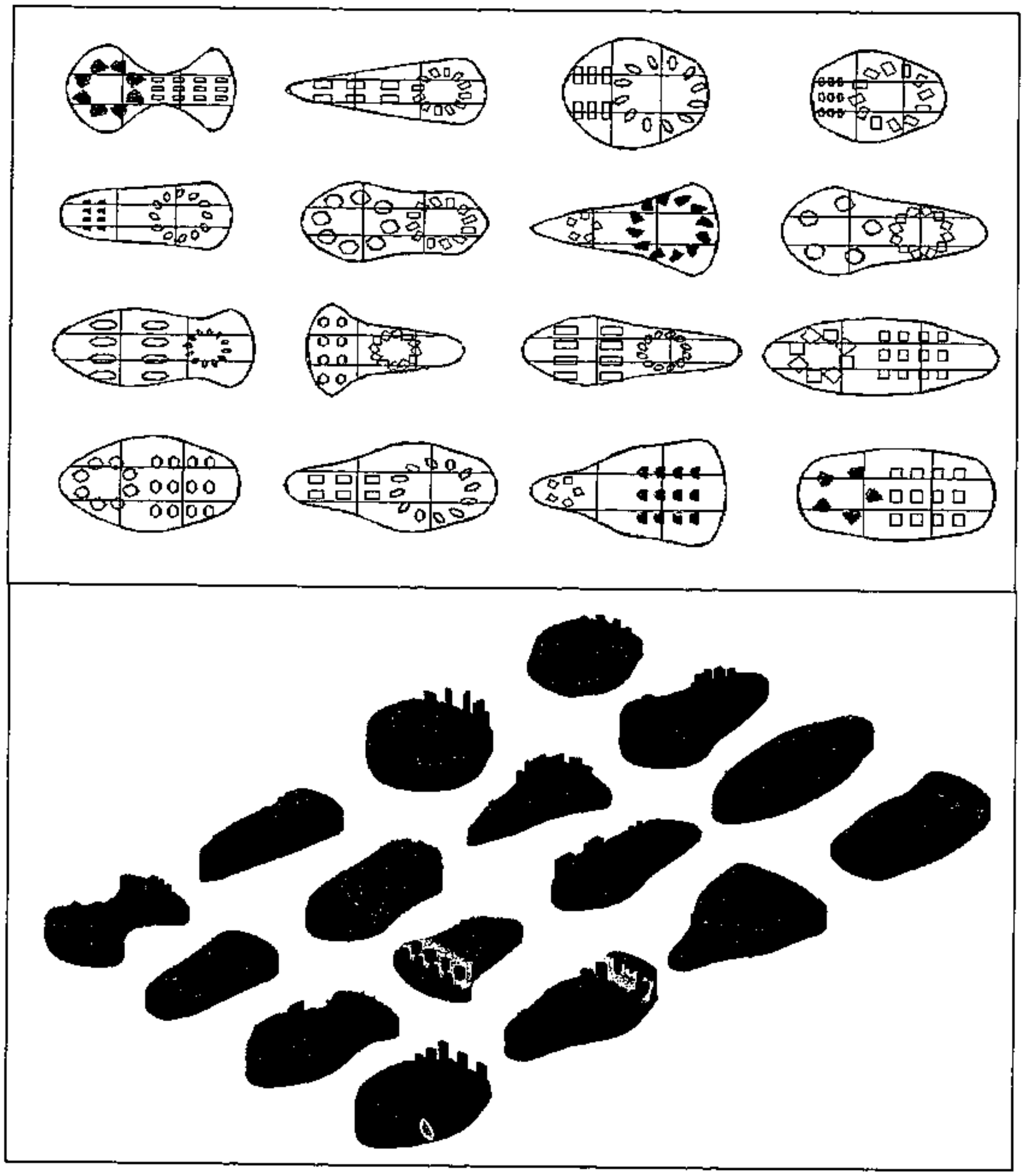


Fig 7.11. Design results based on different design requirements.

7.4.2. Evolving PDA-like Products

For the formative construction in this example, another two functional components, the screen and the functional keypad, are involved. Then based on the illustration before, the built layout of the functional components is the permutation of Screen and

FkPad on the two position (P_0), (P_1). Array-layout style of the functional keys is either centre-symmetry or mirror symmetry as listed in Figure 7.2-f., and function keys' number is variable. Then for different main-body style that the user selected, the results generated are different as shown in Fig. 7.12 below.

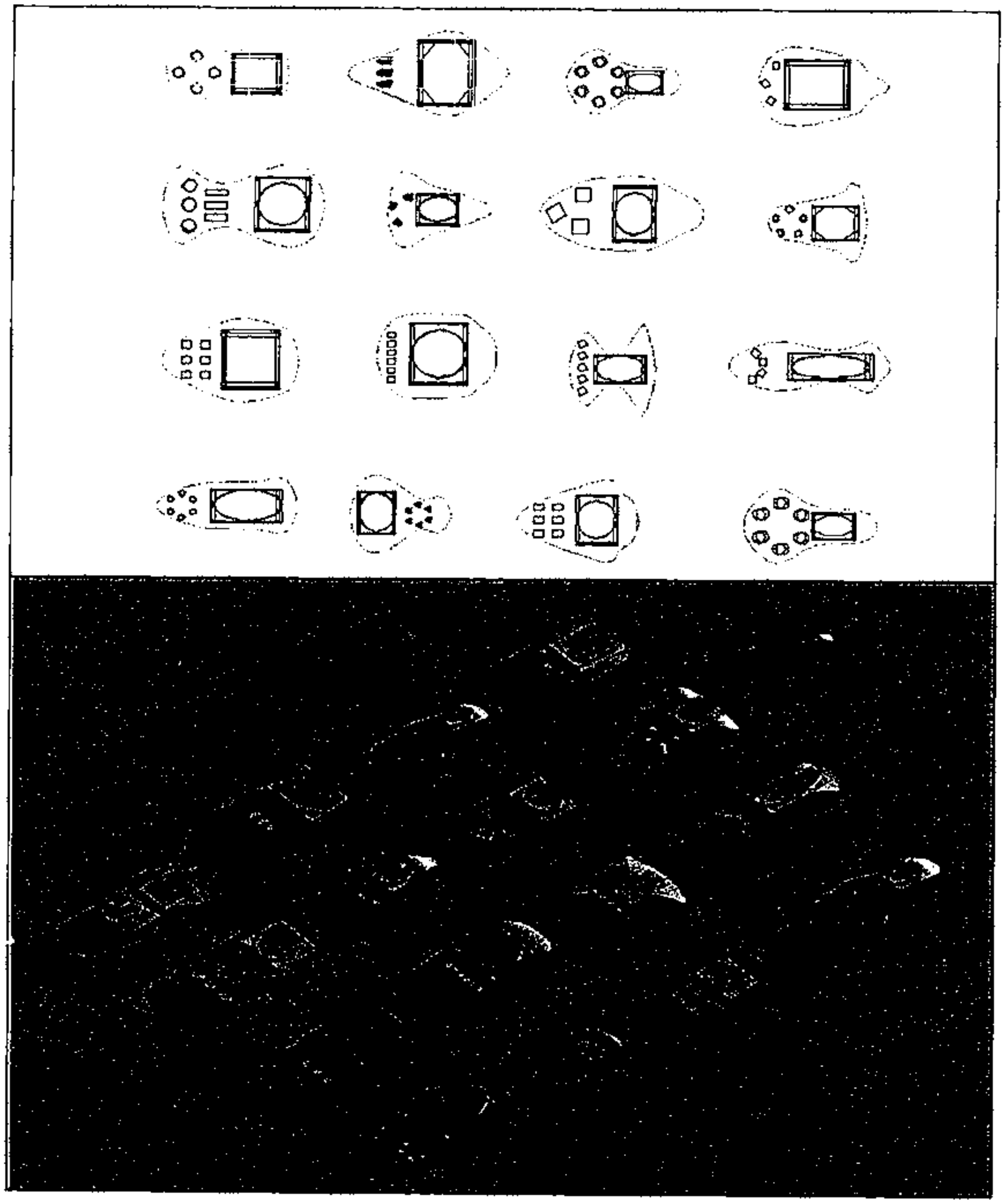


Fig 7.12. Design results based on the selected functional components.

7.5. Summary

As a general conclusion of the experiments it can be said that the system presented in Chapter 6 has proved to be capable of supporting design concept exploration at the early design stage integrated with design knowledge. The developed system has implemented a methodology for a generative and evolutionary product support design system, using GA as its core. It has demonstrated that this system can generate a population of solution alternatives at one time and promising results can be selected through the application of evaluation criteria, and through the adjustment and resetting of parameters provided by the system interface, and varied design solutions can be explored. Also with the further extension of the library of predefined product functional parts, i.e., the rudiments, more product designs can be evolved by this system.

Chapter 8

8. Conclusions and Future Work

This thesis presents research on generative and evolutionary design. In this research a generative and evolutionary design process is proposed as a way for modelling product design problems. In an analogy to natural evolution and selection, Genetic Algorithms are adopted in this research as a key mechanism to support the exploration of initial product design concepts. Results of a series of experiments are presented for demonstration of the basic concepts and the computing software developed.

In this last chapter, conclusions about this research as well as the issues to be addressed in the future research in this area are presented. The conclusions are drawn based on an assessment of the feasibility and limitations of the software system developed in this study, as described in Chapter 7. Directions for future work regarding further experiments with the system, improvement on system's GUI, and extending the system to other types of product designs are discussed.

8.1. Contributions

The research presented in this thesis has two major objectives:

- To introduce generative and evolutionary techniques into product design and model design as a generative and evolutionary process, and
- To integrate Genetic Algorithms into this process to support conceptual design with DFM considerations.

In pursuing these objectives this research has made the following contributions:

1. It developed a model of a generative and evolutionary product design process, which is different from traditional CAD process and parametric CAD process. The most significant point in this process is that at the very beginning of the process, an initial generic design concept is provided, from which a large number of desired alternatives can be derived and explored by designers.

