On the integration of design and manufacturing

ON THE INTEGRATION OF DESIGN AND MANUFACTURING

F.L.M. DELBRESSINE

ON THE INTEGRATION OF DESIGN AND MANUFACTURING

F.L.M. DELBRESSINE

Dit proefschrift is opgemaakt met een Macintosh II en is belicht met een Linotronic 300.
Zetwerk: F. Delbressine in samenwerking met BMS Professional Text Design BV, te Vlodrop
Belichting: Softext Computersatz-Service GmbH, Düsseldorf
Drukwerk: Druckerei Albers, Düsseldorf

ON THE INTEGRATION OF

DESIGN

AND

MANUFACTURING

Proefschrift
ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de Rector Magnificus, prof. ir. M. Tels,
voor een commissie aangewezen door het College
van Dekanen in het openbaar te verdedigen op
dinsdag 19 september 1989 te 14.00 uur

door

FRANCISCUS LEONARDUS MARIE DELBRESSINE

Werktuigbouwkundig Ingenieur
geboren op 26 februari 1957 te Beek

Dit proefschrift is goedgekeurd door de promotoren:

Prof.dr.ir. J.E. Rooda
en
Prof.dr.ir. A.C.H. van der Wolf

Voor: Marion,
mijn ouders

Summary

This study introduces certain concepts for the integration of the design and manufacture of mechanical products. Their power springs from the enforcement of manufacturing restrictions at the geometrical design stage, which ensures a guarantee that a design generated according to the method described will be capable of being fabricated.

Only relatively few concepts are necessary in order to superimpose manufacturing restrictions on the geometrical design process. These are: Manufacturable Transformations, Manufacturable Objects, Manufacturing Machine Models, Implicit Location, and two geometrical modelling representations–Boundary Representation and Constructive Solid Modelling.

A Manufacturable Transformation is a design transformation that has a manufacturable counterpart. For example, the 'combine' transformation can have welding as its manufacturable counterpart.

A Manufacturable Object is a geometrical shape, together with its application rules. The geometrical form is a form that can in principal be fabricated. The relevant application rules guarantee that the geometrical form can be fabricated within a specific context.

A Manufacturing Machine Model is a model of an available machine. The model knows if the corresponding machine is capable of fabricating a Manufacturable Object or executing a Manufacturable Transformation and it also knows how to manufacture such a design transformation. The Manufacturing Machine Model introduces the available machines and associated equipment – in particular their limitations – into the geometrical design process and into the generation of a manufacturing process plan.

Implicit Location is a method whereby the limited accuracy that is attainable by a given manufacturing process may be taken into account. It allows a func-

tional approach to the problem of incorporating manufacturing tolerances and fits into the design phase. The recording of items of reference by the method of Implicit Location allows the incorporation of sensors in the manufacturing process, which means that the values obtained from the sensors can be compared with the required values, so that corrections can be generated if necessary.

The Boundary Representation is introduced since manufacturing processes produce surfaces which have to be incorporated into the geometrical design phase in order to take account of manufacturing restrictions.

In order to reduce human intervention, manufacturing process planning needs a design representation that is based on the initial state of an object, together with a list of the state changes through which the object passes on its way to becoming a finished product. Constructive Solid Modelling has thus been introduced into the geometrical design phase to take account of this. A consequence of this introduction is that only a number of Manufacturable Objects having simple geometrical forms need to be made available, since these can be combined into more complex geometrical forms. The application rules associated with the geometrical forms ensure that the design can in fact be fabricated.

The Manufacturable Object concept is particularly important in the enforcement of manufacturing restrictions on the geometrical design, since Manufacturable Objects represent a formalization of what can actually be fabricated.

The definition of new Manufacturable Objects, in particular their application rules, remains, unfortunately, a tedious task, since the generation of application rules formalizes the context- dependent knowledge of what can actually be fabricated. The rules are not only context dependent, however, they also depend on the equipment available.

Six fundamental fabrication techniques have been considered: primary shaping, forming processes, material removal, joining, coating, and material feature changing (such as hardening or annealing). Of these, three can be dealt with directly on the basis of the concepts introduced here: primary shaping,

material removal, and joining. Coating and material feature changing can readily be incorporated. The forming process might be tackled with the help of the Manufacturable Object concept, but it is not so well suited to the concept as the processes that have been incorporated.

The main advantages of the concepts introduced in this study are that they guarantee, at the geometrical design stage, that a design, once produced, can actually be manufactured. They also offer the rapid generation of manufacturing process plans, and they thus substantially reduce the time taken to traverse the path between design and manufacture.

Samenvatting

Het doel van deze studie is een integratie van de ontwerp- en de fabricagefase van een produkt te realiseren. Integratie is een middel om een verkorting van de benodigde doorlooptijd tussen ontwerp en fabricage van een produkt te realiseren en de kwaliteit van het produkt te verbeteren. De ontwerpfunctie en de fabricagefunctie van een fabriek, een produkt generator, hangen nauw met elkaar samen. De nauwe samenhang noopt tot het gebruik van een wijze van modelleren gebaseerd op een systeem benadering. Gekozen is voor de Proces Interactie Benadering.

De taak van een produkt generator is de gewenste produktfuncties te transformeren in een produkt. Het model van de produkt generator bestaat uit een cybernetische architectuur van functies en interacties. Karakteristiek voor een cybernetische architectuur is dat er een terugkoppeling aanwezig is. Deze terugkoppeling wordt, in het model, gerealiseerd met behulp van een evaluatorische functie en drie interacties. De fysische betekenis van deze terugkoppeling is dat fabricage beperkingen worden gesuperponeerd op de ontwerp functie: de ontwerper wordt in de ontwerpfase van een produkt al geconfronteerd met fabricage beperkingen. Fabricage beperkingen zijn bijvoorbeeld vormen die niet vervaardigd kunnen worden in een bepaald materiaal of op een specifieke plaats in het produkt.

Tot nu toe hanteert geen enkele geometrische ontwerp methode fabricage beperkingen. Het hanteren van deze beperkingen wordt overgelaten aan de ontwerper. De gewenste doorlooptijd verkorting, de gewenste kwaliteit verbetering, de grote hoeveelheid benodigde kennis van fabricage processen, het toenemende aantal functies en het toenemende aantal interacties tussen deze functies van een produkt, maken het de ontwerper bijna onmogelijk fabricage beperkingen te hanteren.

In deze studie is een geometrische ontwerpmethode ontwikkeld, die het mogelijk maakt al in de ontwerpfase fabricage beperkingen te hanteren. In tegen-

stelling tot de meeste ontwerpmethoden is de ontwerpmethode niet gebaseerd op een eindtoestand beschrijving maar op een beschrijving van de begintoestand gecombineerd met de respectievelijke operaties nodig om de eindtoestand te realiseren.

Een voorbeeld van een operatie is het aftrekken van een cylindrisch blok van een rechthoekig blok. Uiteraard dienen deze ontwerp operaties fabriceerbaar te zijn. Dit wordt gerealiseerd door elke ontwerp operatie een bewerking in de fabricage fase als tegenhanger te geven en door een nieuw begrip in te voeren: het fabriceerbare object.

Een voorbeeld van een fabriceerbare ontwerp operatie is de 'voeg samen' operatie. Deze kan als fabricage tegenhanger hebben de bewerking lassen. Een fabriceerbaar object is de ontwerp- en werkvoorbereidings tegenhanger van de toepassing van een of meer gereedschappen, machines en opspanningen. Voorbeelden van fabriceerbare objecten zijn sleuven, kamers, maar ook een buigbewerking kan een fabriceerbaar object zijn.

Een fabriceerbaar object bestaat uit twee subconcepten: de geometrische vorm en de toepassings regels behorende bij deze geometrische vorm. De toepassings regels zorgen ervoor dat een ontwerp fabriceerbaar blijft na de applicatie van de geometrische vorm van een fabriceerbaar object; ze verhinderen een toepassing die niet kan worden gefabriceerd.

De realiseerbare nauwkeurigheid van fabriceerbare objecten, bijvoorbeeld de oppervlakte ruwheid, is afhankelijk van de beschikbare fabricage machines. Elke beschikbare bewerkingsmachine heeft daarom een model. Het model van een machine weet of de desbetreffende machine een fabriceerbaar object c.q. een operatie kan vervaardigen en zo ja met welke nauwkeurigheid deze dat kan. In de ontwerpfase worden, voordat een operatie wordt uitgevoerd, de beschikbare machine-modellen geraadpleegd. Het doel is, de beperkingen van de beschikbare machines te hanteren in de ontwerpfase.

De tot nu toe behandelde begrippen maken het mogelijk, al in de ontwerpfase te garanderen dat een ontwerp vervaardigd kan worden met de beschikbare machines.

De begintoestand beschrijving en de respectievelijke operaties, de fabriceerbare transformaties, de fabriceerbare objecten, en de machine modellen, maken het verder mogelijk om snel een werkvoorbereidings plan te maken. Het machine model weet niet alleen dat een fabriceerbaar object of een fabriceerbare transformatie vervaardigd kan worden, maar het weet ook hoe deze moeten worden gemaakt. De laatste stap die nog moet worden gezet voordat de fabricage kan plaatsvinden, is van werkvoorbereidings plan naar bewerkingsprogramma's voor de desbetreffende machines. De desbetreffende machine modellen zijn in staat ook deze transformatie te realiseren.

De ontwikkelde begrippen zijn geimplementeerd in algorithmen, geprogrammeerd in Smalltalk-80. Het doel van de algorithmen is de gebruikte begrippen te valideren. Het algorithme is gebruikt om een werkstuk te ontwerpen en te fabriceren. Voor de fabricage is gebruik gemaakt van een vijf-assige freesbank.

CONTENTS

Chapter 1

Introduction

One of the characteristic features that distinguishes humankind from other species it its capacity for inventiveness and creativity. The creative talent manifests itself, among other things, in the fine arts and in the ability to manufacture articles. The manufacturing process is based on the concept that an initial idea has to be transformed into a physical object. The very first artifacts were used by their creator: the hunter made and used his own flint tools. At a later stage a creator or a craftsman made implements for others to use. If a new implement was being made, the designer and the maker were the same person and so the design was inherently based on the limitations of the manufacturing process, and design proceeded on the basis of an appropriate method of fabrication.

The two processes of design and manufacturing became separated, however, as may be witnessed by the drawings of Leonardo da Vinci. Booker (1963) states: 'Leonardo's drawings of machines differ very much from similar works of the time; apart from the quality of the drawings, others were content to illustrate what they saw, whereas Leonardo used his pencil to express ideas which could be brought into existence. Many of his drawings can be classed as design sketches, sufficiently clear for the machines to be made by a good craftsman.' Indeed, the design sketches were so clear that, for example, the Science Museum (U.K.) was able to fabricate his machines centuries after they were designed.

The engineering drawing became the principal medium of communication between designer and manufacturer, and the separation of the two functions became ever greater.

The overall structure of the design and manufacturing functions and the communication between the two has not changed appreciably since Leonardo's

time. The main difference between now and then lies in the use in da Vinci's time of human or animal muscle power, or of wind or water power as prime movers.

While the functions, as such, have not changed much, there has been a great change in the manufacturing processes themselves. Machines can be programmed to perform a sequence of operations automatically; new types of machine, such as the robot, have become available; new processes, such as electrospark machining, have been developed; better tools have become available; and programming languages have been developed, such as Automatic Programmed Tools, that describe the geometry of a workpiece and that generate the tool path that the machine must use in its fabrication.

The design function, too, has become more complex, due in large measure to the increasing number of constraints imposed on the design of the product, and due to the increasing interdependence between the various parts of a given item. This increasing degree of complexity has lead to the need for a greater degree of formalization in the design process. The developments sketched out above may be expected to lead to a new medium of communication between the design and the manufacturing process. The primary means of communication, however, is still the engineering drawing. Since Leonardo's days a number of innovations and improvements have been made: standard measurements have been introduced, along with standards of tolerance and fit. Standardized styles of drawing have developed, such as the generation of projection drawings, and the geometrical phase of the design exercise has become automated. It remains a fact, however, that no matter what Computer Aided Design system is used, and no matter how such a package represents the geometrical form of an object internally, the output from the computer/designer combination is virtually always an engineering drawing.

Despite all this seeming sophistication, however, the basic concepts remain little changed since the days of da Vinci: the design and manufacturing processes are separate, and they communicate by means of the engineering drawing, or a modern equivalent of it. What has changed, particularly in the last few years, is the market situation. The current approach to the market requires short production runs, a greater product range, and a reduction in throughput time. These requirements in their turn impose the need for a less

time-consuming path from design to manufacture and this requires a study of the design and manufacturing processes and their interrelationships.

The aim of the design function is the translation of the functional specification of a product into a product that can perform the desired function. This occurs in two phases: the conceptual phase, in which the functional specifications are translated into product ideas; and the geometrical phase, in which the ideas are translated into a product design.

The manufacturing process is based on a plan that takes the design and transforms it into manufacturing processes that are executed using specified equipment. The manufacturing process plan, therefore, is a specification of the manufacturing processes required to shape a given piece of material, together with the sequence in which the processes must be used, in order to obtain the final state described by the designer. The manufacturing process plan comprises an initial description, a list of raw materials, and a specification of the processes required to achieve a given final state. The engineering drawing is a state description of the final product.

The primary aim of the design process is the description of a product in such a way that it can be fabricated. Nevertheless, there is no geometrical approach known that can handle the restrictions imposed by a given manufacturing process. These must be allowed for by the designer, who has to cope with an ever-increasing sophistication in the design process as well as with a growing need for an extensive knowledge of the manufacturing processes available. This leads, in many cases, to time-consuming and costly iterative exchanges between design and manufacturing departments. It must be apparent, therefore, that any attempt to integrate design and manufacturing should regard the restrictions imposed by the manufacturing process as a key issue.

Another point that should be taken into consideration is the fact that most representations of the design of a product are in fact final state descriptions of the product. In order to fabricate it, a manufacturing process plan has to be created ab initio.

Van 't Erve (1988) has shown that it is possible to generate manufacturing process plans by recognizing certain features in a final state design. However,

he states: 'Although the feature concept is very promising, automatic feature recognition still poses a lot of problems. The complexity of the required algorithms grows almost exponentially with the capacity to recognize more complicated features.' Human intervention would therefore seem to be necessary in this approach, too.

Since no geometrical design approach is known which handles manufacturing restrictions, a new approach to the integration of design and manufacturing has been developed, using the Process Interaction Approach (Overwater 1987, Rooda 1987, Wortmann, Rooda, Boot 1989).

In Chapter 2 a model of a product generator is presented the purpose of which is to reveal the interdependence between the design and the manufacturing process. The model also reveals that a full integration of the geometrical design function and the manufacturing process planning function is impossible. Based on the insights gathered from the product generator model developed in Chapter 2, Chapter 3 goes on to explain and discuss the design approach chosen. Chapter 4, again on the basis of the insights developed in Chapter 2, discusses the selected manufacturing process planning approach.

Both the design approach and the manufacturing process planning approach have been implemented using the Smalltalk-80 object-oriented programming environment (Goldberg 1984, Goldberg, Robson 1985). Chapter 5 presents a case study of the ideas developed here in Smalltalk-80, and Chapter 6 gives a summary of the whole study.

Chapter 2

A Product Generator

In this chapter we isolate the most important activities that occur during the product development process and reveal their interdependence within the context of an integrated approach to product design and manufacture.

There are five phases in the life cycle of any product: the orientation phase, the specification phase, the realization phase, the utilization phase, and the elimination phase. Of these five, only three are important during the development of a product: the orientation, specification, and realization phases. The orientation phase commences with the idea that a product is required. Its functions must be described so that a specification can be generated. The specification phase transforms the description of the functions into a product specification which comprises a design and a manufacturing process plan. The final phase in the generation of a product is the realization phase, in which the specifications are translated into a physical product. The operational phases follow on after the development phases. We shall treat the three developmental phases in the following sections.

All the activities that occur during the development of a product are closely intertwined, as will be shown more fully below. For this reason, a systems approach (Bertalanffy 1968) must be adopted. One such approach is the Process Interaction Approach (Overwater 1987, Rooda 1987, Wortmann, Rooda, Boot 1989), which has proved its worth in this area. In the context of the current study, the Process Interaction Approach is a suitable method for the functional specification of industrial systems and for their design, realization, and control. Industrial systems, in the Process Interaction Approach, are considered as a collection of concurrent processes together with associated interactions. A model of an industrial system comprises a collection of passive and active elements and the relationships between them. The relationships can be between the elements of the system, or with elements that are outside the

system. The Process Interaction Approach, together with the necessary concepts, will be discussed in the next section. The approach has been applied to the manufacturing and design functions in order to elucidate their mutual interdependence before going on to describe a product generator in Section 2.3. A product generator is a processor that can transform the functional specification of a product into a product that can perform the required functions. As inputs the product generator takes the functional specification and the raw materials, producing the finished product – or a materialization of the functional specifications – as output. The results of the modelling of a product generator are discussed in Section 2.4.

2.1. Processors and Interactions

A system can be described as a system of mutually interrelated elements (Bertalanffy 1968). The relationships within a system describe the coherence between the elements; i.e. they determine the system's structure and its behaviour. The relationships of the system with external elements determine its purpose.

The Process Interaction Approach, as defined in Wortmann, Rooda, Boot (1989) is summarized below.

Processors

Active elements, called processors, are elements that are capable of changing the state of a system by the performance of actions. The behaviour of a processor is described by the processor's model. There are two kinds of processor: expanded processors and leaf processors. An expanded processor consists of sub-processors and their interactions. Sub-processors are termed child processors. The expanded processors are called parent processors. The model of an expanded processor therefore consists of a collection of processors and interaction paths.

Leaf processors are not expanded, and their models are process descriptions. Passive elements are not capable of changing the state of a system; their importance lies in their presence or in the value assigned to them.

Actions are changes of state of a system. An action can change the value assigned to an element, or it can add elements to or remove elements from the system. A process is the name given to the collection of actions performed by a processor.