2. Based on the model of the generative and evolutionary design process, a generative and evolutionary CAD process is further defined as consisting mainly of two cyclically linked processes. That is, design schema generating process and the design developmental process. The concepts of "Rudiment" and "Formative" are introduced for the system implementation. Rudiments and Formatives are context dependent and knowledge rich elements or building blocks of design that can be selected and evolved into complex designs.

3. DFM considerations are integrated into this process as design knowledge at the early stage in two different ways. That is, they can be specified as constraints in the design requirement as input to the system or as the evaluation and selection criteria during the generative process that can be interactively selected by designers.

4. In the system developed in this study an appropriate representation of 3D product models that can be successfully manipulated by Genetic Algorithms is developed. Rudiment and Formative form the foundation of this representation. As specified in the product design domain, a rudiment defines a set of the functional component classes with related design knowledge about their geometric and feature attributes,

then a formative encapsulates rudiments with relationships, as well as the product configuration rules to be used during the design generating process.

5. A software system with an interactive user interface is developed to validate the model and the representation. This system allows users or designers to supply requirement information, evaluation criteria or selection preference during conceptual exploration stage. The system interface supports designers to express their preference and to exercise their design exploration ability by making various selections when considering different design issues and essential factors. And the software system developed provides a flexible interface, which is easy for any further extension and improvement by the users.

6. The system developed has been tested with several design examples of the consumer products. The system is able to generate a variety of product forms based on rudiments and formatives selected by a user with little intervention. The system is capable of evolving a wide range of different types of product designs through initial easy selection and specification of key function and key parameters of a proposed product. The system is generic, in the sense that it can be extended to other domains.

7. The system is capable of evolving designs guided both by the evaluation rules by software system automatically during the evolution process, or by artificial selection by users, which involves the interaction of designers during the automatic process. The artificial selection, corresponding to natural selection in evolutionary design, can be a useful means of dealing with ill-defined selection

criteria, particularly user centred concerns. It allows designers to have the opportunity to use their experience and intuition.

8. The system allows the simple specification of a range of different product design tasks. New designs are easy for a user to specify, with the minimum of additional evaluation criteria required. In certain selected domains, most design problems can be specified by the user by simply selecting a combination of existing software modules, and the adjustment of the system parameters is mostly unnecessary. This methodology adopted provides the potential for further extension of this system in the future. The next step of this research is to produce a generic design system capable of evolving a wide range of design solutions as well as product design.

In addition the research presented in this thesis also produced the following technical results:

- The computational model implemented models the design process as a generative and evolutionary process. It is not to reflect what is going on inside a designer head, or purely modelling the design process in their mind, but trying to provide a supportive tool for the designer at a certain degree at the early design stage. And the results generated are not a single design solution but a population of design alternatives. The proposed designs are not obtained through some deliberated attempt at producing them, but by generating a wide variety of alternatives and focusing on the most promising ones, which are expected by or innovative to the designer.
- By tackling the issue of modelling 3D forms of design at the early stage of the design process using evolutionary techniques, the study addressed an important

issue that links design and manufacturing process. This involved mapping from a 3D-computer representation of design objects to a binary string manipulated by a GA program, building taxonomy of the primitive forms, considering manufacturing and other constraints in the evaluation process, and visualising the process within a computer supported environment.

- The concept of the design for manufacturability is introduced at the initial stages of conceptual design. The goal is to produce products configured such that, when detail design occurs, a manufacturable part will be realised. In the system developed in this study, the generative design process follows general manufacturability guidelines and rules for early anticipation of manufacturing problems of a proposed certain manufacturing process to avoid problems downstream in the detailed design process. In this system implementation, the consumer product design is selected as the example domain, and some guidelines related to the injection moulding manufacturing method are used. These DFM considerations are used as the configuration rules during formative construction and are transformed as the evaluation criteria during the design valuation and selection.

Also the developed prototype system has been tested by 3 researchers in Design Technology Research Centre, among them one has the background of industrial design, and other two are in computer science. Generally they agree that through this system, after initialisation input by the user with the interaction of the system interface, the system can generate various design alternatives automatically. And the design results generated by the system can be used for further refinement by other CAD tools or the concept exploration for designer. At a certain degree, the tool developed can be considered as an exploration system of design concept that helps the designer in the preliminary design stage. The benefits of an automatic system producing alternative generations of design in real time would be of significant value. While the examples presented currently are based on the generation of simplified product profiles and the result developed is still very abstract. Much still needs to be done before this system can become a useful supporting tool for conceptual design. More related knowledge needs to be embedded and the system interface needs further improvement with design intent capturing capability.

8.2. Future Work

In addition to the results produced, the research presented in this thesis opened up a new research direction and raised a series of theoretical and practical questions that need to be studied further. This gives us several opportunities to extend this research on evolutionary design, in general, and on design representation involving DFM, in particular. In the following sub-sections the plan for future work is suggested.

8.2.1. Further Experiments with the System

More experiments with the developed system are needed before the system can be used as a real world design tool. Due to the limited scope and time allowed in this study, the following issues need to be further addressed:

- Testing the system for various examples especially with real life problems from different areas of design.

- Testing the system with several databases, corresponding to different application domains, loaded simultaneously, in order to validate the generic nature of the generative and evolutionary process.
- Testing the scalability of the system.

These experiments can be done with the current implementation of the system, but modifications are needed in the software in order to provide domain specific interfaces.

8.2.2. Improvement on GUI

To make the system easier to use as well as make it available to other users such as researchers or designers, a more sophisticated graphical user interface is needed. And it is ideal to be used as the part of a collaborative system in which other supporting functions are provided for more detailed exploration of the concepts proposed by the system developed in this study. Program language with Java will benefit the further development of this system. And now Java has been a superior development tool for enterprise applications comparing to C++ and Visual Basic used in the implementation of the system. Since it was designed for networking, Java easily incorporates many features that have been complicated additions to other languages, while maintaining platform independence, its inherent security features, and reduced code size. This will be further developed in future work to make the system more functional with good performance, and web integration.

Also more advanced software packages will be needed to enable the user to interact with the developed system more efficiently and more effectively. The users can input their own descriptions such as the style of the first main-body description.

And also they can build their favoured formatives more visually through the system interface. For example for the definition of the main-body style as shown in Figure 7.1(d). A more friendly and interactive interface for this item can be provided. The users can define their own favoured styles through the system interface directly, as shown in Figure 8.1 and then the corresponding evaluation function will be provided automatically to guide the system.

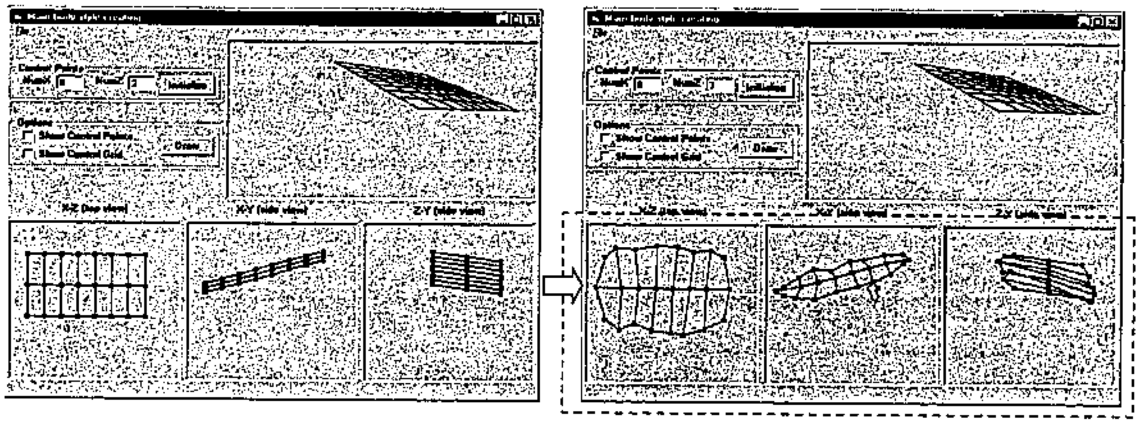


Fig 8.1. The proposed main-body-style creating interface

8.2.3. Extending the System to Other Products

Currently the system has been tested for the consumer electronic products, which are manufactured through injection moulding. But in the future research experiments on other products such as computer keyboard, mouse etc, are needed in order to allow evaluations on the suitability of the process and the system as a generic design supporting tool with DFM considerations.

8.2.4. Adding New Application Domains

Adding new application domains to the ones currently accepted by the system will extend the area of possible applications as well as increase the capability of the system to develop the idea of rudiment and formative further. These can be achieved by the application of object-oriented technology.

Object-oriented technology can be applied to the further extension of the system implementation when more new application domains are involved. Object-oriented technology is a methodology for software design in which the decomposition of a software system is based upon the concept of an object. Object-oriented methodology in programming makes the implementation much simpler than conventional programming tools. Visual Basic, C++ and Java are programming languages, which support the object-oriented programming. The benefits of object-oriented programming include easier program design, as the objects correspond closely to the behaviour of the items being simulated or calculated; fewer program errors, as objects promote modularity and encapsulation; and easier program extension, as new kinds of objects can be added more easily.

At the end, as stated earlier it is expected that the system developed will be generic in the end and it will be integrated into an AI-based generative design system developed at the Design Technology Research Centre at the Hong Kong Polytechnic University. The architecture of this AI-based generative design system is illustrated in Figure 8.2. In this architecture, the generative and evolutionary computation techniques are used to support three activities vital in the design process: concept generation, concept clustering and selection, and concept specialisation (Tang, 1999.

Sun, 1999). The methods and algorithms developed in this project will mainly support concept specialisation.

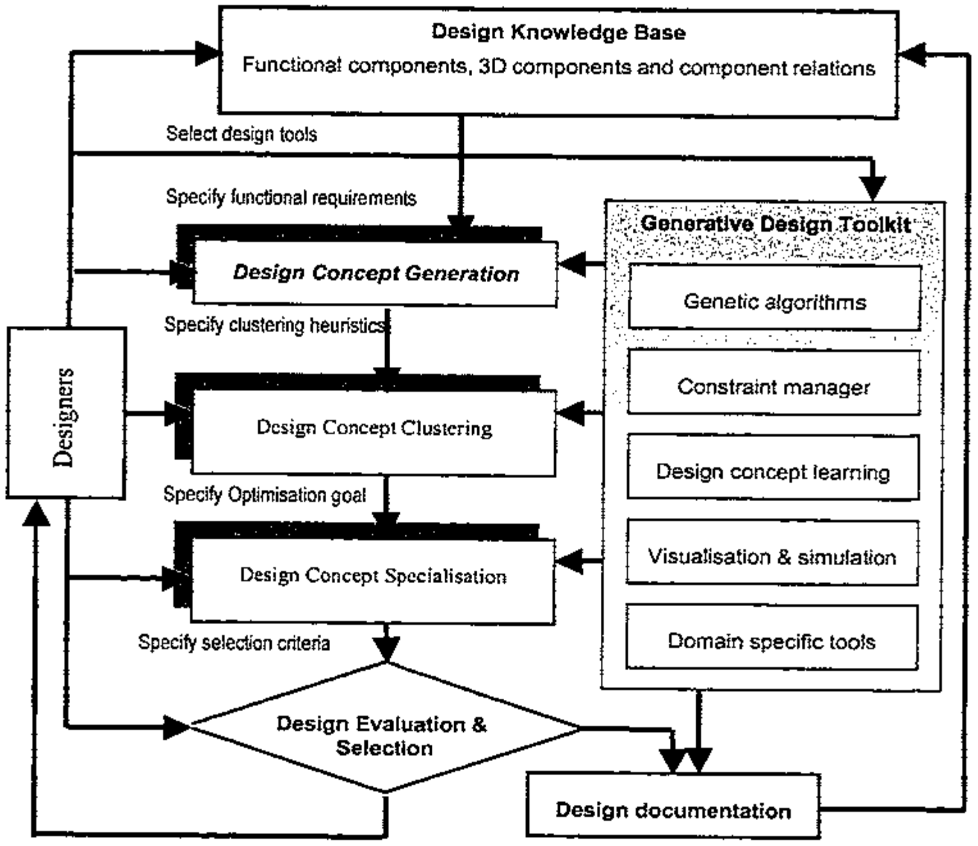

Design Knowledge Base
Functional components, 3D components and component relations
Select design tools
Specify functional requirements
Design Concept Generation
Specify clustering heuristics
Design Concept Clustering
Specify Optimisation goal
Design Concept Specialisation
Specify selection criteria
Design Evaluation & Selection
Design documentation
Designers
Generative Design Toolkit
Genetic algorithms
Constraint manager
Design concept learning
Visualisation & simulation
Domain specific tools

Fig 8.2. Architecture of an AI-based generative design system

8.3. Conclusions

As an overall conclusion, the research presented in this thesis is an attempt to create a computer model of a generative and evolutionary product design process, which is analogous to the evolutionary process in nature. Design for Manufacturability considerations along with other designer-centred issues are integrated into this process as design knowledge to support conceptual design. A theory of rudiment and formative has been established in order to identify a way in which knowledge and information about 3D products can be represented and specified by designers. A software system has been implemented to integrate Genetic Algorithms and other support functions in an interactive and systematic manner. Testing examples with mobile phone and remote controller design have shown that the system is capable of generating a wide range of variety of solutions based on initial design requirements which can be intuitively selected by designers. This research has provided a basis and insights for the development of a more generic evolutionary design system undertaken by the Design Technology Research Centre of School of Design of the Hong Kong Polytechnic University.

Reference

Agarwal, M., and Cagan, J. "A Blend of Different Tastes: The Language of Coffee Makers", *Environment and Planning B: Planning and Design*, vol.25, no.2, pp. 205-226. (1998)

Alexander, C. *Notes On The Synthesis of Form*, Harvard University Press, Cambridge, Massachusetts. (1964)

Back, T. *Evolutionary Algorithms in Theory and Practice*, England, Oxford University Press. (1996)

Barr, A., Feigenbaum, E. A., (eds.), *The Handbook of Artificial intelligence,* vol. 1, Heuristech Press, California, (1981)

Baxter, M. *Product Design: a Practical Guide to Systematic Methods of New Product Development.* London: Chapman & Hall.(1995)

Bell. Adrian D. *Plant Form: an Illustrated Guide to Flowering Plant Morphology.* Oxford University Press. (1991)

Bentley, P. *Generic Evolutionary Design of Solid Objects Using a Genetic Algorithm.* PhD thesis, University of Huddersfield. (1996)

Bentley, P. "Aspects of Evolutionary Design by Computers." *In Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 99-118, Springer-Verlag, London. (1999)

Bentley, P. (eds.) *Evolutionary Design by Computers*. Morgan. Kaufmann Publishers Inc, San Francisco. (1999a)

Bentley, P. "From Coffee Tables to Hospitals: Generic Evolutionary Design". In Bentley, P. (eds.), *Evolutionary Design by Computers*, pp 1-73. Morgan. Kaufmann Publishers Inc, San Francisco. (1999b)

Bentley, P. and Wakefield, J. "Generic Representation of Solid Geometry for Genetic Search". In *Microcomputers in Civil Engineering* vol.11(3), pp. 153-161. Blackwell Publishers. (1996)

Bieniawski, Z.T. "Designing a Design Degree in Engineering", *Design Theory and Methodology*, 68, pp.303-313. (1994)

Black, R. *Design and Manufacture: An Integrated Approach*. Macmillan Press Ltd., London. (1996)

Blessing, L.T.M. "Comparison of Design Models Proposed in Prescriptive Literature". *Proceedings of the COST A3 / COST A4 International Research Workshop on The Role of Design in the Shaping of Technology*, Lyon, 2-3 February 1995, published by the European Committee, pp 187-212, Nov.1996.(1996)

Blessing, L.T.M. "Design Process Capture and Support". *Proceedings of the 2nd Workshop on Product Structuring*, Delft University of Technology, M.Tichem, K.McCallum(eds.), pp 109-121, June 1996.

Bonner, J. T., (eds.), *On Growth and Form,* Abridged edition, Cambridge University Press. (1961)

Bown, John. *Injection Moulding of Plastic Components : A Guide to Efficiency, Fault Diagnosis, and Cure*. London ; New York. (1979.)

Boothroyd, G., and Dewhurst, P. "Design For Assembly- A Designer's Handbook". *Technical Report.* Department of Mechanical Engineering. University of Massachusetts at Amherst. (1983).

Boothroyd G, Dewhurst P, Knight W. *Product Design for Manufacture and Assembly.* Marcel Dekker, New York. (1994)

Brian M. K. *Bridging the Gap Between Design and Manufacturing: the Expert System as a Vehicle for Organisational Change*. PhD Thesis. (1990)

Brown, D. C. and Chandrasekaran, B., "Expert Systems for a Class of Mechanical Design Activity.", in J. S. Gero (eds.) *Knowledge Engineering in Computer-Aided Design*. North Holland. (1985)

Brown, D. C. "Artificial Intelligence Views of Conceptual Design." Keynote Speech on *the Third International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID&CD'00*. Nov., 2000. HongKong. (2000)

Cagan, J., and W.J. Mitchell, "Optimally Directed Shape Generation by Shape Annealing". *Environment and Planning B*, vol. 20, pp. 5-12. (1993)

Carlson, C., Woodbury R. F., " Structure Grammars and Their Applications to Design". *Proceedings of the First International Workshop on Formal Methods in Engineering Design, Manufacturing, and Assembly.* Colorado State University, January 15-17, pp99-131.(1990)

Chakrabarti, A. and Tang M., "Generating Conceptual Solutions on FuncSION: Evolution of a Functional Synthesiser", *Proceedings of The 4th International Conference on Artificial Intelligence in Design.* (1996)

Charles H. Flurscheim. (eds.) *Industrial Design in Engineering, a Marriage of Techniques*, The Design Council, London. (1983)

Chen, K.H. "A Study of Computer-Supported Formal Design". *Design Studies.* 19, 1998. pp. 331-359. (1998)

Chomsky, N. *Syntactic Structures.* The Hague: Mouton. (1957)

Coello, C. A. " An Updated Survey of GA-Based Multiobjective Optimization Techniques ". *ACM Computing Surveys.* (1997)

Coyne, R.D., Rosenman, M. A., Radford, A.D., Balachandran, M., Gero J. S. *Knowledge-based Design Systems*. Addision-Wesley Publishing Company. (1990).

Dasgupta, D. and Michalewica, Z. (eds.) *Evolutionary Algorithms in Engineering Application*, Berlin, Germany: Springer. (1997)

Davis, L. *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold. (1991)

Dawkins, R. "Universal Darwinism". In D. S. Bendall (eds.), *Evolution from Molecules to Men*. Cambridge University Press, Cambridge, Massachusetts. (1983)

Dawkins, R. *The Blind Watchmaker.* Longman Scientific & Technical Publication (1986)

Dawkins, R. *Climbing Mount Improbable*. Penguin Group. (1996)

De Jong, Kenneth A. *An Analysis of the Behaviour of a Class of Genetic Adaptive systems*. PhD Thesis, Department of Computer and Communication Science, University of Michigan, Ann Arbor. (1975)

Dym, C. L., and Levitt, R. E., "Toward the Integration of Knowledge for Engineering Modelling and Computation," *Engineering with Computers*, vol. 7, no. 4, pp. 209-224. (1991)

Duffy, M. R. "Automating the Design of Extrusions: A Case Study in Geometric and Topological Reasoning in Mechanical Engineering Design". *Proceedings, 1988 ASME Computers in Engineering Conference*, vol.1, August 1988. (1988)

Edelman, G. *Brilliant Air, Bright Fire*. Penguin Group. (1992)

Eden, M. "A Two-Dimensional Growth Process". In *Proceedings of Fourth Berkeley Symposium on Mathematics, Statistics, and Probability*, vol. 4, pp.223-239. University of California Press, Berkeley, 1960. (1960)

Erens, F. J. and Stekelenborg, R. H. A. Van "DFP: Design for Purchasing." *CAPE congrespapers*, Alphen ann de Rijin: Samsom. (1993)

Fitzhorn, P. A. " Formal Graph Languages of Shape". *Artificial Intelligence for Engineering, Design, Analysis and manufacturing*, vol. 4, no 3. (1990)

Fitzhorn P A, et al. "A Shape Grammar for Non-Manifold Modelling", *Research in Engineering Design*. vol. 2(3). 159-170. (1991)

Flemming, U. "More than the Sun of Parts: the Grammar of Queen Anne Houses". *Environment and Planning B*, vol. 14, pp. 323-350. (1987).