Interactions

Interactions exchange passive elements between processors. Their aim is the synchronization of two or more processors, or the communication between processors. A processor can only perform two types of interaction: send actions and receive actions. Send actions and receive actions specify which interaction paths are suitable for the interaction. A send action makes an object available for interaction. A receive action transports an object that is available for interaction to the requesting processor via the specified interaction path.

Only one type of interaction mechanism is available: the synchronous mechanism, a characteristic feature of which is that the processor which performs the send action is blocked until a corresponding receive action is performed, and vice versa. The transfer of an object from one processor to another requires no time, provided both processors are capable of sending or receiving the object.

Processors are provided with named send and receive ports. The send and receive actions specify the ports that an interaction may use and, after selection, determine which interaction paths are suitable. A specific port functions only as a receive port or as a send port.

Interaction paths specify the connection between two ports of two different processors. Interaction paths are named, and they are directed. They have a send port as their origin, and they terminate at a receive port.

The model of an expanded processor

As mentioned above, processors can be expanded, meaning that they consist of a parent processor and one or more child processors together with their mutual

interaction paths. Furthermore, the child processors themselves can be expanded processors. A processor that is not expanded is called a leaf processor. All the child processors and the appropriate interaction paths that make up one expanded processor are called a level. The use of these concepts allows the top-down design of industrial systems.

Interaction ports, when connected to a processor, are an important part of the environment of a processor and, combined with the model of a processor, they determine its functionality.

The interaction ports of an expanded processor are actually connected to the expanded processor's children. The child processors of the expanded processor perform the send and receive actions. The send and receive ports of the child processors and the expanded processor are connected via external interaction paths. These have the same name as the send or the receive port of the expanded processor to which they are connected.

The model of a leaf processor

In this work descriptions are given in an object oriented language based on Smalltalk-80. (See Appendix I.) The syntax of an expression consists of an object (such as a variable) followed by a message. When the expression is evaluated it returns an object, which does not have to be the same object as the one in the expression.

A message consists of a function selector with or without associated arguments. For example, the expression "aRobot location" returns the location, position, and orientation of the object aRobot. The function selector is "location" and the message has no arguments. The expression "aMillingMachine moveToAxisVector: anAxisVector" informs the object aMillingMachine that it has to move itself to the axis vector anAxisVector. The function selector of the message is "moveToAxisVector:" and the argument is "anAxisVector".

Self reference, reference to the receiver of a message, is allowed by the use of a special variable "self".

The syntax of some commonly used control structures in Smalltalk-80 is presented below. (See also Appendix I.)

condition **ifTrue:** [trueBlock] **ifFalse:** [falseBlock]

Depending on the value of the condition (true or false) the block named "trueBlock" or the block named "falseBlock" is evaluated.

[conditionBlock] **whileTrue:** [trueBlock]

So long as the conditionBlock remains true the block named "trueBlock" is evaluated.

[conditionBlock] **whileFalse:** [falseBlock]

So long as the conditionBlock is false the block named "falseBlock" is evaluated.

Send and receive actions are frequently used. The syntax of the simplest actions is defined below.

A receive action:

self **receiveFrom:** receivePortName

Receive an object from the receive port named receivePortName.
A send action:

self **send:** object **to:** sendPortName

Send an object to the send port named sendPortName.

By way of illustration, a model of a processor will now be described. A robot, controlled by the robot processor described, takes products from a conveyor belt at a specific location and sets them down at a location that depends on the product selected.

The symbol ← represents an assignment (Appendix I).

robot model

```
body
    productManufacturingInformation ← self receiveFrom: productport.
    self gripper isClosed ifTrue: [self openGripper].
    self moveToPickUpPosition.
    self closeGripper.
    self moveTo: (productManufacturingInformation desiredPosition).
    self openGripper.
    self moveToSafetyPosition.
    self send: productManufacturingInformation to: nextMachinePort
```

The graphical representation of a model

The graphical representation of a model indicates processors by the use of bubbles and interactions by the use of arrows. See Figure 2.1.

Each bubble contains a name for identification purposes and the name of the corresponding process description of the processor. Each interaction path has a name attached to it. Finally, the receive port and the send port at the terminus and at the beginning of each interaction path are named. If the name of the processor is the same as the name of the process description, then only the processor name is shown, which means that only a single processor having that process description is present.

2.2. The Cybernetic Architecture

The way in which systems are structured so that they may cope with their environment is called systems architecture. A variety of architectures relate to the way in which systems process information from their environment in order to achieve their goals. An important architecture is the cybernetic architecure (Wiener 1955). This consists of a processor to be controlled and a control processor. The control processor knows the system's goal (in some sense) and influences the controlled processor in such a way that the system's goal can be achieved. The control cycle can be generalized as follows (Figure 2.2): the goal of system has not been realized and a problem therefore exists.

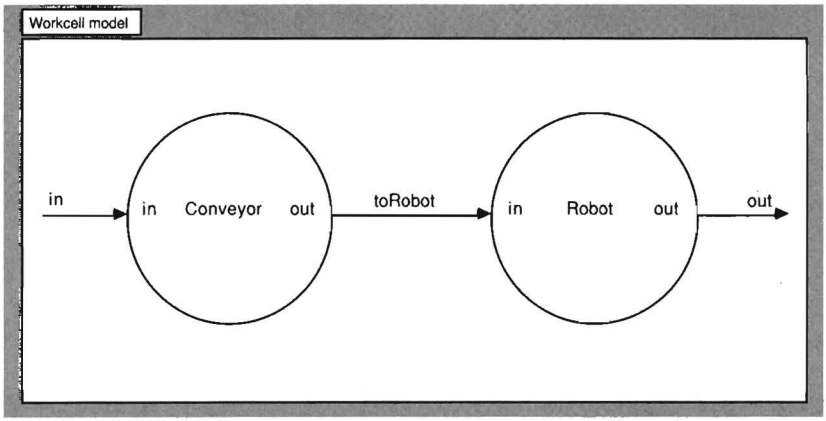


Figure 2.1. The graphical representation of a model

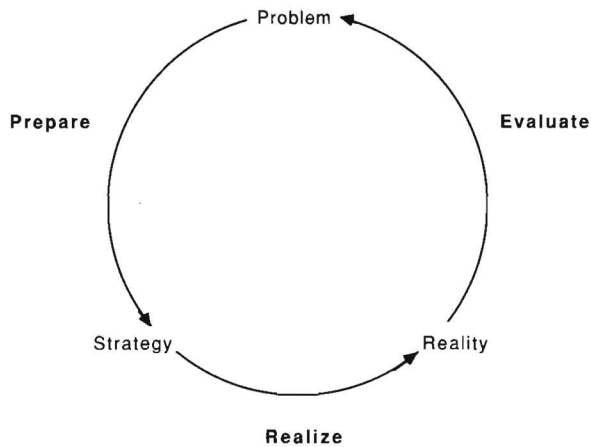


Figure 2.2. The control cycle

The control processor prepares a strategy that influences the controlled processor. The controlled processor now executes the strategy. The system's goals should now be achieved, or at least the difference between the desired state and

the achieved state should have been reduced. The control processor evaluates the state achieved by the system and decides whether there is a problem. If so, the control processor prepares a strategy. This control cycle continues for as long as there is a deviation from the system's goals.

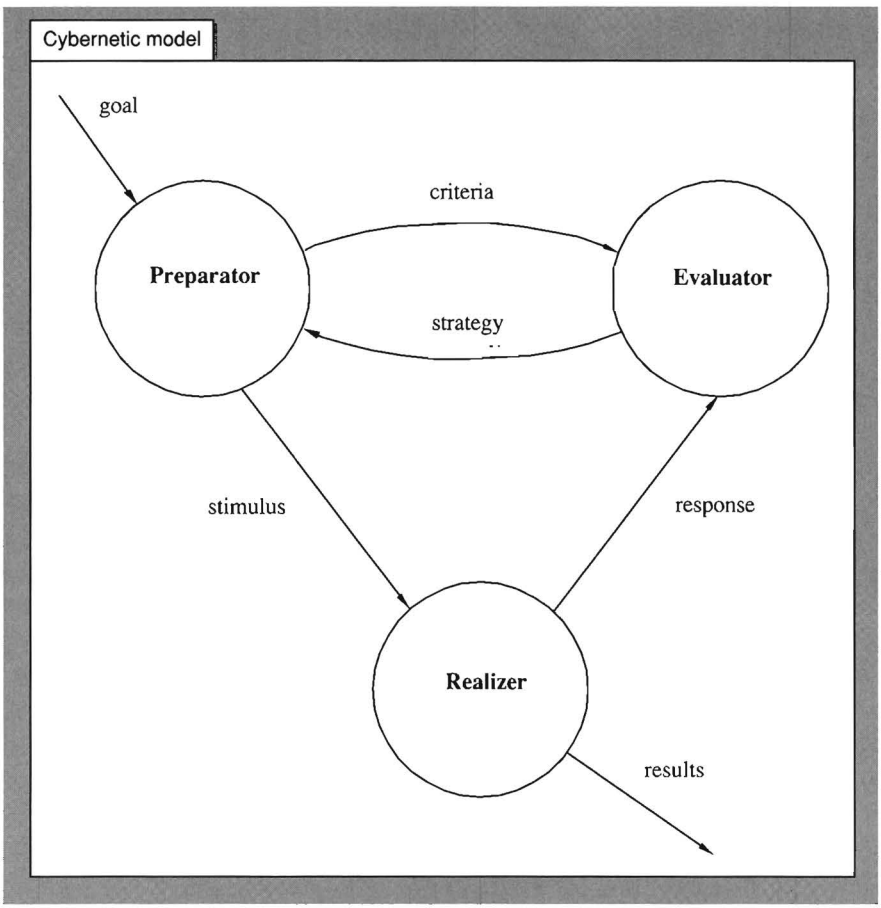


Figure 2.3. The cybernetic architecture

The strategy that is prepared by the control processor decreases in effectiveness if the controlled processor does not behave in the expected manner.

A Process Interaction model of the cybernetic architecture is shown in Figure 2.3 (Haterd 1988).

The cybernetic architecture model consists of three parallel processors: the Preparator, the Realizer, and the Evaluator. The control processor comprises the Preparator and the Evaluator. The controlled processor consists of the Realizer. The Preparator processor prepares the strategy, the Realizer employs the strategy and the Evaluator evaluates the current situation and decides whether there is a problem.

2.3. The ProductGenerator Model

Both the design function and the manufacturing planning function are mutually interwoven with other functions. In order to identify these functions and their relationships a product generator function has been modeled. This is considered as a 'black box', called 'ProductGenerator', which performs its function in order to achieve a stated goal, the transformation of a product functional specification into a product. The product generator model, including the models of the child processors, may be found in Appendix II.

It is unnecessary to model a complete product generator function. Within the context of this study, only the essential functions and their mutual interrelationships have been modeled. A number of logistic functions, such as requirement planning, scheduling, route planning, capacity planning, expediting, inventory control and management, have therefore been omitted from the model.

It has been assumed for the purposes of the model that a customer supplies the functional description of the product to the product generator. The functions from the description are then transformed into a product by the product generator function.

The context of the product generator function determines the relationships between the black box called 'Productgenerator' and its environment. This is illustrated in the graphical representation of the product generator processor in Figure 2.4.

The ProductGenerator processor receives the product functions and the raw material. When the product has been manufactured in such a way that it achieves the required functions, it is removed from the ProductGenerator processor.

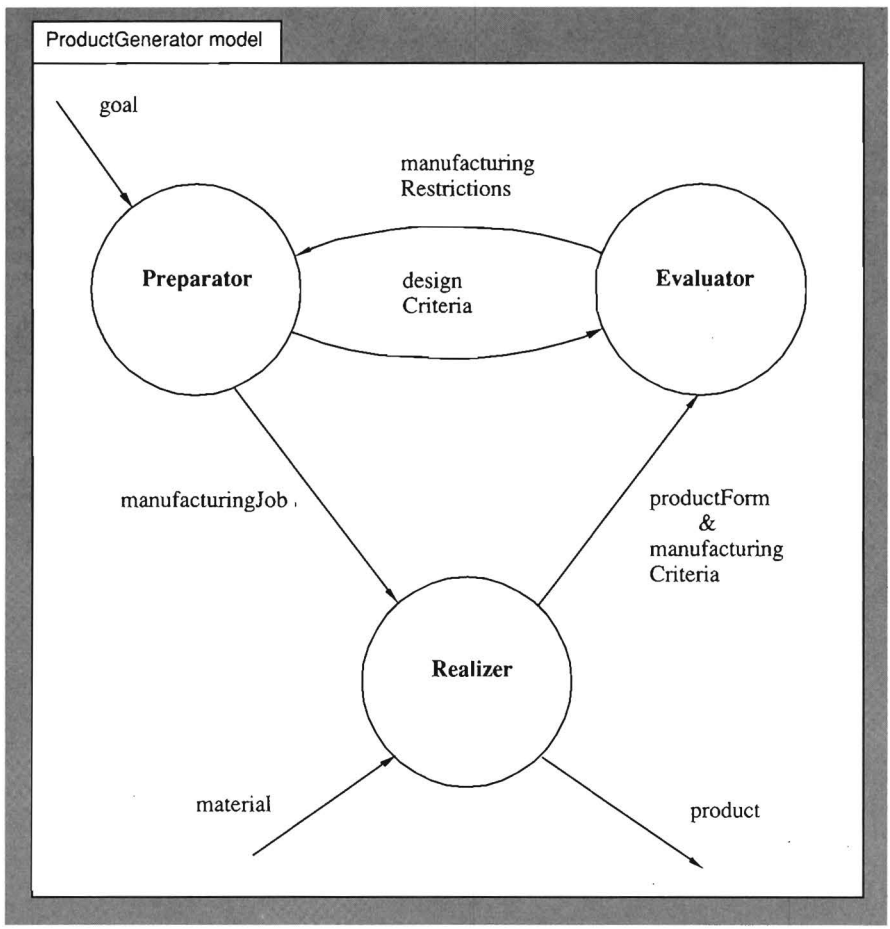


Figure 2.4. The ProductGenerator model

On closer inspection of the ProductGenerator processor, one may distinguish a cybernetic architecture as discussed in Section 2.2.

The Preparator processor prepares the manufacturing of the product. The Realizer processor manufactures the product. The Evaluator processor evaluates the manufacturing and the design processes. This evaluation results in manufacturing restrictions, which are also accumulated from the experience gained from the design and manufacture of previous products.

The Preparator model consists of two processors: a Designer processor and a ManufacturingProcessPlanner processor, with one interaction path: the generateManufacturingPlan interaction path (Figure 2.5).

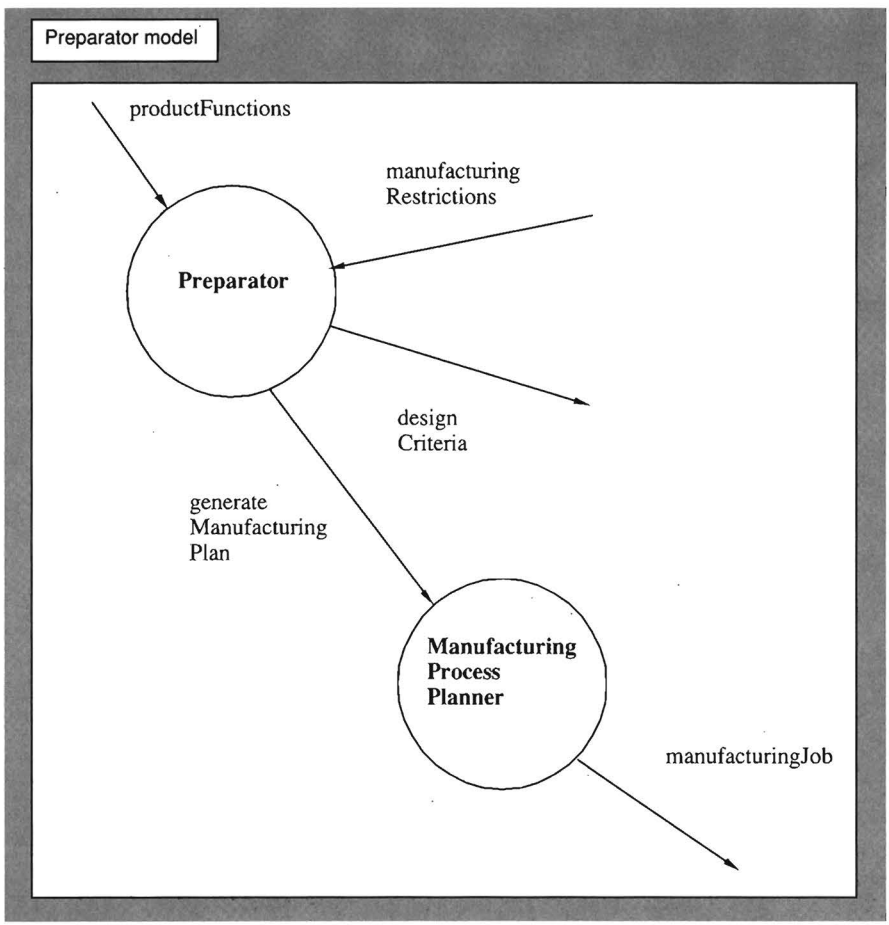


Figure 2.5. The Preparator model

The Designer processor designs the product after receiving the functions of the as yet unfabricated product. The Designer processor inquires about any new manufacturing restrictions that may apply, and updates its knowledge base.

Within the context of this study, design consists of two phases: the conceptual phase and the geometrical phase. The conceptual phase is that phase in which the designer conceptualizes the new product, trying to discover a method to achieve the product's functions. Many ideas and concepts will be formulated and evaluated and the best conceptual formulations will be chosen.

The geometrical phase is that phase in which the concept of the new product is translated into geometrical objects such as shapes, parts, tolerances and fits. The solution chosen will prescribe the individual engineering tasks at a lower level, and so the tasks will be described in this hierarchical manner until the final task is established. In this geometrical phase, the experience gathered from the fabrication of earlier designs and the concomitant manufacturing restrictions are taken into account in the process of developing designs that can successfully be manufactured.