Foley, J., van Dam, A., Feiner, S., Hughes, J. *Computer Graphics Principle and Practice* (second edition). Addison-Wesley. (1990).

Fonseca, C.M. and Fleming, P.J. "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms II: Application Example." *Research Report 565*, Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield, U.K., Jan. 1995. (1995)

Frazer, J.H. and J. Connor. "A Conceptual Seeding Technique for Architectural Design". In PArC79, *Proceedings of International Conference on the Application of Computers in Architectural Design*, pp. 425-34, Berlin, Online Conferences with AMK, 1979. (1979)

Frazer, J. H. "Plastic Modelling." *CADCAM'87 Conference Proceedings EMAP International Exhibitions*, pp. 237-243. (1987)

Frazer, J. H. "Plastic Modelling- The Flexible Modelling of the Logic of Structure and Space." In T. Maver, & H. Wagter (eds.), *CAAD Futures'87-Proceedings of the Second International Conference on Computer-aided Design Futures.* Eindhoven, Elsevier, pp.199-208. (1987)

Frazer, J. H. "Can Computers be Just a Tool?" In *Systemica: Mutual Uses of Cybernetics and Science*, vol. 8, pp. 27-36. Amsterdam 1991. (1991)

Frazer, J. H. *An Evolutionary Architecture*. Architectural Association Publications, London, 1995. (1995)

Frazer, J., Tang, M. X. and Sun, J. "Towards a Generative System for Intelligent Design support". *Proceedings of the Fourth Conference on Computer Aided Architecture Design Research in Asia*. May,1999. (1999)

Frazer, J. H. "In Potentia, The Concept of the Rudiment", *Internal Research Paper* of the Design Technology Research Centre, School of Design. The Hong Kong Polytechnic University. August, 2000. (2000)

Frazer, J. H. "Creative Design and the Generative Evolutionary Paradigm". In P. Bentley (eds.) *Creativity and Design*. Morgan Kaufman press. (2001)

Gero, J. "Design Prototypes: A Knowledge Representation Schema for Design". In *AI Magazine*, vol. 11(4), pp. 26-36, 1990. (1990)

Gero, J. and Kazakov, V. "Adapting Evolutionary Computing for Exploration in Creative Designing". In J. S. Gero and M. L. Maher (eds.*), Computational models of Creative Design IV*, pp. 175-186. Key Centre of Design Computing and Cognition, University of Sydney, Sydney Australia, 1999. (1999)

Gen, Mitsuo and Cheng, RunWei. *Genetic Algorithms & Engineering Design*. Wiley Interscience. (1997)

Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Corporation, Inc. (1989)

Goldberg, D.E. and Deb, K. "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms." *Proceedings of the Foundations of Genetic Algorithms Workshop,* Indiana, July, 1990. (1990)

Goldberg, D. E. and Deb, K. "A comparison of selection schemes used in genetic algorithms", *Foundations of Genetic Algorithms*, G. J. E. Rawlins (eds.), pp.69-93. 1991. (1991)

Graham P. *The Application of Evolutionary and Rule-based Techniques in Computer Aided Design*. PhD Thesis. University of Ulster. (1996)

Gupta, S. K. *Automated Manufacturability Analysis of Machined Parts*. Ph.D. Thesis. Mechanical Engineering, University of Maryland, College Park, MD, September 1994. (1994)

Hanada, H. Liefer, L.J. "Intelligent Design System for Injection Moulded Parts Based on the Process Function Analysis Method", *NSF Engineering Design Research Conference*, College of Engineering, University of Massachusetts, Amherst Massachusetts, June 11-14. (1989)

Hawkes, B. and Abinett, R. *The Engineering Design Process*. The University Press (Belfast) Ltd. (1984)

Helander, M. and Nagamachi, M. *Design For Manufacturability*, Taylor & Francis, London. Washington, D C. (1992)

Holland, J. H. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbour. (1975)

Holland, J. H. *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA. (1992).

Honavar, V. "The Design Process: A Computational Perspective", *NSF Workshop on Decision-Based Design*, September 13-14, Sacramento, CA. (1997)

Houten, F. van (eds.): *Integration of Process Knowledge into Design Support Systems*, Kluwer Academic Publishers, Dordrecht, Boston, and London, (1999).

Jones Christopher. *Design Methods.* A Wiley- Interscience Publication. (1980)

Kaandorp, J. *Fractal Modelling: Growth and Form in Biology.* Springer-Verlag, Berlin, 1994. (1994)

Kanal, L.V. Cumar. (eds.) *Search in Artificial Intelligence.* Springer-Verlag. (1988)

Kauffman, S. *The Origins of Order: Self Organisation and Selection In Evolution.* Oxford University Press, New York. (1993).

Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press. (1992)

Knight, T., "The Generation of Hepplewhite-Style Chair-back Designs", *Environment and Planning B*, vol. 7, pp. 227-238. (1980)

Koning, H., Eizengberg, J., "The Language of the Prairie: Frank Lloyd Wright's Prairie House", *Environment and Planning B,* vol. 8, pp. 2953-323. (1981)

Lindenmayer, A. "Mathematical Models for Cellular Interaction in Development, Parts I and II". In *Journal of Theoretical Biology*, vol. 18, pp. 280-315, 1968. (1968)

Maher, M.L. and Balachandran, B. "Flexible Retrieval Strategies for Case-Based Design", *Artificial Intelligence in Design'94*, J. Gero (eds.), Kluwer Academic Press, pp. 163-180. (1994).

Medland, A.J. *The Computer-Based Design Process*, London. (1986)

Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs.* Sringer-Verlag. (1992).

Michalewicz, Z. "A Perspective on Evolutionary Computation", Department of Computer Science, University of North Carolina Charlotte, *Technical Report. NC28223,* USA. (1993)

Mitchell, W. J. *The Logic of Architecture- Design, Computation, and Cognition.* MIT Press. (1990)

Monedero, J., *Parametric Design: A Review and Some Experiences, in Automation in Construction*, vol. 9, no. 4, pp. 369-377. (2000)

Newell, J.C Shaw and Simon, H. A. "The Process of Creative Thinking". In H. Gruber, G. Terrell, and M. Wertheimer, (eds.) *Contemporary Approaches to Creative Thinking,* pp. 63-119. Atherton Press, New York. (1967)

Oliver, I.M., Smith, D.J. and Holland, J.R.C. "A Study of Permutation Crossover Operators on the Traveling Salesperson Problem". *Proceedings of the Second International Conference on Genetic Algorithms and their Applications.* pp. 224-230. Grefenstette, J.J.(eds.), Lawrence Erlbaum Associates, Hillsdale,. (1987.)

Papalambros, P., and Chirehdast, M. "An integrated Environment for Structural Configuration Design." *Journal of Engineering Design*, vol.1, no.1, pp. 73-95. (1990)

Parmee, I. C. (eds.) *Adaptive Computer in Design and Manufacture.* Engineering Design Centre, University of Plymouth, Drakes Circus, Plymouth PL4 8AA, United Kingdom. Springer. (1998)

Pevsner, N. *A History of Building Types*, Princeton University Press, Princeton. (1976)

Prusinkiewicz, P. "Modelling and Visualisation of Biological Structures". In *Proceedings of Graphics Interface '93*, pp. 128-137, 1993. (1993)

Prusinkiewicz, P. "Visual Models of Morphogenesis". In *Artificial Life*, 1(1/2), pp.67-74, 1994. (1994)

Prusinkiewicz, P. and Lindenmayer, A. *The Algorithmic Beauty of Plants.* Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer. (1994)

Pugh, S. *Total Design: Integrated Methods for Successful Product Engineering*, Addison-Wesley Publishing Company, Wokingham, England. (1991)

Reddy, G., and J. Cagan, "Optimally Directed Truss Topology Generation Using Shape Annealing". *ASME Journal of Mechanical Design*, vol. 117, no. 1, pp. 206-209. (1995a)

Reddy, G., and Cagan, J. "An Improved Shape Annealing Algorithm For Truss Topology Generation," *ASME Journal of Mechanical Design*, vol. 117, no. 2(A). pp. 315-321.(1995b)

Reich, Y. (eds.) Special Issue on Research Methodology. In *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing (AI EDAM),* vol.8 (4), 1994. (1994a)

Reich, Y. "Layered Models of Research Methodologies". In *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing (AI EDAM*), vol.8 (4), pp. 263-274, 1994. (1994b)

Rooney, J. and Steadman P. (eds.) *Principles of Computer-aided Design*. The Open University. UCL press limited. (1997)

Rosen, D. W. *A Feature-based Representation to Support the Design Process and the Manufacturability Evaluation of Mechanical Components*. Ann Arbor, Mich. : U.M.I., 1993, c1992. PhD Thesis.--University of Massachusetts, 1992. (1993)

Roy, R., Furuhashi, T. and Chawdhry, P. K. (eds.) *Advances in Soft Computing: Engineering Design and Manufacturing*. London. Springer.