When the design phase has been completed, an order to generate the manufacturing process plan is issued. The ManufacturingProcessPlanner processor then generates the manufacturing process plan by first planning the sequence of operations and the types of manufacturing processes necessary to fabricate the product. The constraints on this planning process are the desired quantity, quality, available facilities, tooling and labour. When it has been determined what available devices, tool sets, and setups are to be used for each part of the task, the order togenerate the programs for the equipment is given.

If necessary, a model of the work cell of the equipment to be used is generated, which can be used to validate the machine programs generated, the selected tool sets, equipment and setups.

Note that there is, to some extent, an overlap between the existing manufacturing processes and hence it is almost always possible to replace one manufacturing process by another. For example, a hole can be fabricated either with a drill or a reamer. The result of this overlap is a combinatorial explosion of the number of ways in which a given product can be manufactured.

When the manufacturing process plan has been generated and validated, the Realizer process receives the order to manufacture the product according to its design. This is done by executing the manufacturing process plan that has been generated, using the specified equipment, after having obtained the necessary materials.

After the product has been manufactured according to the machine programs, the geometrical form of the product has to be determined. The geometrical form generated is evaluated against the design as reference. If the geometrical form does not match the designed form, then a correction cycle has to be executed.

This loop continues until the geometrical form matches the required form of the product, as specified by the design, and no further corrections are thus needed. The product has now been satisfactorily manufactured, and may be sold to the customer.

The Realizer process then informs the Evaluator about the manufactured form of the product and the way in which it has been manufactured, which are both of great interest, since the Evaluator processor generates its knowledge base of manufacturing restrictions on the basis of manufactured form, design criteria, and manufacturing criteria. The design criteria and the manufacturing criteria interactions contain knowledge that may be applied to future design and manufacturing processes, respectively.

The model contains two major feedback loops. The first occurs in the Realizer processor, and may be found represented in Appendix II. The manufactured product is compared with the design and, should any deviations occur from the design, corrections are implemented.

The second major feedback loop (Figure 2.6) runs as follows: from the Preparator child processor Designer to the Preparator child processor ManufacturingProcessPlanner, using the interaction path generateManufacturingPlan; from the ManufacturingProcessPlanner processor to the Realizer processor using the manufacturingJob interaction path; from the Realizer processor to the Evaluator processor using the productForm&manufacturingCriteria in-

teraction path; and back to the Preparator child processor Designer using the manufacturingRestrictions interaction path.

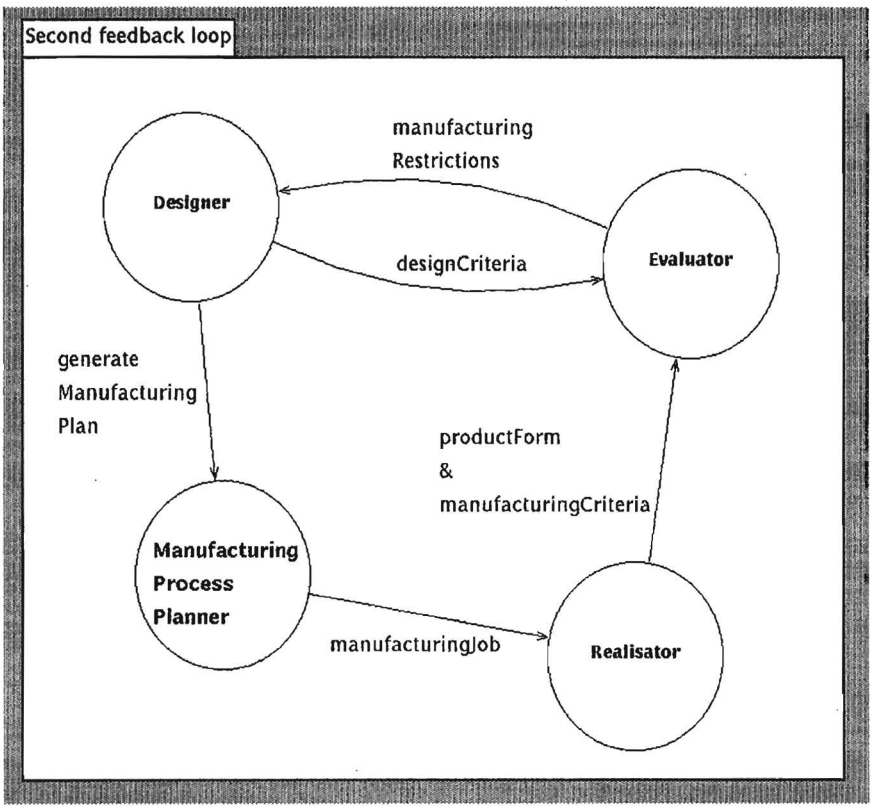


Figure 2.6. The second feedback loop

This last loop imposes manufacturing restrictions on the Designer processor. Furthermore, it initiates a learning system since the Designer processor of the Preparator learns from the stored knowledge of products that have been previously designed and made because the Evaluator processor evaluates the productForm&manufacturingCriteria interaction and generates the interaction manufacturingRestrictions.

The first feedback loop is quite common in factories, an example being the implementation of a ProductGenerator processor.

The second feedback loop is new to the area of the geometrical approaches to design. Manufacturing restrictions have not hitherto been handled within a geometrical approach to design: this has always had to be left to human intervention.

2.4. Summary

The systems approach described in this chapter shows that the integration of design and manufacturing depends on more than the preparatory and the realization functions. The evaluation function and the manufacturing restrictions interaction are often neglected when attempting to integrate design and manufacturing.

Some of the functions described in the model developed here cannot yet be formalized, since they belong to that group of poorly understood tasks that can be performed by humans with relative ease. For instance, the design function, viewed as a preparatory function, can be an inventive function; the evaluative function is one of pattern recognition. Neitheris particularly well understood and cannot therefore be formalized to any great extent.

If the contents of the designCriteria interactions and the manufacturingCriteria part of the productForm&manufacturingCriteria interaction were to be formalized, this could lead to the formalization of the evaluation function, which prompts a second remark: some interactions are very difficult to formalize due to the very high level of abstraction of human thought.

The model of the product generator contains feedback loops which incorporate functions and interactions that cannot be formalized: or not at present, in any case. It would seem, therefore, that insofar as such matters cannot be formalized, completely integrated manufacturing without the necessity for human intervention is not yet possible.

Mechanical design is a creation of the human mind, and it is only constrained by the demand that the functional specification of the new product has to be realized. How the design is executed depends on the designer. In contrast to design, however, manufacturing processes are subject to physical limitations: for instance, only a certain geometrical accuracy is achievable.

The geometrical modelling function must therefore be structured in such a way that manufacturing restrictions are taken into account in any attempt to integrate design with manufacturing.

In the next chapter such an approach to geometrical design will be introduced, one which does take account of the manufacturing restrictions.

Chapter 3

The Design Process

This chapter is devoted to an examination of the overall structure of the design process and of the way in which the design process can be integrated with the manufacturing process planning function. In addition, a concept is proposed for a geometrical design system that can achieve the required integration, taking account of the restrictions imposed by the manufacturing processes.

The process of design commences with the recognition of the need for a particular product. This is followed by the conception of an idea that will fulfill that need. In other words, design starts with the recognition of a problem, proceeds with a definition of the problem, passes through a development program, and ends with the fabrication and assessment of the product.

Within the context of this study, the design function is considered to consist of two phases: first the conceptual phase, in which the need for a product is transformed into ideas; and then the geometrical phase, in which the ideas are translated into a design. The conceptual phase results in a set of possible solutions to the problem posed. These are then evaluated and validated within the context of the following phase: the solution that is adopted must be the one that is most suitable for adoption in the geometrical phase of the process.

The geometrical phase deals with physical function, form, fit, tolerance, weight, stiffness and so on. These physical qualities are the means by which the functions that the product has to perform may be realized.

Since the conceptual phase is in large measure a creative and an inventive process, it cannot readily be formalized. It must therefore be left out of account in any attempt to integrate design with manufacture. It is the geometrical phase that offers the best avenue of approach in this direction.

The primary purpose of the geometrical phase of the design process is the generation of an unambiguous, complete representation of the product; one, furthermore, that can be manufactured. Besides this, the representation must be suitable for use as input to other, down-line functions, such as the manufacturing planning function. For this reason, the product representation must include such data as dimensions, tolerances and fits, as well as material specifications. Above all, in order to reduce the need for human intervention, the product representation has to be as complete as possible.

The need for a complete and unambiguous product representation should be self evident. The problem is, however, that it is relatively simple to produce an inherently ambiguous product representation, as will be shown below.

The requirement that a product representation must represent a product that is capable of being manufactured necessitates the enforcement of manufacturing restrictions at the geometrical stage of the design process. The question then inevitably arises: what kind of manufacturing restrictions are relevant, and how may they be utilized?

The major manufacturing restrictions are those due to forms that cannot be fabricated, and these fall into three categories: those that cannot be made, no matter what technology or equipment is used; forms that cannot be manufactured by a specific technology; and forms that cannot be manufactured by a specific machine or equipment.

The notion that a given form cannot be manufactured, no matter what technology, process or equipment is used, is a limitation that is more apparent than real since, given suitable materials, the inventiveness of the human mind permits the fabrication of virtually any conceivable form.

An example of a form that cannot be fabricated by a specific technology is presented in Figure 3.1.

The cavity cannot be milled, on the assumption that the tolerances specified are much smaller than the radius of the smallest available mill, since two of the corners are not rounded. The piece could be fabricated by electrospark erosion, however.

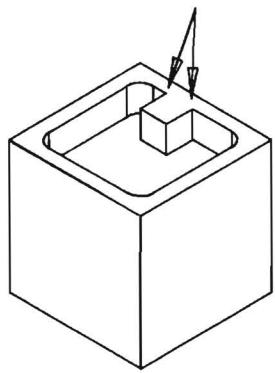


Figure 3.1. A cavity that cannot be milled

The final category–forms that cannot be fabricated by a specific machine or equipment–is of minor importance when determining whether a design can be manufactured. It is almost always possible to select a suitable machine. It is shown below that manufacturing restrictions that depend on the equipment to be used can be incorporated into the geometrical design phase.

If one assumes that all geometrical forms can be constructed by using a limited number of basic objects, then the second category allows the use of a structured approach to the enforcement of manufacturing restrictions in the design phase. These basic objects, defined as geometrical forms that can be manufactured, are called Manufacturable Objects. One can only establish empirically that a geometrical form is a Manufacturable Object.

The Manufacturable Object concept is the design and manufacturing planning counterpart of the application of a combination of one or more tools, machines and setups in the manufacturing phase. Examples of Manufacturable Objects are cavities, slots, shapes obtainable by bending, or assemblies.

The concept of a Manufacturable Object consists of a geometrical form together with its application rules. The application rules ensure that a design can be manufactured. Figure 3.2 shows a form that can be manufactured in principal, although the application is incorrect.

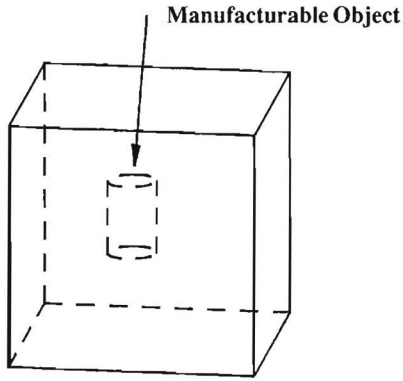


Figure 3.2. An incorrect application for a Manufacturable Object

An example of the use of an application rule is given in Figure 3.3.

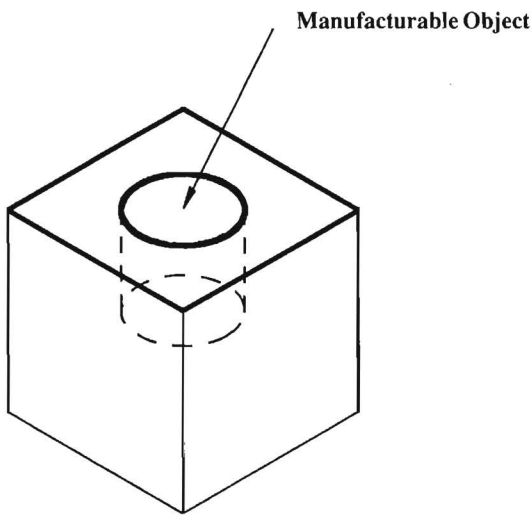


Figure 3.3. An example of an application rule

Figure 3.3 shows the removal of a cylinder from a cube. One application rule that guarantees that the removal can in fact be performed is: at least one planar face of the cylinder to be removed must coincide with a face of the cube.

There are two distinct aspects to the notion of a Manufacturable Object: one relating to the geometrical design and one to the manufacturing process planning. In the geometrical design phase it is only necessary to be certain that a Manufacturable Object has a stated geometrical form and that it can indeed be fabricated. In other words, it is only the final state of the Manufacturable Object that is of importance. In the manufacturing planning process, however, it is not the final state that is important, but rather the way in which that final state can be achieved. Thus, in the manufacturing process planning phase, the knowledge of how a Manufacturable Object may be fabricated must be available. In the design phase certain parameters relevant to the accuracy achievable with a given Manufacturable Object must be available, such as surface roughness and fit. This may be achieved by the incorporation, within the design approach, of a model that 'knows' the accuracy of which the machine is capable and that 'knows' how each Manufacturable Object must be fabricated according to its specifications.

A Manufacturing Machine Model is thus a formalization of the manufacturing abilities of a given type of manufacturing machine.

Every manufacturing process, such as turning, milling, and so on, generates surfaces, so another consequence of the demand that the product description that is produced in the geometrical design phase must be capable of fabrication is that surfaces have to be dealt with in the design phase. Edges and vertices are by-products of the fabrication of surfaces.

The requirement that a product description must be suitable as input to other, down-line functions, combined with the requirement that the geometrical design phase must generate a description of a product that can actually be fabricated, imposes a certain requirement on the internal geometrical representation of the design. The manufacturing function and, thus, the manufacturing process planning function, needs a representation that is based on the initial state of the product and the changes through which the various phases of the design pass on the way to the realization of the final object. The manufacturing

function transforms raw material through a succession of operations, which are specified in the state description emanating from the geometrical design phase.

The generation of a manufacturing process plan is still a skilled task which can be materially assisted by a clear state description and its state changes. Most geometrical design representations, however, pass descriptions of only the final state of an object to be fabricated to the manufacturing process planning phase.

In order to determine what kinds of geometrical modelling methods are suitable for the present purposes, the two main categories will now be examined. These are a combination of engineering drawings and wire frame representations, and solid models. The main distinction between them is ambiguity: the engineering drawing and wire frame are inherently ambiguous (Arbab 1982, Campman 1987). Nonetheless, the engineering drawing is still the most commonly used design representation.

The following two sections will describe these two categories in greater detail and their suitability will also be discussed.

3.1. Engineering Drawings and Wire Frames

The engineering drawing is the oldest geometrical representation of a design used in mechanical engineering. As has been stated in Chapter 1, Leonardo da Vinci was the first person to draw design sketches that were sufficiently clear that a craftsman could use them (Booker 1963).

An engineering drawing consists of an adequate number of views of a design, each view being a projection of a three dimensional object onto a plane surface. A view is drawn using points, lines and curves. Unfortunately, not every set of points, lines and curves is a legitimate view of a physical object. Furthermore, no two projection views of a physical object can be completely independent. A set of consistent, legitimate views has therefore to be produced. It is unfortunately virtually impossible for a human to maintain the consistency of a set of engineering drawings during the design process. Inconsistencies are always

present, and they cause problems during the manufacturing process planning phase. In this phase, an inverse projection has to be created. The views have to be combined into a three dimensional model of the object and any inconsistencies in the set of views supplied must be resolved by reference to the designer's intentions and on the basis of manufacturing experience: by the use of 'common sense', in other words. Unfortunately, 'common sense' is very difficult to formalize.

There are other severe drawbacks to engineering drawings. First of all, not every physical object can be unambiguously represented by a set of engineering drawings. Physical objects that are constructed of surfaces that are smoothly curved in two directions cannot be unambiguously represented by any finite number of views, although it is perfectly possible to define them exactly.

Second, engineering drawings are based on vertices and edges, but these are by-products of manufacturing processes, which produce surfaces. The surfaces, however, can only be inferred from the engineering drawings.

Third, it is entirely possible to produce an engineering drawing of an object that cannot be fabricated. Figure 3.4 shows an engineering drawing of an object that cannot be produced in steel (for instance), since the axle, as drawn, cannot be inserted into the hole.

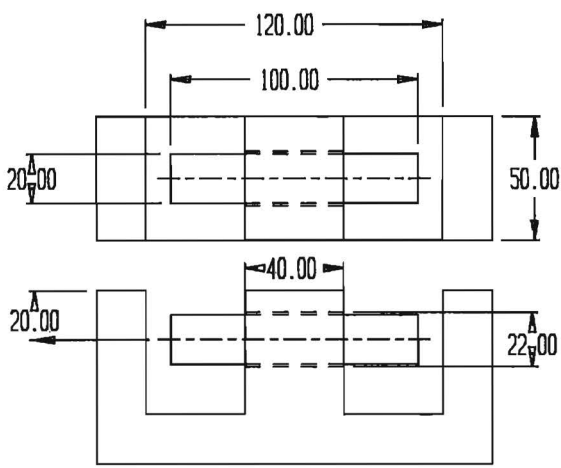


Figure 3.4. An engineering drawing which cannot be manufactured

Finally, engineering drawings are passed to the manufacturing planning office as final state descriptions. The manufacturing planning process must therefore generate the manufacturing plan ab initio.

It should now be apparent, in view of the requirements stated at the beginning of this chapter, why engineering drawings cannot be considered as a suitable medium for the transmission of design information.

We turn now to a discussion of the wire frame representation of an object. This consists of a set of spatial curves, called wires, that represent the edges of the physical solid. An edge is a curve on the boundary of the physical object along which the derivative of the surface is discontinuous. Wire frame models are easy to work with, although they do not capture enough of the shape properties of a physical object to convey the notion of solidity, since they do not model surfaces, which leads to the type of ambiguity illustrated in Figure 3.5.

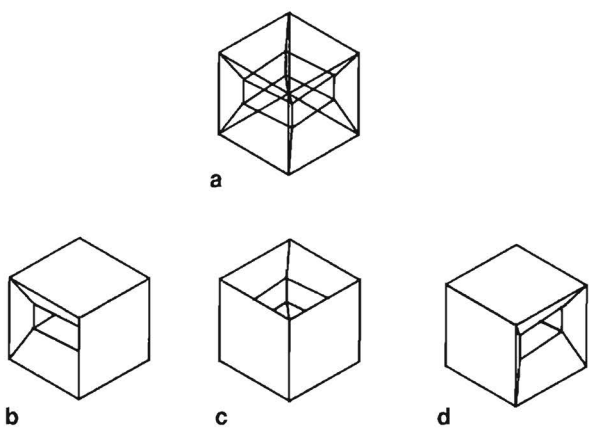


Figure 3.5. An ambiguous wire-frame

Figure 3.5 shows three physical solids (b, c, d) which have the same wire frame representation (a). Wire frames are even less informative than engineering drawings when describing the boundary surfaces of physical objects. In partic-

ular, they cannot handle curved surfaces. For example, the only wires that can be associated with a finite, circular cylinder are two parallel circles. But this is the same representation as, among other things, two cones, as is illustrated in Figure 3.6.

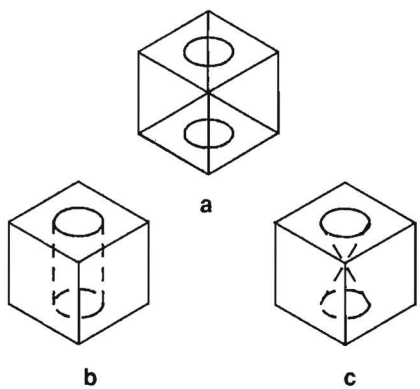


Figure 3.6. A wire-frame model of curved surfaces

Very smoothly curved surfaces, such as spheres, have no edges to represent at all. They must be represented by the addition of wires where no real edges are present.

In conclusion we can state that the same wire frame may represent more than one physical, solid object. Furthermore, it is passed on to the manufacturing planning department as a final state description of the object. Wire frames, therefore, cannot be considered to be suitable for the integration of design with manufacturing.

3.2. Solid Modelling

Solid modelling refers to a class of geometrical models that unambiguously represent the shape of physical solid objects. An important characteristic of a

physical solid object is its geometric form. Physical solids are therefore modelled as subsets of Euclidean three dimensional space, the geometrical properties of which correspond to those of the modelled objects. One such set of geometric properties is discussed in Requicha (1980), whose discussion, somewhat paraphrased, is used in what follows.

- *Rigidity:* rigidity refers to the property of the invariance of shape with respect to locating operations such as positioning and orientating.
- *Homogeneous three dimensionality:* a solid must occupy a volume. It can inevitably have no isolated or 'dangling' boundary segments, nor infinitely thin cracks.
- *Finiteness:* a solid must occupy a finite amount of space.
- *Closure under physical operations:* the result of the performance of such physical operations on a solid as relocation, or the addition and removal of solid material, must again be a solid object.
- *Boundary determinism:* the boundary of a solid must unambiguously determine its exterior and its interior.

Requicha (1977) argues that a suitable model for physical solids is the set of all closed subsets of E^3 (i.e. Euclidean three dimensional space).

(See Appendix III for the formal definitions.) A set is closed if it contains its own boundary. Furthermore, a set of mathematical operators that operate on these closed sets is required in order to represent the result of manufacturing processes such as gluing, material removal, or assembly, and also to solve such shape-related operations as checking for interference and adjacency.

The set-theoretic operations that are obvious candidates for such a task are the operations of union, intersection and subtraction. However, closed sets do not remain closed under these operations, as can be seen in Figure 3.7.

The subtraction operation in particular can cause fundamental problems, since the result of its application is unbounded. Closed sets are therefore not suitable for the consistent modelling of physical objects.

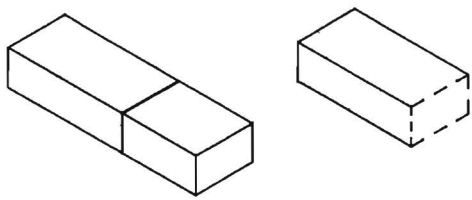


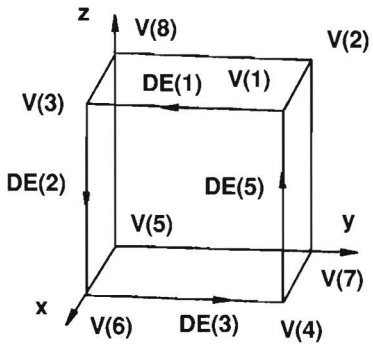
Figure 3.7. The subtraction operation performed with closed sets

The notion of a physical boundary is quite 'fuzzy' at a microscopic scale. When modelling physical objects, their boundary should therefore be regarded as an abstract concept rather than a physical entity. The ideal geometric shape and the position of the boundary can only be approximated to any required degree of precision. Physical objects are therefore modelled as open sets of material points bounded by surfaces, edges and vertices, which they do not contain and which do separate them from another set of void points in space. The two immediate consequences of this viewpoint are that surfaces, edges and vertices become abstract delimiters that separate two regions of space – an occupied and a void region – and, since objects do not contain their boundaries, they cannot be connected at infinitely thin regions, such as corners or edges (Arbab 1982).

A large number of different solid modelling representations have been devised that satisfy the above properties. The two most promising representations, Constructive Solid Modelling and Boundary Representation, will be discussed below.

Boundary Representation is a method for describing a physical solid object in terms of its topological boundary. This boundary is divided into a finite number of faces, each of which can be defined in turn in different ways. One popular method is the representation of each face in terms of its boundary edges and vertices (Braid 1974), see Figure 3.8.

Only the hull of the solid object is described, yet it is possible to determine its interior or its exterior without ambiguity.



Nr.	x	y	z
1	100	100	100
2	0	100	100
3	100	0	100
4	100	100	0
5	0	0	0
6	100	0	0
7	0	100	0
8	0	0	100

Figure 3.8a. Vertices list

Nr.	vertex1	vertex2
1	1	3
2	3	6
3	6	4
4	7	4
5	4	1
6	1	2
7	8	2
8	2	7
9	7	5
10	6	5
11	5	8
12	8	3

Figure 3.8b. Edges

Nr.	edge	same direction
1	1	true
2	2	true
3	3	true
4	4	true
5	5	true
6	6	true
7	7	true
8	8	true
9	9	true
10	10	true
11	11	true
12	12	true
13	1	false
14	2	false
15	3	false
16	4	false
17	5	false
18	6	false
19	7	false
20	8	false
21	9	false
22	10	false
23	11	false
24	12	false

Figure 3.8c. Directed edges list

Planar face equation: $A * x + B * y + C * z + D = 0$

Nr.	A	B	C	D	Directed edges			
1	1	0	0	−100	1	2	3	5
2	0	1	0	−100	20	18	17	16
3	0	0	1	−100	13	6	19	12
4	−1	0	0	0	7	8	9	11
5	0	−1	0	0	14	24	23	22
6	0	0	−1	0	10	21	4	15

Figure 3.8d. Faces list

Figure 3.8. A Boundary Representation of a cube

Since manufacturing processes generate surfaces, the Boundary representation comes closest to a geometrical model that is directly suitable for input to the manufacturing process planning. It does, however, have one major drawback: it is a final state description of the object to be fabricated. Only the final state of the design is passed on to the manufacturing process planning phase, and so the manufacturing process plan, which is required for the fabrication of the final state, must be generated ab initio. In the discussion of engineering drawings above, we stated that generative manufacturing process planning remains a skilled task. The same reasoning would apply if all the intermediate states, as well as the final state, were passed on to the manufacturing process planning department. As such, therefore, Boundary Representation is not a suitable method for our purposes.

Constructive Solid Modelling is based on the fundamental concept that a solid object can be represented as a series of additions and subtractions of various simpler solids. A representation of a physical solid object can be visualized in the form of a tree structure the leaves of which are primitive solids, the branches being nodes where operations are performed on the solids. This is illustrated in Figure 3.9.

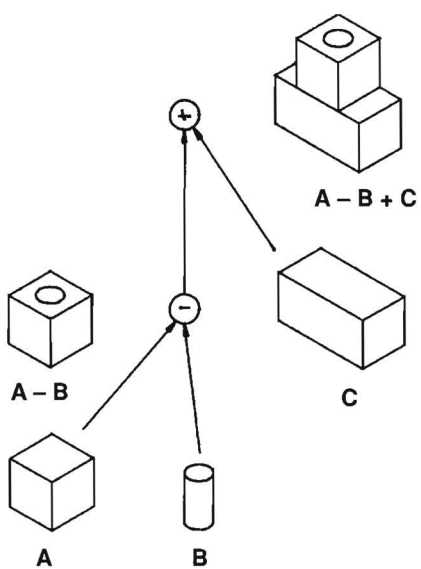


Figure 3.9. A Constructive Solid Modelling tree

Constructive Solid Modelling representations are complete and unambiguous, but they are not unique. As shown in Figure 3.10, a number of possible Constructive Solid Modelling representations may represent the one physical solid object. In fact, the number of representations can be infinite.

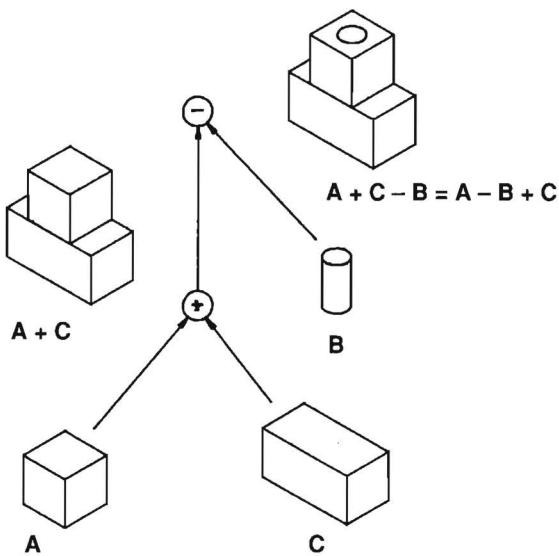


Figure 3.10. A different Constructive Solid Modelling tree for the same final geometrical form

The variety of representations of the same physical object is one of the ways in which alternative manufacturing process plans may be generated.

One disadvantage of this method of representing the design of products is that it is possible to generate a representation that bears no relationship to the operations required for its manufacture. Another disadvantage is that no representation of faces is available, except in the primitive solid model, and so a design representation that has to deal with manufacturing restrictions, based on the Constructive Solid Modelling technique alone, is useless. It has to be converted to a Boundary Representation.

3.3. Manufacturing-Oriented Design

Neither of the design representations discussed above is suitable for our purposes by itself. When modified and combined, however, a suitable representation can be contrived. The Manufacturing-Oriented Design representation proposed here is such a modelling technique.

Manufacturing-Oriented Design is a manufacturing oriented approach to geometrical design that is based on solid modelling representations. It is a combination of a modified Constructive Solid Modelling representation and a Boundary representation. The latter representation is updated after each operation, and each operation performed is recorded on the Constructive Solid Modelling Tree.

One of the modifications to the Constructive Solid Modelling representation is that each design transformation, each node of the tree, requires a manufacturable counterpart. (See Appendix IV.) For instance, the subtract transformation has a material removal technique as its manufacturable counterpart. This can be milling, electrospark erosion, or another suitable technique. Furthermore, the addition transformation must be split into two distinct operations: combine and merge. The combine transformation, for instance, has welding or gluing as its manufacturable counterpart. The merge transformation generates user defined primitive solids that may be manufactured by casting, for instance. These design transformations are called Manufacturable Transformations.

The combination of a Boundary representation and a Constructive Solid Modelling representation, even though it is extended with the inclusion of Manufacturable Transformations, is still not sufficient for our purposes, since it is possible to generate a design that cannot be fabricated.

Figure 3.11, for instance, shows a cylinder that is subtracted from a cube in a way that cannot be manufactured. The material cannot be removed.

In order to be able to handle manufacturing restrictions, Manufacturing-Oriented Design must be supplemented with the concepts of Manufacturable Objects, Manufacturing Machine Models, and Implicit Locating.

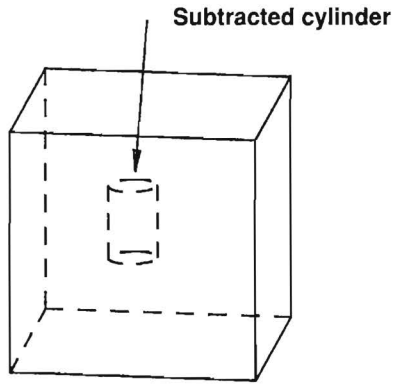


Figure 3.11. A subtraction that cannot be manufactured

As has been explained, a Manufacturable Object is a geometrical form which, it has been demonstrated, can be fabricated. The concept of a Manufacturable Object is the design and manufacturing process planning counterpart to the application of a combination of one or more tools, machines and setups (see Appendix IV). The Manufacturable Object concept consists of a geometrical form together with its application rules.

A Manufacturing Machine Model is a model of an available manufacturing machine that 'knows' the accuracy that can be achieved by the machine, as well as which tools and workpiece setups can be used to fabricate (a part of) each available Manufacturable Object or Manufacturable Object transformation according to the specifications. The Manufacturing Machine Model concept handles the part of the manufacturing restrictions that depend on the equipment used.

We have not yet dealt with two kinds of manufacturing restrictions: tolerances and fits. In order to deal with the design counterparts of the limitations on manufacturing accuracy the concept of Implicit Location is introduced. An Implicit Location specifies the location (position and orientation) of a solid object or a Manufacturable Object with constraints relative to another solid object. Examples of these constraints are faces that have to meet, edges and

vertices that have to coincide, or a peg that has to be inserted in a hole. The main advantage of the Implicit Location concept lies in the fact that tolerances and fits can be dealt with functionally, as they are in the product's manufacturing phase, since the constrained design objects and the relevant constraints are recorded on the Constructive Solid Modelling tree. This allows identification of the manufactured referential objects in correspondence with the designed referential objects during the course of fabrication, as well as the location of the physical solid or the Manufacturable Object, as specified by the restraints.

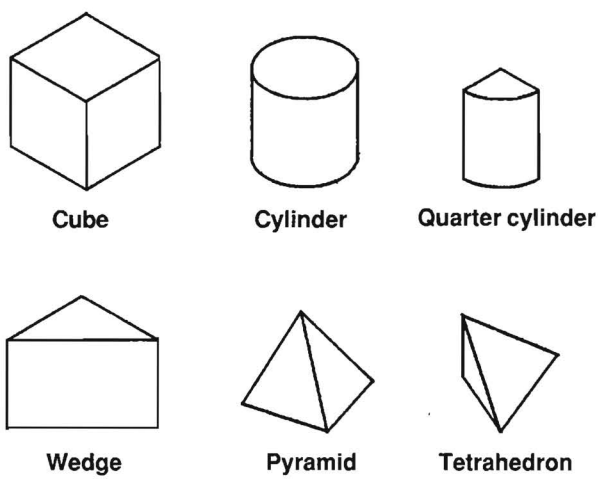


Figure 3.12. The available primitive solids

The two geometrical design representations of a Manufacturing-Oriented Design also need primitive solids. The definition of a new primitive solid can be tedious and a few primitives have therefore been predefined.

These are a cube, a wedge, a tetrahedron, a quarter cylinder, a cylinder, and a pyramid, all of which are illustrated in Figure 3.12.

In addition, a number of operations on solids, such as 'locate', 'merge', 'combine', 'subtract', and 'apply Manufacturable Object' have been implemented in order to generate new solids, as is shown in Appendix IV. For convenience,

primitives such as a cylinder have also been implemented, even though they can be generated from other primitives.

A typical example of the use of Manufacturing-Oriented design is presented in Chapter 5.

3.4. Summary

The design representation developed here differs from those with which most designers are familiar, in that it takes account of manufacturing restrictions. Furthermore, it adopts a different viewpoint towards the integration of design and manufacturing.

Of the six main categories of manufacturing process – primary shaping, forming processes, material removal, joining, coating, and material feature changing – three can be dealt with directly: primary shaping (the merge transformation), material removal (through the Manufacturable Object concept and the subtract transformation), and joining (through the combine transformation). The last two categories can be incorporated with only relatively minor adaptations of the concept.

Forming processes may be tackled with the aid of the Manufacturable Object concept, but they are not so well catered for as the other categories, since the outcome of the manufacturing phase depends much more on such features of the material as its elasticity, texture, grain size, stress state, and effective deformation. Further research is needed in order to include the forming process.

The most important concept of Manufacturing-Oriented Design is the Manufacturable Object concept which represents an attempt to formalize the knowledge possessed by a skilled worker. Furthermore, it can be a powerful instrument in the handling of manufacturing restrictions in the design phase. Unfortunately, the definition of new Manufacturable Objects, and their application rules in particular, is a tedious task. These rules formalize knowledge about what shapes, in which application contexts, can be manufactured. Currently, the number of Manufacturable Objects available and in particular the number of application rules for these Manufacturable Objects is too small. Further research is required before it can be guaranteed that a given design can in fact be fabricated.

Another important concept when handling manufacturing restrictions in the geometrical design phase is the Manufacturing Machine Model concept. This concept introduces the available machinery and equipment, in particular their limitations, into the design phase.

Finally, the Implicit Location concept pairs the functional relations between different features of the design, such as faces and edges, within the limitations of the accuracy that can be attained by the manufacturing process.

Chapter 4

Manufacturing Process Planning

The function of manufacturing process planning is the generation and validation of a manufacturing process plan for a design, the objective being the selection and detailed definition of the manufacturing processes that have to be performed in order to transform raw material into a prescribed shape. Primarily this means defining technically and economically feasible manufacturing processes. The available resources, such as manufacturing machines, tools and labour, serve as constraints.