Salomons, O. W. *Computer Support in the Design of Mechanical Products, Constraint Specification and Satisfaction in Feature Based Design for Manufacturing*. PhD thesis of University Twente. (1995)

Schaffer, J.D., Caruna, R.A., Eshelman, L.J., Das, R., in *Proceedings of the 3rd International Conference on Genetic Algorithms and Applications*; Schaffer, J.D. (Eds.); Morgan Kaufman: San Mateo, CA, 1989, pp 51-60. (1989)

Schmidt, L. C. and Cagan, J. "Grammars for Machine Design," J. S. Gero and F. Sudweeks (eds.), *Artificial Intelligence in Design '96*, Kluwer Academic Publishers, The Netherlands, pp. 325-344. (1996)

Shah, J.J. "Feature Transformations Between Application-Specific Feature Spaces". *CAE Journal*. vol. 5(1), pp.9-15. (1988)

Shah, J.J. "Assessment of Features Technology." *Computer-Aided Design*. vol. 23(5): pp.331-343. (1991)

Sims, K., "Evolving Virtual Creatures", *Computer Graphics Proceedings*, pp. 15-22. (1994)

Sims, K., "Artificial Evolution for Computer Graphics". *Computer Graphics*, vol., 25, no. 4, July 1997, pp319~328. (1997)

Sims, K., "Interactive Evolution of Dynamical System". Thinking Machine Corporation, *Technical Report Series*, pp171-178. (1997)

Smithers,T., Conkie,A., Doheny, J., Logan, B., Millington, K., Tang, M. X. "Design as Intelligent Behaviour: An AI in Design Research Program" Journal of *Artificial Intelligence in Engineering*. vol. 5, pp78-109. (1990)

Steadman, J. P., *Architecture Morphology, An Introduction to the Geometry of Building Plans*. Pion Limited, London, pp.9-20.(1983)

Stiny, G., Grips, J., *Algorithmic Aesthetics*, University of California Press Ltd. (1978)

Stiny, G., "Introduction to Shape and Shape Grammars", *Environment and Planning B*, vol. 7, pp.343-351. (1980a)

Stiny, G., Mitchell, W. J., "The Grammar of Paradise: on the Generation of Mughul Gardens", *Environment and Planning B*, vol. 7, pp.209-226. (1980b)

Stuart, Pugh. *Total Design: Integrated Methods for Successful Product Engineering.* Publisher Wokingham, England . Addison-Wesley Publisher. Co.,1991. (1991)

Sun, J., Frazer, J., Tang M. X. "Shape Representation for Genetic Algorithms in Evolutionary Design". *Proceeding of the Second International Conference on Computer-Aided Design and Conceptual Design, CAID&CD'99.* Nov., 1999. (1999a)

Sun, J., Frazer, J., Tang M. X., "Application of Evolutionary Techniques in Design for Manufacturability". *Proceeding of the Fifth International Conference on Computer-Aided Conceptual Design CACD'99*, Lancaster University. May,1999. (1999b)

Sun, J., Frazer, J., Tang, M. X., (2000), "Research on Applications of Genetic Algorithms to Computer Aided Product Design- Case Studies on Three Approaches.", *Proceedings of the Third International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID&CD'00.* Nov., 2000. HongKong. (2000a)

Sun, J., Frazer, J., Tang, M. X. "Using Evolutionary Techniques to Shape Optimisation in Product Design". *27th International Conference on Computers & Industrial Engineering.* 11-13, Oct. 2000, Beijing, China. (2000b)

Sun, J., Frazer, J., Tang, M. X. "Shape Optimisation in Design for Manufacturing Using Evolutionary Techniques". *ICME2000-2nd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering.* 21-23 June 2000, Capri (Naples), Italy. (2000c)

Syswerda, G., "Uniform Crossover in Genetic Algorithms". In Schaffer, D. (eds.), *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann Publication, (1989).

Tang, M. X. "A Knowledge-based Architecture for Intelligent Design Support". *International Journal of Knowledge Engineering Review*. vol.12 (4), pp. 387~406. (1997)

Tesar, P. "The Other Side of Types". In Rockcastle. G. (eds.) Type and the (Im)possibilities of Convention, pp.165-175. University of Minnesota College of Architecture and Landscape Architecture, Minnesota, 1991. (1991)

Thornton, A. C. *Constraint Specification and Satisfaction in Embodiment Design*. PhD thesis. University of Cambridge, Department of Engineering. (1993)

Thornton, A. C. "Genetic Algorithms Versus Simulated Annealing: Satisfaction of Large Sets of Algebraic Mechanical Design Constraints", *Artificial Intelligence in Design '94*, pp. 381~398. (1994)

Tomiyama T., Yashikawa H. "Extended General Design Theory", in: *Design Theory for CAD*, (eds.) Yoshikawa H., Warman E. A., Elsevier Science Publisher (North Holland), IFIP, 1987, pp.95-124. (1987)

Turing, A. "Intelligent Machinery". In C. R. Evans and A. D. J. Robertson, (eds.) *Cybernetics: Key Papers,* University Park Press, Baltimore Md. and Manchester (1968)

Ulam, S. "On Some Mathematical Properties Connected with Patterns of Growth of Figures". In *Proceedings of Symposia on Applied Mathematics*, vol.14, pp.215-224. American Mathematical Society, 1962. (1962)

Watabe, H. and Okino, N. "A study on Genetic Shape Design". *Proceedings of the Fifth International Conference on Genetic Algorithms.* pp.445-450. (1993)

Wagner, G. and Altenberg, L. "Complex Adaptations and the Evolution of Evolvability". In *Evolution*, vol. 50, no. 3, pp. 967-976. (1996)

Wallace David R. *A Computer Model of Aesthetic Industrial Design*. MIT thesis. (1991)

Wallace David R. and Mark J. Jakiela. "Automated Product Concept Design: Unifying Aesthetics and Engineering". *IEEE Computer Graphics & Applications*. 1993.vol.13, no. 4, pp 66-75. (1993)

Wallace, K. M.(eds) 'Pahl, G. and Beitz, W, *Engineering Design, a Systematic Approach'*. (2nd eds) Springer-Verlag, London. (Translators: Wallace KM, Blessing, LTM and Bauert, F.). (1996)

Wierda, L. S. "Linking Design, Process Planning and Cost Information by Feature-Based Modelling", *Journal of Engineering Design*. vol. 2, no. 1. 1991, pp3-19.(1991)

Winter, G., Periaux, J. (eds.) *Genetic Algorithms in Engineering and Computer Science*. Printed in Great Britain.(1996)

Wittenoom, R., "General Case Parametric Representation and Exchange - Future Directions", *ISO/WD Technical Report Type 3, ISO TC184/SC4/WG12 N292*, National Institute of Standards Technology, Gaithersburg, Maryland, USA. (1999)

Wright, I. C. *Design Methods in Engineering and Product Design*, Engineering Design Institute, Department of Mechanical Engineering, Loughborough University. (1998)

Yannoulakis, N. J. *A Manufacturability Evaluation and Improvement System*, Ph.D. Thesis. (1991)

Appendix A

This appendix illustrate the program implementation of one evaluation criteria, the main-body style involved in this system as illustrated in chapter 7:

```
'**************************************************
'The Evaluation Function                          *
'***************************************************

Public Sub Evaluate()

   'Dim tmp(3) As String
   Dim fitval As Integer, tmp As Integer
   Dim f0 As Integer, f1 As Integer, f2 As Integer, f3 As Integer
   Dim f4 As Integer, f5 As Integer, f6 As Integer
   Dim mem As Integer, i As Integer, ca As Integer
   Dim MaxLsh As Double, MaxWsh As Double, MaxHsh As Double
   Dim MinLsh As Double, MinWsh As Double, MinHsh As Double
   Dim cmfit1(PopMax - 1) As Double   'need further refine in one
   Dim cmfit2(PopMax - 1) As Double
   Dim cmfit3(PopMax - 1) As Double
   Dim cmfit4(PopMax - 1) As Double
   Dim TmpTotal_cmfit(PopMax - 1) As Double
   Dim tmp1 As Double, tmp2 As Double
   Dim l1 As Double, w1 As Double, h1 As Double, nkst As Integer, _
        l2 As Double, w2 As Double, h2 As Double, fknos As Integer, fkst As Integer, _
        l3 As Double, w3 As Double, h3 As Double, vdst As Integer
  Dim bcno As Integer, fvno As Integer, rvno As Integer, snsno As Integer
  Dim MaxH As Double, MidH As Double, MinH As Double
  Dim MaxW As Double, MidW As Double, MinW As Double

'get the variable values

For i = 0 To Popsize - 1

l1 = Allgen(i, 0)    ' Before map to Allpheno 0~100
w1 = Allgen(i, 1)    'gen/pheno
h1 = Allgen(i, 2)
l2 = Allgen(i, 4)
w2 = Allgen(i, 5)
h2 = Allgen(i, 6)
'fknos = Rcvar(7)
'fkst = Rcvar(8)
l3 = Allgen(i, 9)
w3 = Allgen(i, 10)
h3 = Allgen(i, 11)
'vdst = Rcvar(12)
bcno = Allgen(i, 13)   '0~100
fvno = Allgen(i, 14)
rvno = Allgen(i, 15)
snsno = Allgen(i, 16)
```

```
If h1 > h2 Then
  If h1 > h3 Then
                MaxH = h1
                If h2 > h3 Then
                                MidH = h2
                                MinH = h3
                Else: MidH = h3
                     MinH = h2
                End If
  Else: MaxH = h3
        MidH = h1
        MinH = h2
  End If
ElseIf h2 > h3 Then
                  MaxH = h2
             If h1 > h3 Then
                       MidH = h1
                       MinH = h3
             Else: MidH = h3
                  MinH = h1
            End If
Else: MaxH = h3
    MidH = h2
    MinH = h1
End If


If w1 > w2 Then
  If w1 > w3 Then
    MaxW = w1
      If w2 > w3 Then
         MidW = w2
         MinW = w3
      Else: MidW = w3
          MinW = w2
      End If
  Else: MaxW = w3
      MidW = w1
      MinW = w2
  End If
ElseIf w2 > w3 Then
  MaxW = w2
    If w1 > w3 Then
       MidW = w1
         MinW = w3
      Else: MidW = w3
          MinW = w1
    End If

Else: MaxW = w3
    MidW = w2
    MinW = w1
End If
```

```
'Calculate the fitness to each main-body style
Select Case BC_styl
'When proposed to be style-1
    Case 0
     cmfit2(i) = 0.5 * (200 - Diff(MaxW, w2) - Diff(w1, w3)) '100 * w2 / MaxW
'When proposed to be style-2
    Case 1        'w1<w2<w3 w1-->0
      tmp1 = 0.5 * w3 / (l3 + l2 / 2 + 1)
      tmp2 = 0.5 * w2 / (l1 + l2 / 2 + 1)
     cmfit2(i) = (100 - Diff(MaxW, w3) + 100 * TorF(tmp1, tmp2) + 100 * TorF(w2, w1)) / 3
'When proposed to be style-3
    Case 2        'w1<w2<w3
      tmp1 = 0.5 * w3 / (l3 + l2 / 2 + 1)
      tmp2 = 0.5 * w2 / (l1 + l2 / 2 + 1)
      cmfit2(i) = 0.2 * (500 - Diff(MaxW, w3) - Diff(w1, w2) - Diff(w2, w3) - Diff(w1, w3) -
Diff(tmp1, tmp2)) '100 * (w3 / MaxW + w2 / MidW + MinW / w1) / 3
'When proposed to be style-4
   Case 3         'Maxw1 mid w2 minw3
      'tmp1 = 0.5 * w2 / (l1 + l2 / 2 + 1)
      'tmp2 = 0.5 * w3 / (l3 + l2 / 2 + 1)
      cmfit2(i) = 0.5 * (100 - Diff(MaxW, w1) + 100 * TorF(w2, w3))  '100 * w2 / MaxW

End Select
```

```
' get the total_fit base on the weights
For mem = 0 To (Popsize - 1)
    TmpTotal_cmfit(mem) = 0.5 * cmfit2(mem) + 0.3 * cmfit3(mem) + 0.2 * cmfit4(mem)
    'this function can be further generalised
  Next
```

```
'for artificial selection
  If Aselt_flag = True Then
    For mem = 0 To 8          'total now is 9(Popsize - 1)
      Total_cmfit(mem) = 100 * Idx_favor(mem) + TmpTotal_cmfit(mem)
    Next

  For mem = 9 To Popsize - 1        '(Popsize - 1)
```

```
      Total_cmfit(mem) = TmpTotal_cmfit(mem)
    Next
  Else
    For mem = 0 To Popsize - 1          '(Popsize - 1)
      Total_cmfit(mem) = TmpTotal_cmfit(mem)
    Next
  End If
  Aselt_flag = False

End Sub
```

Appendix B

This appendix shows more pictures of design results as well as shown in chapter7:

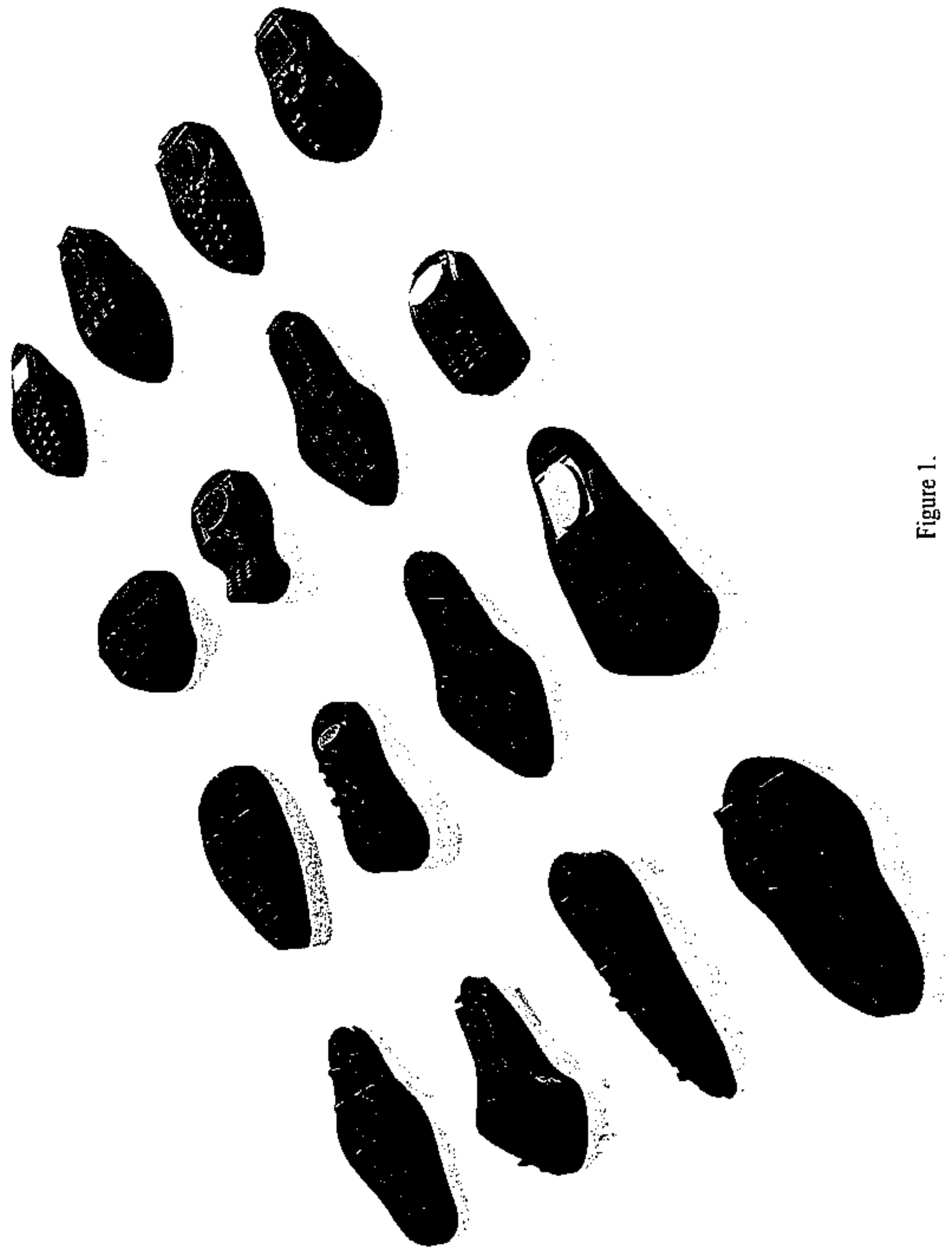


Figure 1.

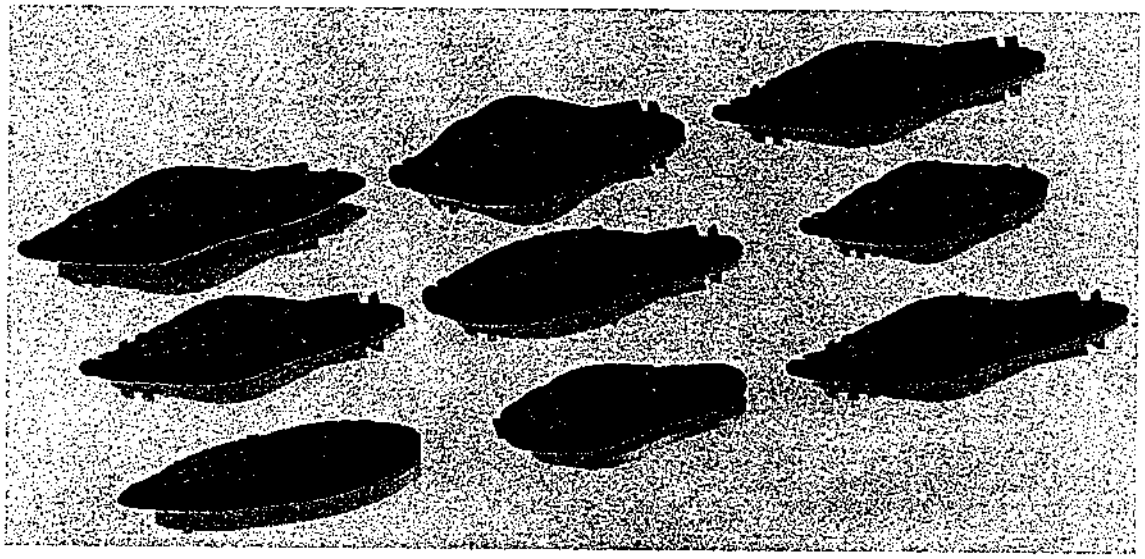


Figure 2.

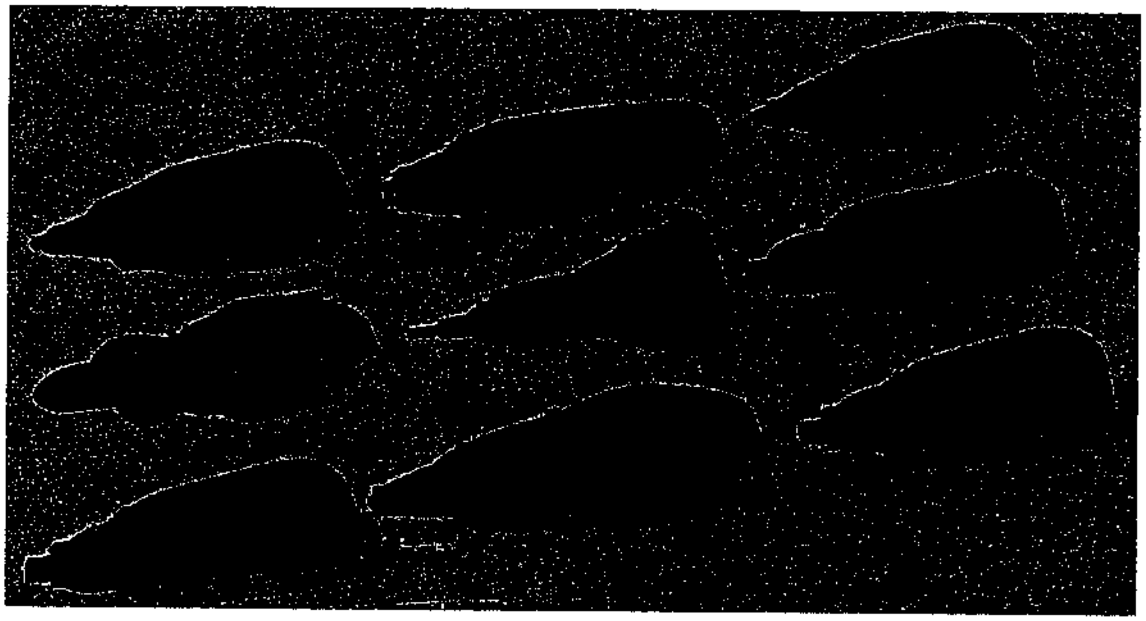


Figure 3.