The production of a manufacturing process plan includes the selection of feasible manufacturing processes, the selection of machines and tools sets, the generation and selection of setups, the selection of tools, the design and selection of jigs and fixtures and, finally, the calculation of manufacturing conditions and the generation of tool paths.

The selection of technically and economically feasible manufacturing processes depends on the product design, the machines selected, the tools and setups chosen, the batch size, and the necessary jigs and fixtures. The purpose of this phase is the selection of those manufacturing processes that can realize the product specification at an acceptable cost. The determination of the sequence of manufacturing processes depends on the design of the product, the machine chosen, the tool set selected, and the necessary manufacturing operations. Its purpose is twofold. First, tolerances and precisions may require a specific fabrication sequence and, second, the nonproductive costs of loading/unloading and setting-up have to be minimized.

The selection of machines and tool sets depend on the design of the product, the characteristics of the machines and tools that are available, the cost of the manufacturing operation with each machine and each appropriate tool set, and the batch size. The major criteria are cost, availability of the necessary machines and tools and, sometimes, quality.

The generation and selection of setups depend on the design of the product, the specifications of the machine chosen, and the specifications of the selected tool set. This phase comes into play since a given machine may not be able to perform a specified manufacturing operation with a specific setup.

The selection of tools depends on the product design, the machine selected, the setups chosen, the tool set selected, the manufacturing operations chosen, the sequence of manufacturing operations, and the selected jigs and fixtures.

The design and selection of jigs and fixtures takes account of the product design, the machine chosen, the manufacturing operations, the available jigs and fixtures, the selected tools and the selected manufacturing conditions. The outputs of this phase are not only the selected jigs and fixtures, but also the product position and orientation in relation to the manufacturing machine.

The determination of the manufacturing conditions and the tool paths requires details of the product design, the specifications of the selected manufacturing machine, the specifications of the selected tools, the specifications of the selected jigs and fixtures, and the position and orientation of the product relative to the selected manufacturing machine. The outputs of this phase are the manufacturing conditions, the estimated tool wear, the estimated manufacturing times, the estimated manufacturing costs and the tool paths. These outputs are required for the generation of a manufacturing program for each machine (Erve 1988).

4.1. Types of Manufacturing Process Planning

Two basic types of manufacturing process planning may be distinguished: the retrieval type and the generative type.

Retrieval planning is based on group technology methods, by which manufacturing parts are coded and classified into family groups. Each family comprises parts that have a sufficient number of parts in common that they can be fabricated by a common method.

The generative type of planning generates a new manufacturing process plan for every part ab initio. This type of planning uses mathematical models for the description of the selection process (Weill, Spur, Eversheim 1982).

The generative type of manufacturing process planning is far more powerful than the retrieval type, since the latter is limited by the number of predefined groups. Unfortunately, generative planning requires human inventiveness, and this cannot yet be formalized, which means that human intervention will necessarily be associated with generative planning, at least for the foreseeable future. Human skill will also be needed for another reason: the combinatorial explosion. Even if only a few alternative machines, tools, setups and jigs are available, the number of possible manufacturing process plans soon becomes extremely large. If each possible plan had to be evaluated every time an article went into production, there is no conceivable way that the item would be manufactured within an acceptable time frame, if at all. The only way to filter this wealth of information is to use the skilled worker's experience as a guide through the maze in the selection of the most economically viable manufacturing process plan.

4.2. The Approach Chosen

As the reader may by now have gathered, manufacturing process planning is not a particularly straightforward activity. All the activities in the process are to some extent interdependent. The concepts of Manufacturing-Oriented Design introduced in the previous chapter do facilitate the preparation of a manufacturing process plan, in that five of the concepts incorporated in Manufacturing-Oriented Design – the Constructive Solid Modelling tree of a geometrical design, the Manufacturable Objects, the Manufacturable Transformations, Implicit Location, and the Manufacturing Machine Models – are relevant to the preparation of a manufacturing process plan.

The Constructive Solid Modelling tree of a design includes the specification of the raw materials needed and the transformations applied in the design phase. Each design transformation can be a Manufacturable Transformation, an Implicit Location, or the application of a Manufacturable Object.

As has been mentioned above, each manufacturing machine available has a model, the Manufacturing Machine Model. The model of a manufacturing machine knows if the corresponding manufacturing machine is capable of executing a Manufacturable Transformation, an Implicit Location, or the application of a Manufacturable Object. If the machine is capable of executing them, then the Manufacturing Machine Model knows, or is able to calculate, which combinations of setups, tools and fixtures is suitable for their execution. Furthermore, the Model is able to determine the manufacturing conditions and to generate the necessary tool paths. It also allows the simulation of the machine and the product while generating the tool paths.

The Constructive Solid Modelling tree can thus be used as a guide for the generation of a manufacturing process plan. It can be regarded as a high level outline of the manufacturing process plan. The Constructive Solid Modelling tree, which is also called the history of a design, is passed on to the ManufacturingProcessPlanner processor (see Chapter 2) as the contents of the interaction generateManufacturingPlan.

Optimization may require a change in the sequence in which the design transformations are applied which may, in turn, lead to an intermediate state that cannot be manufactured. Since our objective is to take explicit account of manufacturing limitations, it is good practice not to change the order in which transformations are executed, unless one is perfectly certain that the changed sequence can in fact be executed and will produce the required product.

It is usual to generate a selection of alternative manufacturing process plans, since the planning process is time consuming and the selected machines, tools, jigs and fixtures may not be available when required. They may, for instance, be undergoing repair. The approach chosen allows the generation of alternative manufacturing plans by the selection of other machines, tools, setups, jigs and fixtures.

4.3. Summary

The concepts selected for incorporation in Manufacturing-Oriented Design categorize the manufacturing process planning phase as a powerful type of retrieval manufacturing process planning operation.

Although it is possible to select machines, tools, setups, jigs and fixtures automatically, given the design's Constructive Solid Modelling tree and the models of the available manufacturing machines, the necessary equipment is selected by a human planner. The interdependence of the activities associated with manufacturing process planning, coupled with the potential non-availability of a certain item of equipment due to their being broken or defective, leads one to the conclusion that the fully automatic generation of a manufacturing process plan is overly complex and, furthermore, undesirable, since skilled planners can use their experience to select suitable combinations of machines, tools, setups, etc., with relative ease. The user is therefore requested to select the desired machines and tools. The validity of the choices is then tested against the model of the machine selected. The Manufacturing Machine Model is asked whether the corresponding machine is capable of executing the given Implicit Location, the Manufacturable Transformation or the application of a Manufacturable Object with the chosen tools. The model is able to deduce this information on the basis of its knowledge of the design transformations, or it can determine it by calculating the tool paths needed; for example by comparing them with the reach of the selected machine.

The output of the manufacturing process plan serves as input for a number of down-line functions: capacity planning, requirements planning, machine load balancing, master production scheduling, and routine planning. A discussion of the interface between the manufacturing process plan and these further functions is beyond the scope of this study.

The fabrication of a Manufacturable Object requires tools, but which tools are required depends on the context within which the work is executed, the place, and the application environment.

This is illustrated in Figure 4.1, in which the Manufacturable Object to be fabricated is a groove in an axle.

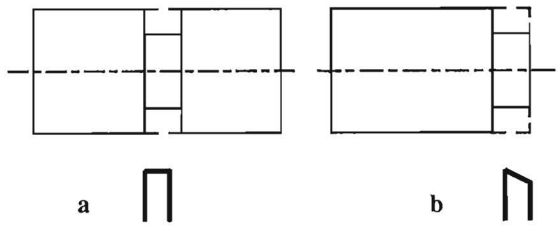


Figure 4.1. The context dependence of Manufacturable Objects

This can be manufactured using a lathe and a grooving tool (Figure 4.1a). The same Manufacturable Object but applied at a different location on the axle (Figure 4.1b), can be produced using a lathe and an end facer. The facer cannot produce the Manufacturable Object shown in Figure 4.1a. For each Manufacturable Object one thus needs to know which tool to select for the given context of application.

In order to test the concepts included in the Manufacturing-Oriented Design, a limited number of machines, setups, tools, jigs and fixtures have been modelled. Two machines, six categories of tools, a welding manipulator, two setups, and two fixtures and jigs have been modelled. Software tools are available to permit the modelling of new items of equipment, and so an extension of the capabilities of the manufacturing process planning phase is relatively simple.

Of course, the currently limited number of machines, tools, setups, jigs and fixtures available as models does represent a disadvantage in that it may be possible that no available combination is capable of executing a Manufacturable transformation or of fabricating a Manufacturable Object. The number of combinations available as models will have to be enlarged.

Chapter 5

A Typical Example

This chapter presents a case study that serves as an example of the concepts introduced in the foregoing chapters. The work is based on software written in Smalltalk-80, which is an object-oriented computer programming language (Goldberg 1984, Goldberg, Robson 1985).

For the purposes of our example, we consider the design of a product the manufacture of which requires two material removal operations. The design of a manufacturing process plan forms part of the example. The example product is depicted in Figure 5.1.

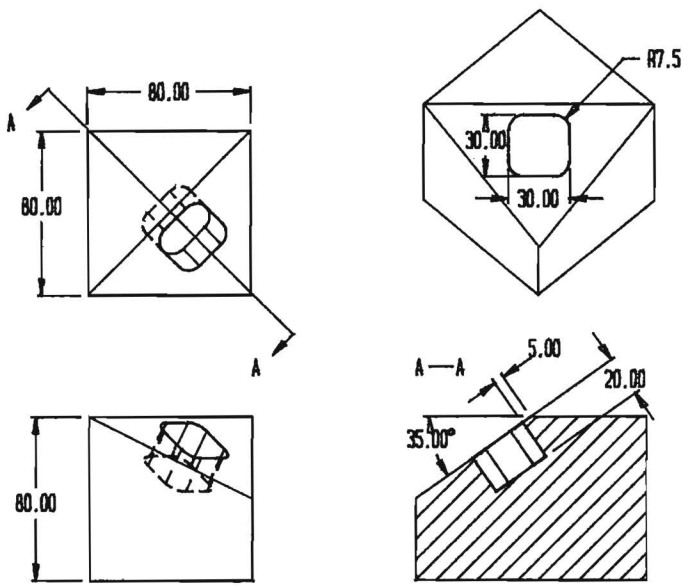


Figure 5.1. The product to be designed

The manufacturing process plan, once generated, is transformed into a manufacturing machine program which has been executed using a Maho 700S five-axis milling machine.

The geometrical design phase starts with an evaluation of the Smalltalk-80 expression 'ProductDesignInterface open'. The view that functions as a user interface then opens up on the computer screen.

The design of the product commenced with the copying of a primitive cube, followed by resizing it to a side length of 80 mm. Before a copy of a primitive is made, the material of which it is to be fabricated, the surface roughness and, if appropriate, the fit of the solid are requested. The material to be used determines the conditions of the manufacturing process that will be specified in the manufacturing process planning phase. In the design phase the desired accuracy is required to check whether a supplier can offer the accuracy desired, or whether the machines available can produce the object from a basic piece of material. The accuracy of an available machine may be checked by reference to the Manufacturing Machine Model, which is an implementation of one facet of the manufacturingRestrictions interaction mentioned in Chapters 2 and 3.

The next step in the design phase was the application of a Manufacturable Object which removes all the material above a plane surface. The Manufacturable Object must first itself be made, however. This is done by copying a primitive Manufacturable Object together with the relevant surface roughness and, where appropriate, the fit. These values are combined with the type of the Manufacturable Object and compared with the values attainable with the available manufacturing machines by consultation of the relevant Manufacturing Machine Models.

In the program as it is currently implemented there are four primitive Manufacturable Objects available: a cylindrical hole, a planar material removal, a rounded cavity, and a cylinder. The first three are material removal operations and the last one is an assembly primitive.

Figure 5.2 shows the Manufacturable Objects required for this example. As can be seen, we need a copy of the planar material removal primitive and a scaled copy of the rounded cavity Manufacturable Object, in addition to the cube mentioned above.

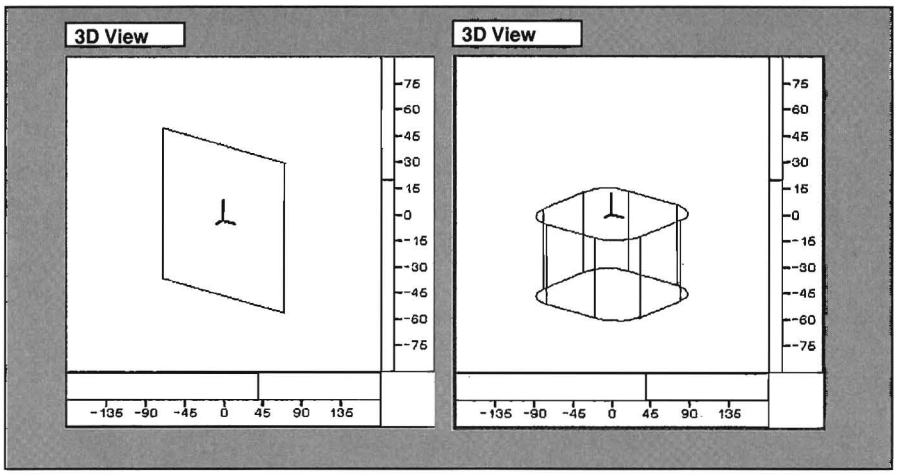


Figure 5.2. The Manufacturable Objects needed

Figure 5.2a shows the planar material removal Manufacturable Object. Figure 5.2b shows the rounded cavity. This has a width of 30 mm, a height of 30 mm, a depth of 20 mm, and a corner radius of 7.5 mm. The Manufacturable Objects generated are added to the Manufacturable Objects list of the design user interface.

Before the material removal Manufacturable Object can be used, it has to be located (positioned and orientated) which operation can be performed in two ways: explicitly or implicitly. Explicit location means that the coordinates of the reference frame – the central axis system of the design object – are explicitly specified. The concept of Implicit Location, as has been explained in Chapter 3, means that the location, the position and the orientation of the object are specified by the constraints. The Implicit Location concept is an aspect of the manufacturingRestrictions interaction introduced in Chapter 2. The outcome of the location of the planar material removal Manufacturable Object is shown in Figure 5.3.

The next step is the application of the planar material removal Manufacturable Object, the outcome of which is revealed in Figure 5.4.

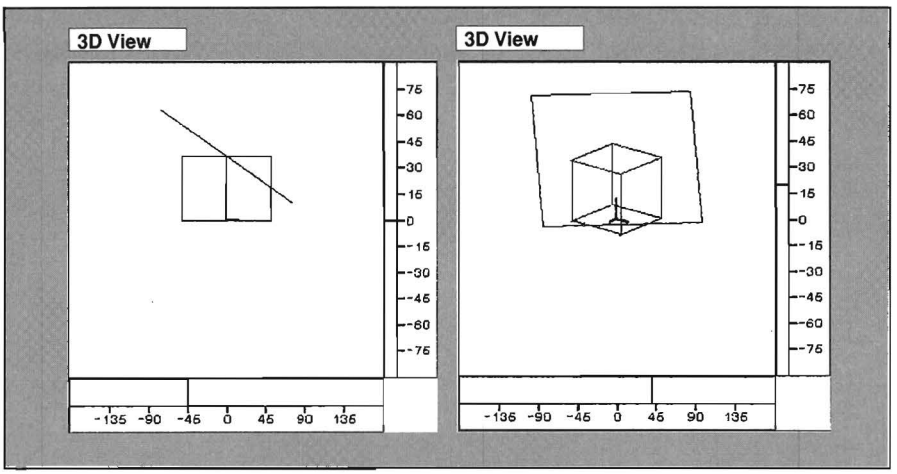


Figure 5.3. The outcome of the Implicit Location

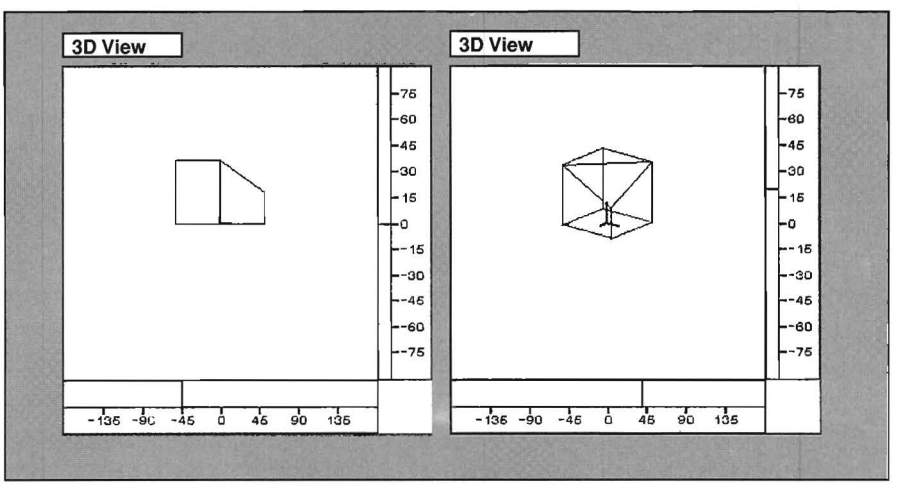


Figure 5.4. The result of the application

The material above the plane surface has been removed. Before a Manufacturable Object can be applied its validity has to be tested using the relevant application rules.

We now have to locate the Manufacturable Object that corresponds to the rounded cavity, and then we have to apply it. The location and the application of this Manufacturable Object are similar to the previous operations, except that a different Implicit Location method has been used. The result is shown in Figure 5.5.

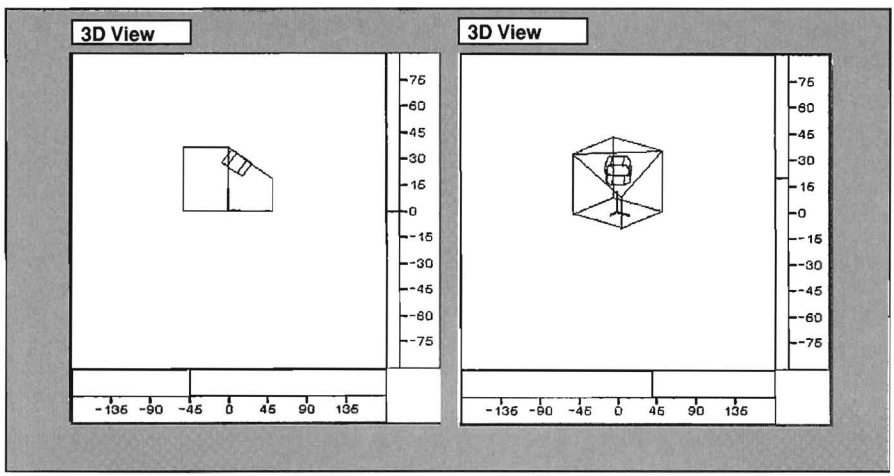


Figure 5.5. The final design result

The design of the product is now complete. Its design history, the Constructive Solid Modelling tree, is shown in Figure 5.6.

Figure 5.6 shows how the product has been designed. The top left-hand list is the history, the Constructive Solid Modelling tree of the design. The item selected in this list is the first 'ApplyManufacturableObject' operation. The type of Manufacturable Object applied, a PlanarMaterialRemoval, and the planar face equation, are shown in the bottom right-hand view.

The process of creating a manufacturing process plan for this product is performed by passing on the history of the design to the manufacturing process planning phase. This is a task for the generateManufacturingPlan interaction discussed in Chapter 2. Before doing this, however, a model of the work cell to

be used has to be configured. This is necessary for the simulation and the validation of the manufacturing process plan that has been generated.

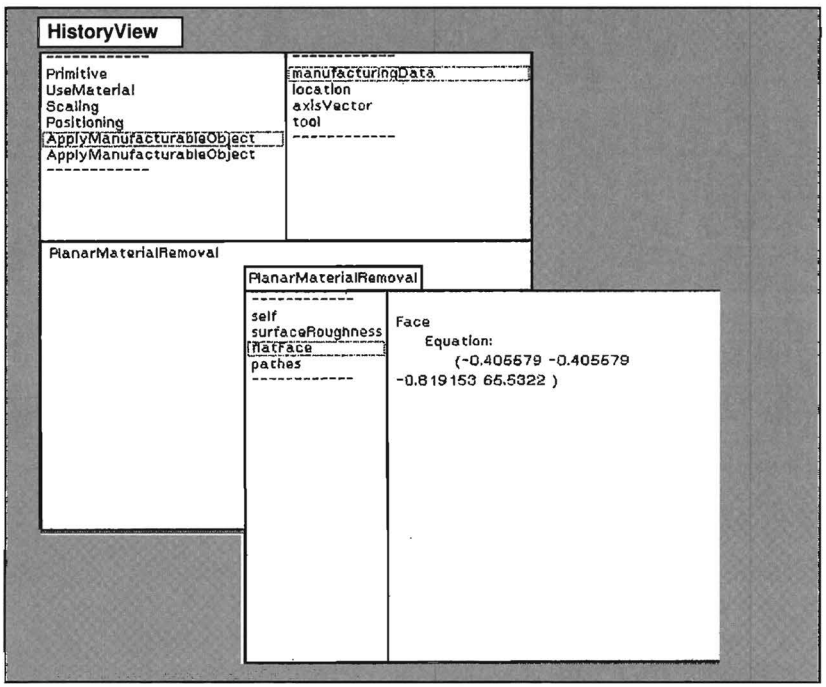


Figure 5.6. The history of the designed product

After configuration of the work cell with the relevant machine(s) – in this case only a Maho 700S milling machine – the user interface appears as shown in Figure 5.7.

The central list shown in Figure 5.7 shows the machines in use. The tools currently in use are shown in the right-hand list. The design history of the product – the Constructive Solid Modelling tree – can be found in the top left-hand list of the user interface. This last list is rather special, in that the user may only pass down the list in a stepwise fashion, since the history of the design is used as a guide for the manufacturing planning phase. The item selected is

therefore the last item in the list which, in Figure 5.7 is shown as Positioning, has no manufacturable counterpart. The earlier items in the Constructive Solid Modelling tree, up to the Positioning item, represent a description of the initial, raw shape of the product to be manufactured. The Positioning item may be ignored for this particular machine as a consequence of the setup chosen. The next step, therefore, is the application of the planar material removal Manufacturable Object by selecting 'step' in the menu of the history list. Before it is applied, the Manufacturing Machine Model (of the machine selected) is interrogated to determine whether the machine is capable of executing the operation chosen. If it cannot, then the user is requested to select another machine from a list of suitable machine types that is provided automatically.

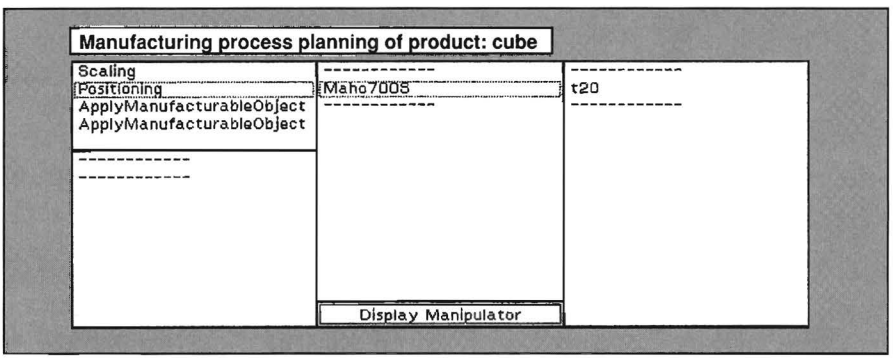


Figure 5.7. The manufacturing process planning user interface

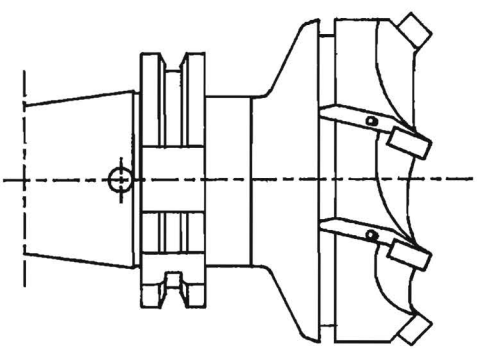


Figure 5.8. A face shell end mill

The Manufacturing Machine Model of the Maho 700S milling machine knows that the planar material removal operation can be executed by using the face shell end mill shown in Figure 5.8.

This particular mill is not at present inserted in the machine's tool holder, so a change of tool is needed. A list of suitable tools is supplied automatically to the user. After a suitable tool has been selected, the tool paths needed to execute the planar material removal operation are generated and simulated, as shown in Figure 5.9, which shows a simulation of the manufacturing machine and the simulated product.

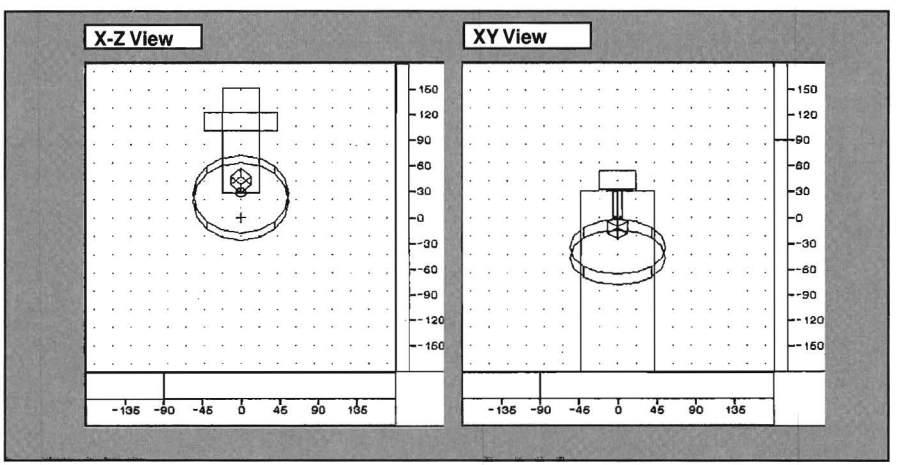


Figure 5.9. The simulation of the generated toolpaths

The final step in this exercise is the application of the Manufacturable Object to generate the rounded cavity. This is executed on the Maho 700S by means of a slot mill, shown in Figure 5.10, which means that another tool change must be performed.

Before the tools are fitted to the selected machine they are checked for suitability using the design transformation as reference. For example, the tool type, radius and free height are checked.

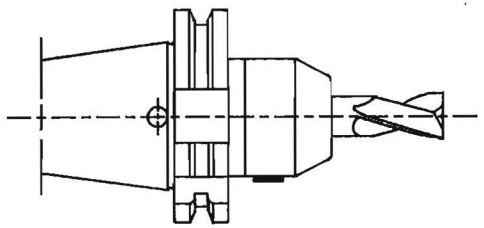


Figure 5.10. A slot mill

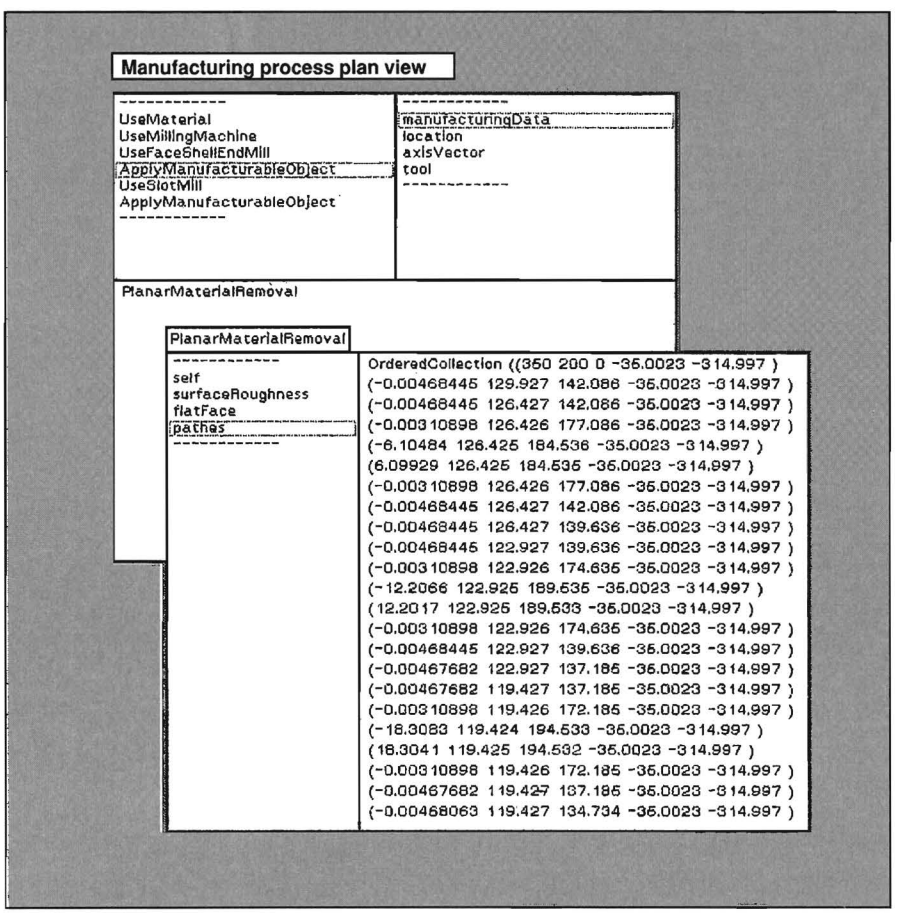


Figure 5.11. The result of the manufacturing process planning phase

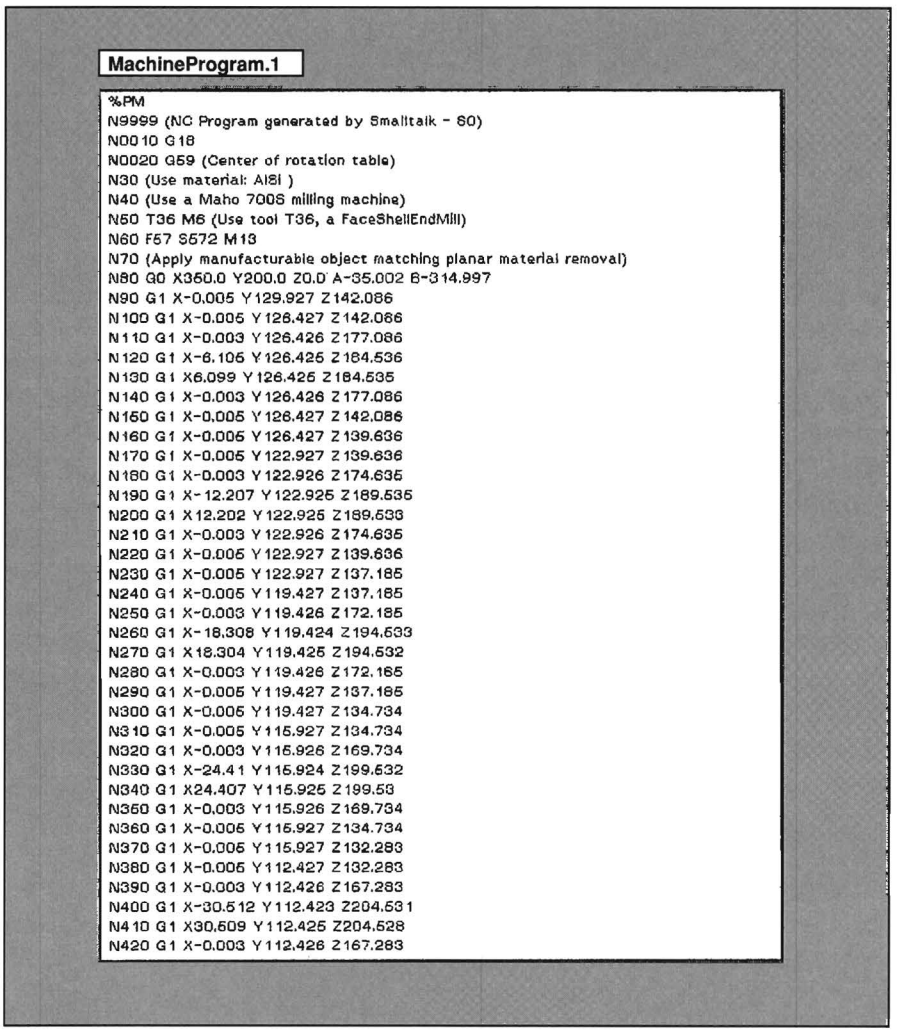


Figure 5.12. The generated machine program

The final result of the manufacturing process planning phase is shown in Figure 5.11, which shows the manufacturing process plan that has been generated as a result of the foregoing steps.

The top left-hand list of Figure 5.11 shows the manufacturing operations that have to be executed.

The bottom right-hand list shows the paths the milling machine has to follow in order to execute the planar material removal Manufacturable Object. The elements of the 'OrderedCollection' of paths are axis vectors that specify an end position for the milling machine. The manufacturing process plan thus generated can be transformed into one or more machine programs. The generated machine program is shown in Figure 5.12.

The automatically generated machine program was used to control the Maho 700S milling machine, with the result shown in Figure 5.13.

The fabricated product was measured and was found to comply with the design specifications.

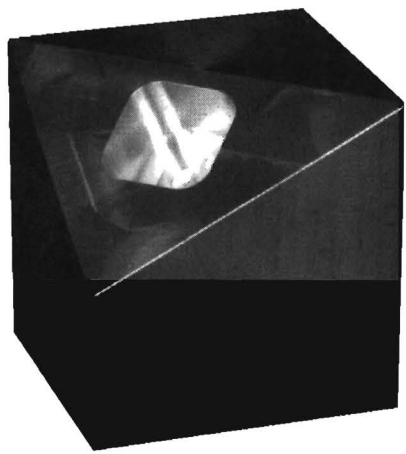


Figure 5.13. The manufactured product

5.1. Remarks

The foregoing discussion merits some further remarks.

Every concept introduced in this study has been demonstrated with the exception of the Manufacturable Transformation concept. An example of a Manufacturable Transformation is the welding together of two parts by a robot.

It may be that, during the development of a manufacturing process plan, the implementation or execution of a number of Manufacturable Objects will cause a degree of mutual interference. Under such circumstances it would probably be possible to devise a more economical manner to execute the Manufacturable Objects. Minimization of the cost of production depends rather on the batch size or production run contemplated, but no attention has been paid to optimization problems of this type in this study.

The actual workshop situation on which this study was implemented used one fixed set up: one machine and a clamp. This resulted in a rather restricted manufacturing process plan and, clearly, limited the type of products that could be manufactured.

Finally, the design of the product used in this typical example is rather simple. It is easy to design and fabricate more complex products, however, since the capabilities of Manufacturing-Oriented Design is restricted only by the number of Manufacturable Objects and Manufacturing Machine Models available.

5.2. Summary and Conclusions

The software developed during the course of this study, which is based on the concepts discussed, has been shown to achieve the desired integration of design and manufacturing. The package as it stands at present, however, is not suitable for industrial use, since it contains only a few Manufacturable Object primitives and Implicit Location methods, and is restricted by the incorporation of models of only a few machines, tools, setups, jigs and fixtures. The last two categories can be extended with relative ease, however, but the extension of the first category – the number of Manufacturable Object primitives – is a fairly complex operation which will be dealt with more fully in the following chapter.

A number of areas have not yet been sufficiently explored to allow their inclusion in the software package, one among which is the automatic generation of fixtures and setups. Markus (1987) has proposed strategies for the generation of modular fixtures on the basis of known setups. Once again, the generation of setups is a task that can be accomplished with relative ease by a skilled worker, but one that is difficult to formalize. The design representation chosen may well considerably ease the automatic generation of setups, but further work remains to be done in this area.

The power of the concepts developed allow other functions to be included within the package: structural analysis, for example, or collision-free path planning, or 'Design for Assembly' (Boothroyd 1982), as well as such functions as requirement planning, scheduling, route planning, capacity planning, expediting and inventory control. Collision-free path planning is particularly important when generating manufacturing plans for robots. Lozano-Perez (1981) has carried out extensive research in this area. As a step towards the full implementation of collision-free path planning within the current software, a collision detection routine has beenincorporated, which signals the occurrence of a collision, if one should occur.