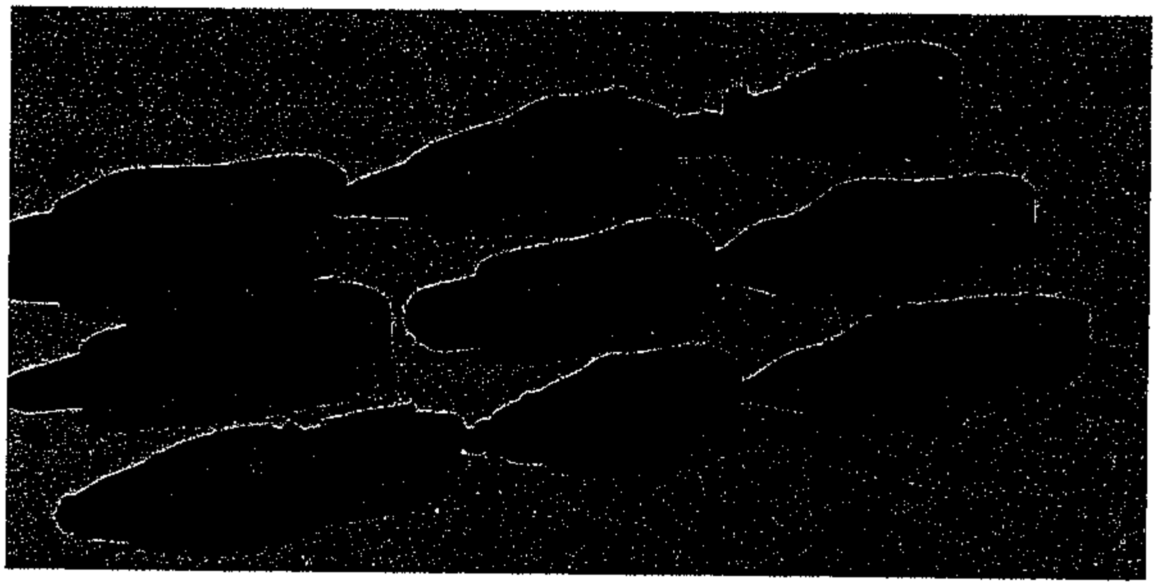


Figure 4.

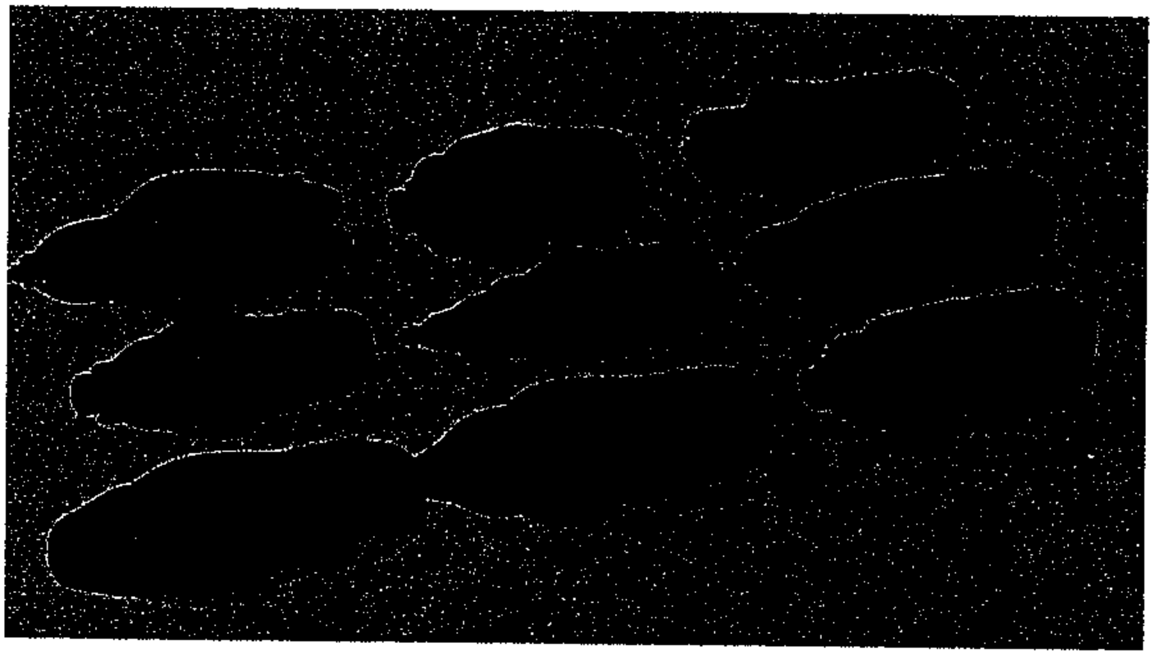


Figure 5.

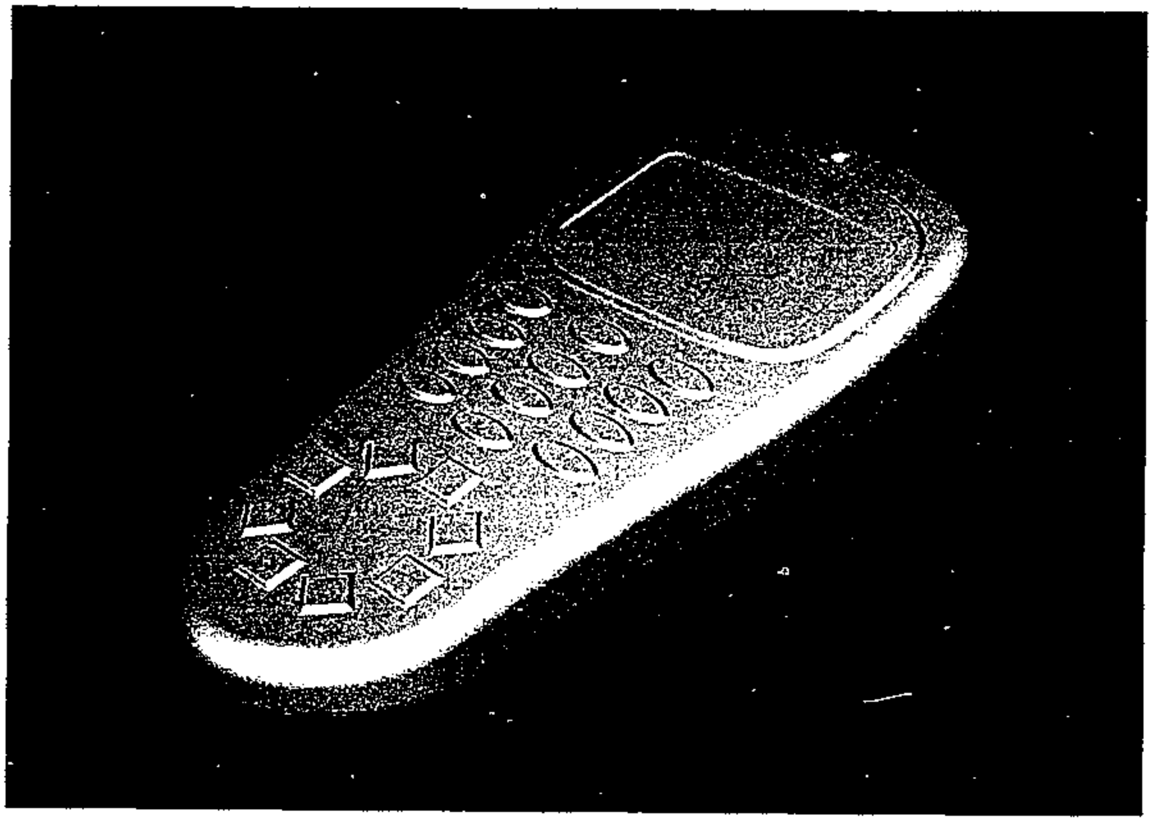


Figure 6.