In combination with the Process Interaction Approach, the concepts presented here could be extended to the control of more than single machine tool units: they may even be extended as far as the control of a complete factory, with the

manufacturing process plan in that case serving as a prescription for the desired actions for the factory controller.

Although far from being a complete package, the software in its present state of development has demonstrated that the concepts developed in the previous chapters are suitable for the integration of geometrical design and manufacturing.

Chapter 6

Results and Conclusions

The foregoing chapters have introduced certain concepts for the integration of the design and manufacture of mechanical products. The experience gained thus far with these concepts has shown that they are capable of achieving the desired integration. Their power springs from the enforcement of manufacturing restrictions at the geometrical design stage, which ensures a guarantee that a design generated according to the method described will be capable of being fabricated. Only relatively few concepts are necessary in order to superimpose manufacturing restrictions on the geometrical design process. These are: Manufacturable Transformations, Manufacturable Objects, Manufacturing Machine Models, Implicit Location, and two geometrical modelling representations – Boundary Representation and Constructive Solid Modelling.

A Manufacturable Transformation is a design transformation that has a manufacturable counterpart. For example, the 'combine' transformation has welding as its manufacturable counterpart.

A Manufacturable Object is a geometrical shape, together with its application rules. The geometrical form is a form that can in principal be fabricated. The relevant application rules guarantee that the geometrical form can be fabricated within a specific context.

A Manufacturing Machine Model is a model of an available machine. The model knows if the corresponding machine is capable of fabricating a Manufacturable Object or executing a Manufacturable transformation and it also knows how to manufacture such a design transformation. The Manufacturing Machine Model introduces the available machines and associated equipment – in particular their limitations – into the geometrical design process and into the generation of a manufacturing process plan.

Implicit Location is a method whereby the limited accuracy that is attainable by a given manufacturing process may be taken into account. It allows a functional approach to the problem of incorporating manufacturing tolerances and fits into the design phase. The recording of items of reference by the method of Implicit Location allows the incorporation of sensors in the manufacturing process, which means that the values obtained from the sensors can be compared with the required values, so that corrections can be generated if necessary.

The Boundary Representation is introduced since manufacturing processes produce surfaces which have to be incorporated into the geometrical design phase in order to take account of manufacturing restrictions.

In order to reduce human intervention, manufacturing process planning needs a design representation that is based on the initial state of an object, together with a list of the state changes through which the object passes on its way to becoming a finished product. Constructive Solid Modelling has thus been introduced into the geometrical design phase to take account of this. A consequence of this introduction is that only a number of Manufacturable Objects having simple geometrical forms need to be made available, since these can be combined into more complex geometrical forms. The application rules associated with the geometrical forms ensure that the design can in fact be fabricated.

The Manufacturable Object concept is particularly important in the enforcement of manufacturing restrictions on the geometrical design, since Manufacturable Objects represent a formalization of what can actually be fabricated. The definition of new Manufacturable Objects, in particular their application rules, remains, unfortunately, a tedious task, since the generation of application rules formalizes the context- dependent knowledge of what can actually be fabricated. The rules are not only context dependent, however, they also depend on the equipment available. For example, the creation of a hole in a block requires a certain amount of free space around the block in the vicinity of the hole to be made, in order that the available equipment can reach the site to begin its work. This dependence on the actual equipment available to do a particular job is handled by the Manufacturing Machine Model. The context-dependency of the concept is handled by the application rules, which can lead to a great number of rules for each object due to the large number of contexts in

which it can be fabricated. This clearly reveals the need for more research at a fundamental level in order to formalize what can be fabricated in each given application context. It is suggested that mathematical rather than empirical methods might prove more powerful in this endeavour.

Six fundamental fabrication techniques have been considered: primary shaping, forming processes, material removal, joining, coating, and material feature changing (such as hardening or annealing). Of these, three can be dealt with directly on the basis of the concepts introduced here: primary shaping, material removal, and joining. Coating and material feature changing can readily be incorporated. The forming process might be tackled with the help of the Manufacturable Object concept, but it is not so well suited to the concept as the processes that have been incorporated.

The processes might also be tackled at the manufacturing process planning stage, when this generates corrective actions. This occurs in the manufacturing phase when the form as fabricated does not match the form desired. A series of 'what to do when ... ' rules has to be generated and included in the Manufacturable Object concept. This remains a matter for further research.

Designers may think that the concepts developed and presented here may limit their freedom by the imposition of manufacturing restrictions on the design phase. But such restrictions have to be dealt with, either in the geometrical design phase or in the manufacturing process planning phase. The separation of the design function from the manufacturing function has lead, over the centuries since the time of Leonardo da Vinci, to the neglect of manufacturing restrictions during the course of the development of a design.

In this regard, designers have acquired more freedom, but the consequence has often been an increase in the number of problems that have to be solved at the manufacturing process planning phase. The introduction of manufacturing restrictions at as early a stage as possible – i.e. at the geometrical design phase – does lead to a desirable integration of the design and manufacturing processes.

The main advantages of the concepts introduced here are that they guarantee, at the geometrical design stage, that a design, once produced, can actually be manufactured. They also offer the rapid generation of manufacturing process

plans, and they thus substantially reduce the time taken to traverse the path between design and manufacture.

References

Arbab F.,
Requirements and architecture of a CAM-oriented CAD system for design and manufacture of mechanical products.
Ph.D. Thesis, University of California,
University Microfilms International, Ann Arbor, MI, 1982.

Asimow A.,
Introduction to Design.
Prentice-Hall Inc., Englewoods Cliffs, NJ, 1962.

Bertalanffy von L.,
General System Theory. Foundations, Development, Applications.
George Braziller, New York, NJ, 1968.

Blume C., Jakob W.,
Programmiersprachen fuer Industrieroboter (in German).
Vogel-Buchverlag, Wurzburg, 1983.

Booker P.J.,
A history of engineering drawing.
Chatto & Windus, London, 1963.

Boothroyd C.,
Design for assembly handbook.
University of Massachusetts, 1982.

Braid I.C.,
Designing with Volumes.
Cantab Press, Cambridge, England, 1974.

Campman A.M., Jenniskens H.J.G.,
Standaard kwaliteit van metalen onderdelen voor professionele apparatuur (in Dutch).
N.V. Philips gloeilampen fabrieken, Eindhoven, 1987.

Erve van 't A.H.,
Generative Computer Aided Process Planning for Part Manufacturing,
An Expert System Approach.
Ph.D. Thesis, Fac. of Mechanical Engineering,
Twente University of Technology, Enschede, 1988.

Finkel R., Taylor R. Bolles R., Paul R., Feldman J.,
Al, A Programming System for Automation.
Stanford Artificial Intelligence Laboratory.
Memo AIM-243, Stan-CS-74-456, Palo Alto, CA, 1974.

Goldberg A.,
Smalltalk-80: The Interactive Programming Environment.
Addison-Wesley Publishing Company, Reading, MA, 1984.

Goldberg A., Robson D.,
Smalltalk-80: The Language and its Implementation.
Addison-Wesley Publishing Company, Reading MA, 1985.

Haterd van de A.W.J.M.,
Prestaties in industriele systemen, een kwestie van besturen (in Dutch).
Master's Thesis, Fac. of Mechanical Engineering,
Eindhoven University of Technology, Eindhoven, 1988.

Klooster W. v.d., Boot W.C.,
Private Communications, Eindhoven, 1985.

Liebermann L.I., Wesley M.A.,
AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly.
In: IBM Journal of Research and Development pp.321-333, (1977).

Lozano-Perez T., Automatic Planning of Manipulator Transfer Movements.
In: IEEE Transactions on Systems, Man and Cybernetics
vol: 11 number: 10 pp.681-698 (1981).

Lozano-Perez T., Robot Programming.
Massachusetts Institute of Technology,
A.I. Memo No. 698, Cambridge, MA, 1982.

Markus A.,
Strategies for the automatic generation of modular fixtures.
Computer and Automation Institute, Hungarian Academy of Sciences, 1987.

Mommertz K.H.,
Bohren, Drehen und Frasen, Geschichte der Werkzeugmaschinen.
Rowohlt Taschenbuch Verlag Gmbh, Reinbeck, 1981.

Newman W.M., Sproull R.F.,
Principles of Interactive Computer Graphics.
McGraw-Hill Book Co., Tokyo, 1981.

Overwater R.,
Processes and Interactions. An Approach to the Modelling of Industrial Systems. Ph.D. Thesis, Fac. of Mechanical Engineering,
Eindhoven University of Technology, Eindhoven, 1987.

Palitzsch G.,
Dictionary of Production Engineering.
Verlag W. Girardet, Essen, 1972.

Paul R.P.,
Robot Manipulators, Mathematics, Programming and Control.
The MIT Press, Cambridge, MA, 1984.

Ranky P.G., Ho C.Y.,
Robot Modelling, Control and Applications with Software.
IFS Publications Ltd. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

Requicha A.A.G.,
Mathematical Models of Rigid Solid Objects.
Production Automation Project TM-28, University of Rochester, 1977.

Requicha A.A.G.,
Representations for Rigid Solids; Theory, Methods and Systems.
In: Computing Surveys vol: 12 number: 4 pp.437-464 (1980).

Rooda J.E.,
De kunst van het automatiseren (in Dutch).
Inaugural address, Eindhoven University of Technology, Eindhoven, 1987.

Rooda J.E., Boot W.C.,
Procescomputers, Basisbegrippen (in Dutch).
Academic service, The Hague, 1984.

Rooda J.E., Boot W.C.,
Procescomputers, Processen en interacties (in Dutch).
Memorandum nr. WPA 252, Fac. of Mechanical Engineering,
Eindhoven University of Technology, Eindhoven, 1986.

Rooda J.E., Boot W.C.,
Systemen en Modellen (in Dutch).
Memorandum nr. 39, Fac. of Mechanical Engineering,
Twente University of Technology, Enschede, 1983.

Simmons G.F.
Introduction to Topology and Modern Analysis.
McGraw-Hill Book Company, Inc., New York, 1963.

Vankan H., Het aftrekken van volumina (in Dutch).
Memorandum nr. WPA 682, Fac. of Mechanical Engineering,
Eindhoven University of Technology, Eindhoven, 1989.

Weill R., Spur G., Eversheim W.,
Survey of Computer-Aided Process Planning Systems.
In: Annals of the CIRP vol: 32 number: 2 pp.539-542 (1982).

Wiener N.,
Cybernetics or control and communication in the animal and the machine.
Wiley & Sons, New York, NJ, 1955.

Wijk J.J. van,
On new types of solid models and their visualization with ray tracing.
Ph.D. Thesis, Delft University of Technology,
Delft University Press, Delft, 1986.

Winston P.H.,
Artificial Intelligence.
Addison-Wesley Publishing Company, Reading, MA, 1981.

Wortmann A.M., Rooda J.E., Boot W.C.,
Basisbegrippen van de Proces-Interactie Benadering (in Dutch).
Memorandum nr. WPA 0658, Faculty of Mechanical Engineering,
Eindhoven. University of Technology, Eindhoven, 1989.

Appendix I

Smalltalk-80

Smalltalk-80 is a large, graphical, interactive programming environment. In this appendix the basic concepts of the Smalltalk-80 object-oriented programming language Smalltalk-80 will be discussed. The programming environment has been described by Goldberg (1984). This appendix rephrases the definitions given in Goldberg and Robson (1985).

I.1. The Main Concepts of Smalltalk–80

The main concepts of Smalltalk-80 are objects and messages. An object represents a component of the Smalltalk-80 software system. For example, objects represent numbers, strings and views of other objects. An object consists of some private memory and a set of operations. The nature of an object's operations depends on the type of component it represents. For instance objects representing vectors can perform vector operations.

A message is a request for an object to execute one of its operations. A message specifies which operation is desired, but not how that operation should be performed. The receiver, the object to which the message was sent, determines how to carry out the operation requested. For instance, a complex number responds differently to an addition operation than does an integer.

The set of messages to which an object can respond is called its interface with the rest of the system. The only way to interact with an object is through its interface. A crucial property of an object is that its private memory can be manipulated only by its own operations. A crucial property of messages is that they are the only way to invoke an object's operations.

An important part of designing Smalltalk-80 programs is determining which kinds of objects should be described and which message names provide a useful vocabulary of interaction among these objects.

I.2. Classes and Instances

A class describes the implementation of a set of objects that all represent the same kind of system component. The individual objects described by a class are called its instances. A class describes the form of its instances' private memory and it describes how they carry out their operations. For instance, the class Dictionary specifies the implementation of objects representing dictionaries. This class describes how to add elements to and remove elements from a dictionary.

The instances of a class are similar in both their public and private properties. An object's public properties are the messages that make up its interface. All instances of a class have the same message interface, since they represent the same kind of component. An object's private properties are a set of instance variables that make up its private memory and a set of methods that describe how to carry out its operations.

Each class has a name that describes the type of component that its instances represent; for instance, the class Rectangle represents rectangles.

Each instance variable in an object's private memory refers to one object, called its value. Each method in a class explains how to perform the operation requested by a particular type of message. When that type of message is sent to any instance of a class, the method is executed. A class includes a method for each type of operation that its instances can perform. A method may specify some changes to the object's private memory and/or some other messages to be sent. A method also specifies an object that should be returned as the value of the invoking message.

A small subset of the methods in the Smalltalk-80 system are not expressed in the Smalltalk-80 language. They are called primitive methods. For example,

the instances of the class Integer use a primitive method to respond to the message +.

The Smalltalk-80 system includes a set of classes that provides the standard functionality of a programming language and environment such as arithmetic, data structures, control structures and input/output facilities.

I.3. Expression Syntax

An expression is a sequence of characters that describes an object called the value of the expression. There are four types of expressions in the Smalltalk-80 programming language: literals, variable names, message expressions and block expressions.

Literals describe certain constant objects, such as numbers and character strings.

Variable names describe the accessible variables. The value of a variable name is the current value of the variable with that name.

Message expressions describe messages to receivers. The value of a message expression is determined by the method the message invokes. That method is found in the class of the receiver.

Block expressions describe objects representing deferred activities. Blocks are used to implement control structures.

Literals

Five kinds of objects can be referred to by literal expressions: numbers, individual characters, strings of characters, symbols and arrays of other literal constants.

Variables

The memory available to an object is made up of variables. A name of a variable is a simple identifier: a sequence of letters and digits beginning with a letter. Some examples of variables are: index, counter, Number. There are two kinds of variables in the system, distinguished by how widely they are accessible. Instance variables are private. Shared variables can be accessed by more than one object. Private variables are required to have lower case initial letters; shared variables are required to have upper case initial letters.

A literal constant will always refer to the same object, but a variable name may refer to different objects at different times. The object referred to by a variable is changed when an assignment expressions is evaluated. An assignment prefix is composed of the name of the variable whose value will be changed followed by a left arrow (←).

Messages

Messages represent interactions between components of the Smalltalk-80 system. A message requests an operation on the part of the receiver.

Some examples of message expressions and the interactions they represent are:

10 * 23	computes the product of 10 and 23
list add: newComponent	adds the object newComponent to the data structure called list
rectangle center	returns the center of the object named rectangle

A message expression describes a receiver, selector and possibly some arguments. A message's selector is a name for the type of interaction the sender desires with the receiver. The selector of a message determines which of the receiver's operations will be invoked.

Blocks

Blocks are objects that are used frequently in the control structures in the Smalltalk-80 system. A block represents a deferred sequence of actions. A block expression consists of a sequence of expressions separated by periods and delimited by square brackets. For example

```
[counter ← counter + 1.
 array at: index put: 0]
```

When a block expression is encountered, the statements enclosed in the brackets will not be executed immediately. The value of a block expression is an object that can execute the enclosed expressions at a later time when requested to do so. For example, the expression

```
actions
   at: 'initialize'
   put: [array at: 1 put: 100.
         array at: 4 put: 10]
```

does not actually send any "at: put:" messages to the array. It associates a block with the string 'initialize'. The sequence of actions a block describes will take place when the block receives the message value.

Control structures

A control structure determines the order of some activities. An example of a control structure that is implemented with blocks is simple repetition, which is represented by a message to an integer with timesRepeat: as the selector and a block as the argument. The integer will respond by sending the block as many value messages as its own value indicates. For example, the following message doubles the value of the variable named total six times.

```
6 timesRepeat: [total ← 2* total]
```

Two common control structures that are implemented with blocks are conditional selection and conditional repetition. Conditional selection is similar to IF

THEN ELSE statements in Pascal-like languages and conditional repetition is similar to WHILE and UNTIL statements in those languages.

The conditional selection of an activity is provided by a message to a Boolean operator with the selector **ifTrue:ifFalse:** and two blocks as arguments. The only objects that understand this message are the Boolean objects true and false, which are two special literals in the Smalltalk-80 system.

Booleans are returned from messages that ask simple yes-no questions, such as <, =, <=, and >. The following example shows a conditional selection

```
total > 20
    ifTrue: [amount ← 20]
    ifFalse: [amount ← total]
```

The conditional repetition of an activity is provided by a message to a block with the selector whileTrue: and another block as an argument. The receiver block sends itself the message value and, if the response is true, it sends the other block value and then starts over, sending itself value again. When the receiver's response to value becomes false, it stops the repetition and returns from the whileTrue: message. For example, conditional repetition could be used to initialize all of the elements of the array named anArray.

```
counter ← 1.
[counter<= anArray size]
        whileTrue:
            [anArray at: counter put: 100 + counter.
            counter ← counter + 1]
```

Blocks also understand a message with the selector whileFalse: that repeats the execution of the argument block as long as the value of the receiver is false.

I.4. Subclasses

Every object in the Smalltalk-80 system is an instance of a class. A subclass specifies that its instances will be the same as instances of another class, called

its superclass, except for the differences that are explicitly stated. A system class named Object describes the similarity of all objects in the system; so every class will at least be a subclass of Object. A class description specifies how its instances differ from the instances of its superclass. A subclass is a class and can therefore have subclasses. Each class has one superclass, although many classes may share the same superclass, so the classes form a tree structure. A class has a sequence of classes from which it inherits both variables and methods. This sequence begins with its superclass and continues with its superclass's superclass, and so on. The inheritance chain continues until Object is encountered. Object is the only class without a superclass.