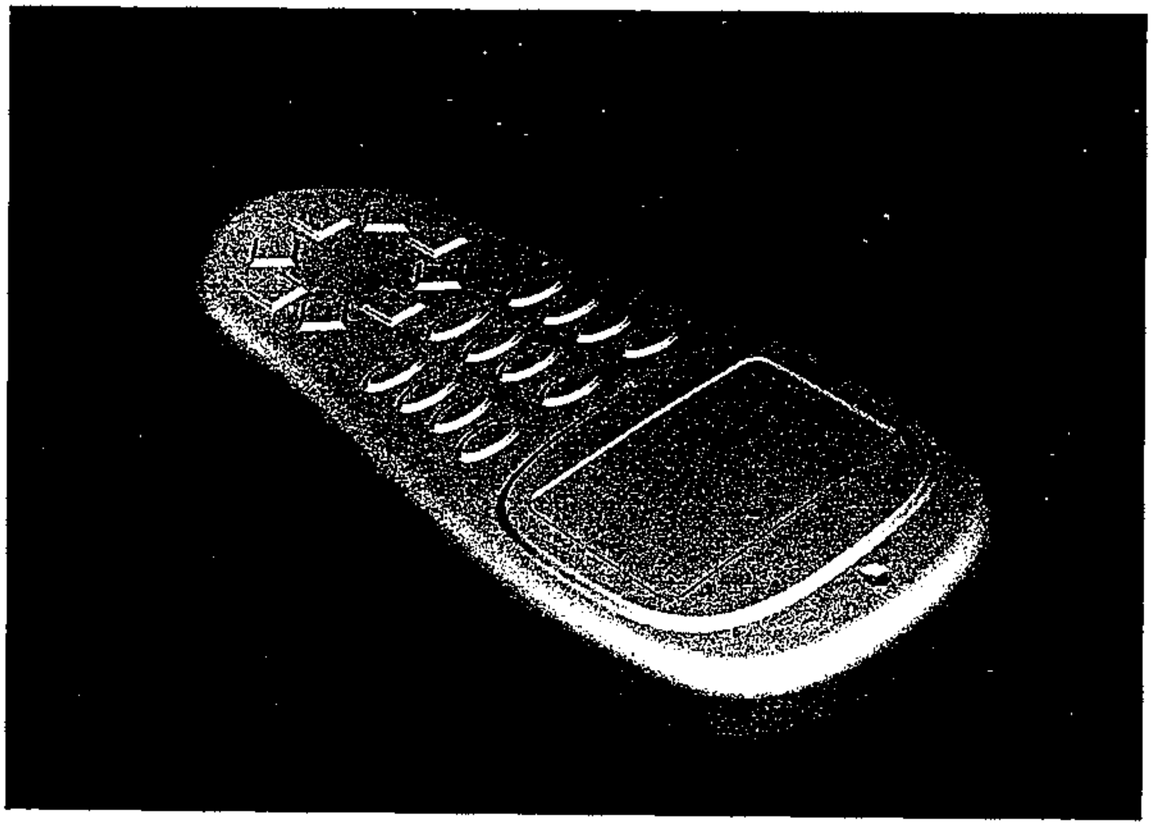


Figure 7.

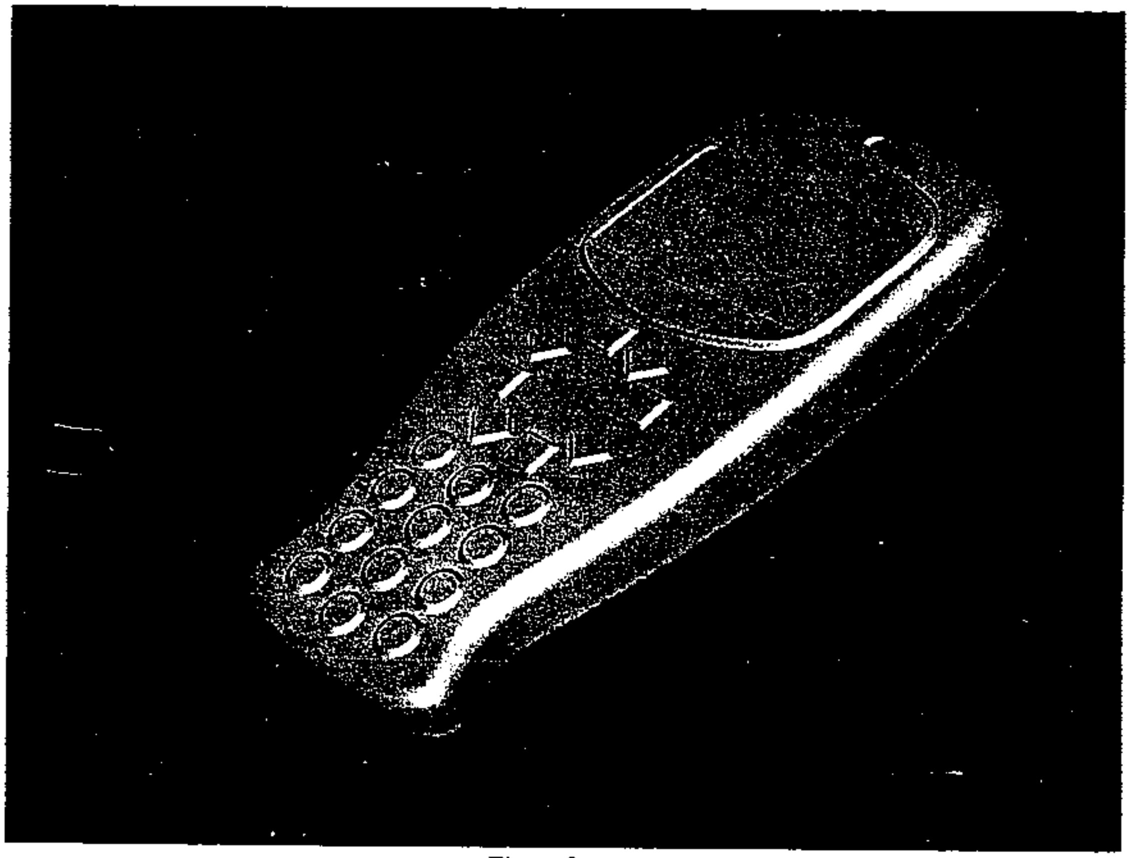


Figure 8.

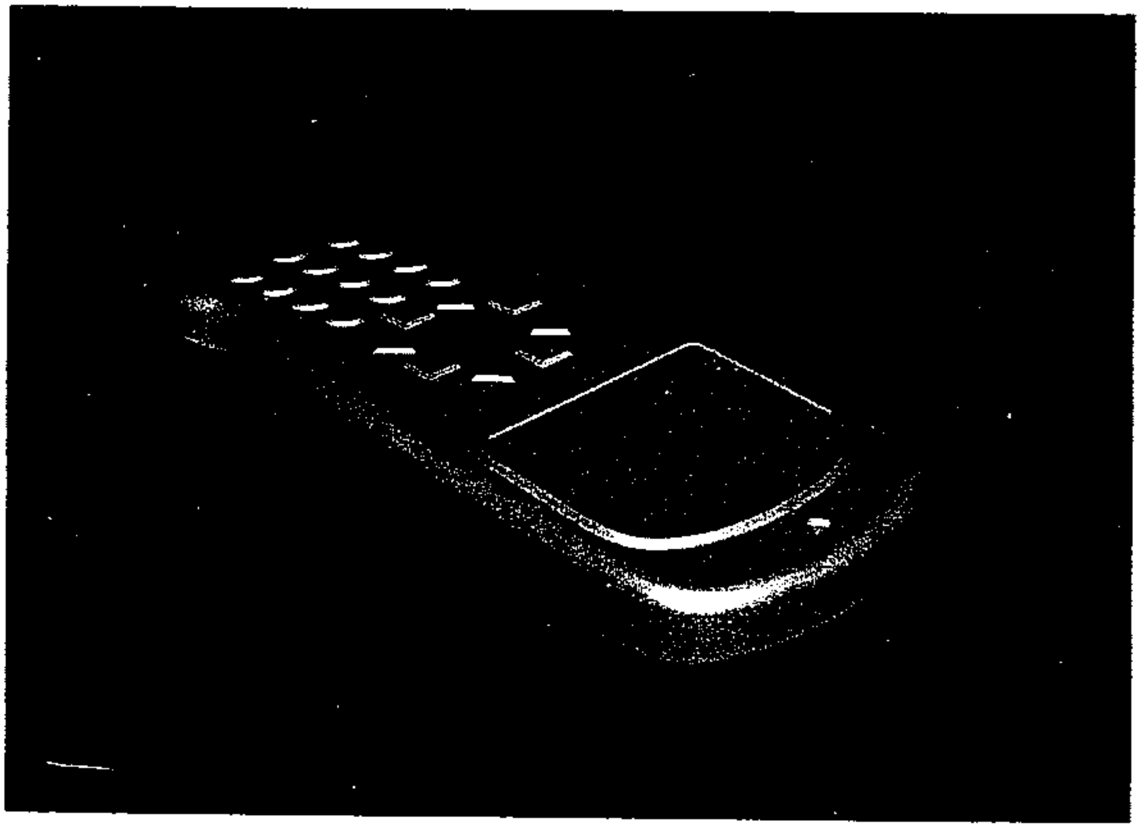


Figure 9.

Appendix C

This appendix listed out the papers published as part of research undertaken for this thesis follow:

1. Sun, J., Frazer, J., Tang, M. X., (2000), "Research on applications of Genetic Algorithms to Computer aided product Design- Case studies on three approaches.", *Proceedings of the Third International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID&CD'00.* Nov., 2000. HongKong.

2. Sun, J., Frazer, J., Tang, M. X. (2000), "Using Evolutionary Techniques to Shape Optimisation in Product Design". *27th International Conference on Computers & Industrial Engineering.* 11-13, Oct. 2000, Beijing, China.

3. Sun, J., Frazer, J., Tang, M. X. (2000), "Shape Optimisation in Design for Manufacturing Using Evolutionary Techniques". *ICME2000-2nd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering.* 21-23 June 2000, Capri (Naples), Italy.

4. Sun, J., Frazer, J., Tang M. X. (1999). "Shape Representation for Genetic Algorithms in Evolutionary Design". *Proceeding of the Second International Conference on Computer-Aided Design and Conceptual Design, CAID&CD'99.* Nov., 1999.

5. Frazer, J., Tang, M. X. and Sun, J.(1999). "Towards a Generative System for Intelligent Design support". *Proceeding of the Fourth Conference on Computer Aided Architecture Design Research in Asia.* May,1999.

6. Sun, J., Frazer, J., Tang M. X., (1999). "Application of Evolutionary Techniques in Design for Manufacturability". *Proceeding of the Fifth International Conference on Computer-Aided Conceptual Design CACD'99,* Lancaster University. May 1999.