Method determination

When a message is sent, the methods in the receiver's class are searched for one with a matching selector. If none is found, the methods in that class's superclass are searched next. The search continues up the superclass chain until a matching method is found. The search for a matching method follows the superclass chain, terminating at class Object. If no matching method is found in any class in the superclass chain, the receiver is sent the message doesNotUnderstand:; the argument is the offending message. There is a method for the selector doesNotUnderstand: in Object that reports the error to the programmer.

Appendix II

The ProductGenerator model

The graphical representation of the ProductGenerator model is shown in figure II.1.

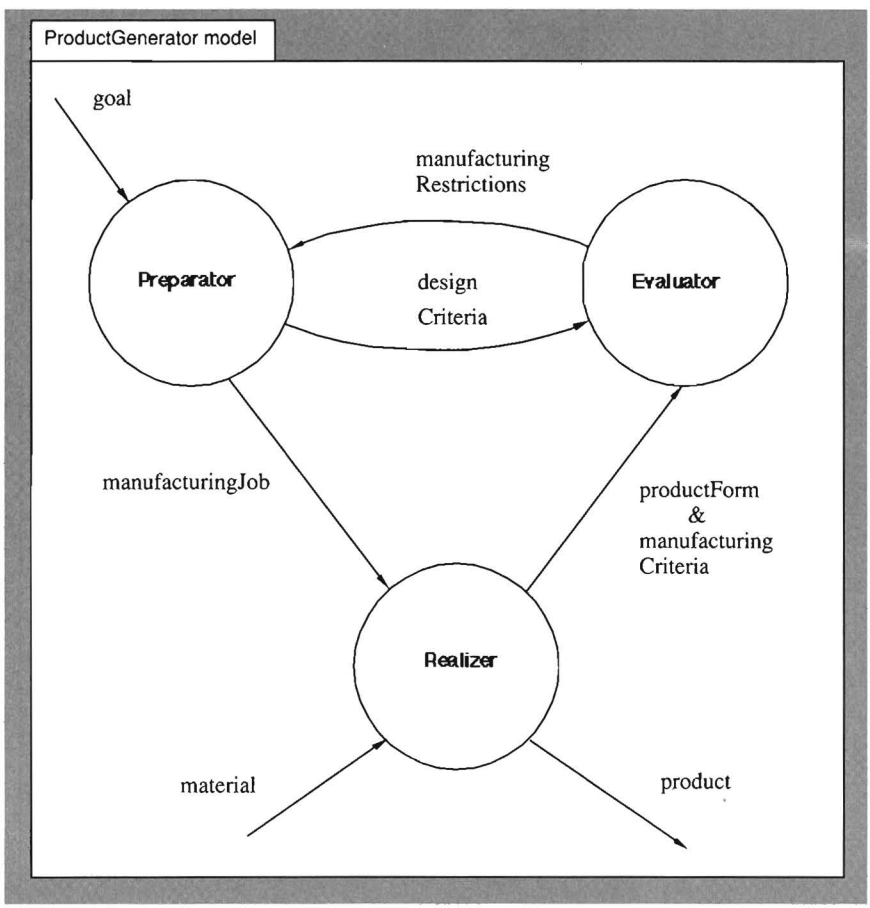


Figure II.1. The ProductGenerator model

The graphical representation of the Preparator model is shown in figure II.2.

Realizer model (figure II.1)

```
body
    manufacturingProcessPlan ← self receiveFrom: 'preparator'.
    material ← self receiveFrom: 'material'.
    product ← self manufactureProductUsing: manufacturingProcessPlan
            and: material.
    [measurements ← self measure: product.
     measurements complyWith: design]
        whileFalse: [self correct: product using: design].
    productForm ← self manufacturedForm: product.
    self send: product to: 'product'.
    self send: productForm to: 'evaluator'.
    manufacturingCriteria ←
        self generateEvaluationCriteriaFrom: product
                and: design.
    self send: manufacturingCriteria to: 'evaluator'
```

Evaluator model (figure II.1)

```
body
    designCriteria ← self receiveFrom: 'preparator'.
    productForm ← self receiveFrom: 'realizer'.
    manufacturingCriteria ← self receiveFrom: 'realizer'.
    manufacturingRestrictions ←
        self generateManufacturingRestrictionsFrom: productShape
                and: designCriteria
                and: manufacturingCriteria.
    self send: manufacturingRestrictions to: 'designer'
```

Designer model (figure II.2).

```
body
    productFunctions ← self receiveFrom: 'productFunctions'.
    manufacturingRestrictions ← self receiveFrom: 'evaluator'.
    manufacturingRestrictions isEmpty
        ifFalse: [ manufacturingRestrictionsCollection
                addAll: manufacturingRestrictions].
    design ← self generateDesign: productFunctions
                    using: manufacturingRestrictionsCollection.
    designCriteria ← self generateDesignEvaluationCriteriaFrom: design.
    self send: design to: 'manufacturingProcesPlanner'.
    self send: designCriteria to: 'evaluator'
```

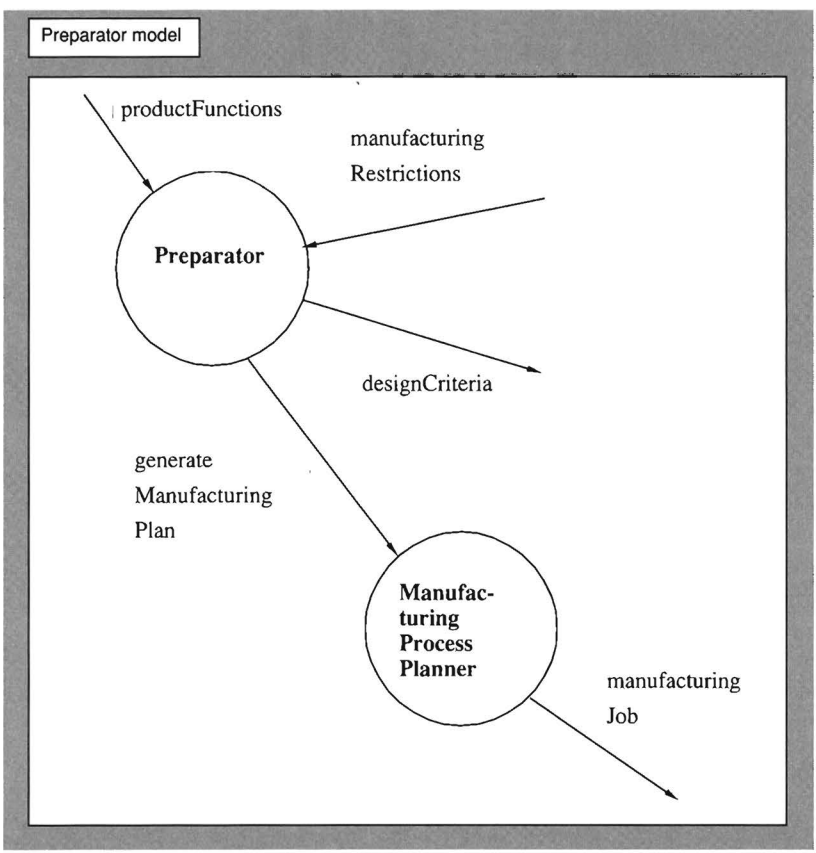


Figure II.2. The Preparator model

ManufacturingProcessPlanner model (figure II.2).

```
body
    design ← self receiveFrom: 'designer'.
    manufacturingProcessPlan ←
            self generateManufacturingProcessPlanUsing: design.
    self send: manufacturingProcessPlan to: 'realizer'
```

Appendix III

Definitions of Metric Spaces

Some definitions of metric spaces, needed for solid modelling, are given in the following sections (Simmons, 1963).

The distance between two points x and y in Euclidean three dimensional space E^3 is defined as $d(x, y) = \sqrt{((y1-x1)^2+(y2-x2)^2+(y3-x3)^2)}$.

A metric space is defined as a set X together with the non-negative function d on X*X. in such a metric space, the function d is called the metric and the members of X are called points.

Definition 1: Let X be a metric space with metric d. If x_0 is a point of X and r is a positive real number, the open sphere $s(x_0, r)$ with center x_0 and radius r, is the subset of X defined by
$s(x_0, r) = \{x \mid d(x, x_0) < r\}$

Definition 2: Let X be a metric space. A subset G of X is called an open set if, given any point $x \in G$, there exists a positive real number such that $s(x, r) \subseteq G$.

Definition 3: Let X be a metric space, and let A be a subset of X. A point x in X is called a limit point of A if each open sphere centered on x contains at least one point of A different from x.

Definition 4: Let X be a metric space. A subset F of X is called a closed set if it contains each of its limit points.

Definition 5: Let X be a metric space, and let A be a subset of X. The closure of A denoted by closure(A), is the union of A and the set of all its limit points.

Definition 6: Let X be a metric space and A a subset of X. A point x in X is called a boundary point of A if each open sphere centered on x intersects both A and its complement A' = X-A. The boundary of A, represented as $\beta(A)$, is the set of all boundary points of A.

Definition 7: Let X be a metric space and let A be a subset of X. A point in A is called an interior point of A if it is the center of some open sphere contained in A. The interior of A, denoted by int(A), is the set of all its interior points. The exterior of A, denoted as ext(A), is the complement of the closure of A.

Appendix IV

Some Definitions of Manufacturing-Oriented Design

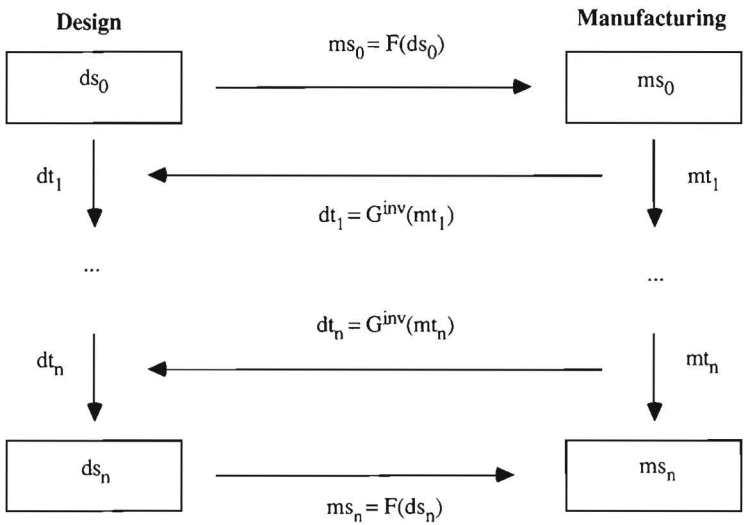


Figure IV.1. The relations between design and manufacturing in Manufacturing-Oriented Design

ds_i = the i^{th} design state. $i \in 0, 1, 2, 3, ...$

DS = the set of all design states that can be manufactured.

$ds_i \in DS$.

$dt_i(ds_{i-1})$ = the i^{th} design transformation performed on the $i\text{-}1^{th}$ design state.

$dt_i(ds_{i-1}) = ds_i$

IV.1. $ds_n = dt_n(dt_{n-1}(dt_{n-2}(...(dt_1(ds_0))))$

DT = the set of all design transformations that can be manufactured.

$dt_i \in DT$

ms_i = the i^{th} manufacturing state. $i \in 0, 1, 2, 3, ...$

MS = the set of all realizable manufacturing states.

$ms_i \in MS$.

$mt_i(ms_{i-1})$ = the i^{th} manufacturing transformation performed on the $i-1^{th}$ manufacturing state.

$mt_i(ms_{i-1}) = ms_i$

IV.2. $ms_n = mt_n(mt_{n-1}(mt_{n-2}(...(mt_1(ms_0))))$

MT = the set of all manufacturing transformations.

$mt_i \in MT$

IV.3. F = the function F transforms a design state into the corresponding manufacturing state (see figure IV.2).

IV.4. $ms_i = F(ds_i), ds_i = F^{inv}(ms_i)$

Equation IV.4. is needed in order to satisfy the unambiguity demand.

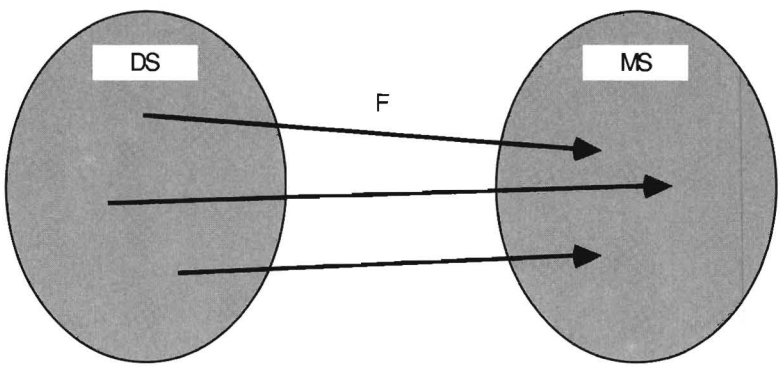


Figure IV.2. The relation between design and manufacturing states

IV.5. G = the mapping G transforms a design transformation into one or more manufacturing transformations, see figure IV.3.
G = G(dt_i, manufacturingMachine, set-ups, jigs, fixtures)
G^{inv}= G^{inv}(mt_i, manufacturingMachine, set-ups, jigs, fixtures)

IV.6. $\{\forall mt \in MT \mid \exists dt \in DT \wedge dt = G^{inv}(mt)\}$

The mapping G^{inv} is a one-to-one mapping from a manufacturing transformation onto the corresponding design transformation. The mapping G^{inv} introduces manufacturing restrictions in the design phase.

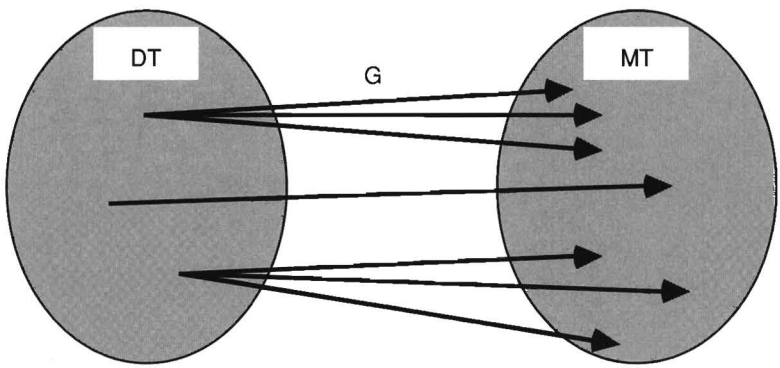


Figure IV.3. The relations between design and manufacturing transformations

IV.1. Manufacturable Objects

mod = a Manufacturable Object in the design phase.
MOd = the set of all Manufacturable Objects in the design phase.
mod ∈ MOd
mom = a Manufacturable Object in the manufacturing phase.
MOm = the set of all Manufacturable Objects in the manufacturing phase.

mom $\in$ MOm

IV.7. F(mod) = mom

AMOD = the set of all applications of the set MOd.

IV.8. AMOD $\subset$ DT

AMOM = the set of all applications of the set MOm.

IV.9. AMOM $\subset$ MT

From IV.6., IV.8. and IV.9. follows

IV.10. $\{\forall mt \in AMOM \mid \exists dt \in AMOD \wedge dt = G^{inv}(mt)\}$

IV.2. The Defined Design Transformations

a) **Apply Manufacturable Object**

ApplyManufacturableObjectD(ds_i, mod, L_{mo}) $\rightarrow$ ds_{i+1}

$ds_i \in DS \wedge ds_{i+1} \in DS \wedge mod \in MOd$

L_{mo} = The location, the position and orientation, of application of the Manufacturable Object.

The manufacturing counterpart:

ApplyManufacturableObjectM(ms_i, mom , L_{mo}) $\rightarrow$ ms_{i+1}

$ms_i = F(ds_i) \wedge ms_{i+1} = F(ds_{i+1}) \wedge mom = F(mod)$

The manufacturing counterpart is, for instance, the material removal of the manufacturing counterpart mom of the design Manufacturable Object mod.

b) **Locate**

LocateD(ds_i) $\rightarrow$ ds_{i+1}

$ds_i \in DS \wedge ds_{i+1} \in DS$

The manufacturing counterpart:

LocateM(ms_i) $\rightarrow$ **ms_{i+1}**

$ms_i = F(ds_i) \wedge ms_{i+1} = F(ds_{i+1})$

The manufacturing counterpart is the locating, the positioning and the orientating, of the manufacturing counterpart ms_i of the design state ds_i.

c) Implicit Locating

ImplicitLocatingD(ds_i, relativeToDesignObject, constraints) $\rightarrow$ **ds_{i+1}**

$ds_i \in DS \wedge ds_{i+1} \in DS \wedge$ relativeToDesignObject $\in DS$

The manufacturing counterpart:

ImplicitLocatingM(ms_i, relativeToManufacturedObject, constraints) $\rightarrow$ **ms_{i+1}**

$ms_i = F(ds_i) \wedge ms_{i+1} = F(ds_{i+1}) \wedge$

relativeToManufacturedObject = F(relativeToDesignObject)

d) Combine

Combine(ds_i, ds_p) $\rightarrow$ **ds_{i+1}**

$ds_i \in DS \wedge ds_p \in DS \wedge ds_{i+1} \in DS$

The manufacturing counterpart is, for instance, the welding or gluing of the manufacturing counterparts ms_i and ms_p of the design states ds_i and ds_p.

Weld(ms_i, ms_p) $\rightarrow$ **ms_{i+1}**

Glue(ms_i, ms_p) $\rightarrow$ **ms_{i+1}**

$ms_i = F(ds_i) \wedge ms_{i+1} = F(ds_{i+1}) \wedge ms_p = F(ds_p)$

e) **Merge**

Merge(ds_i, ds_p)→ ds_0

$ds_i \in DS \wedge ds_p \in DS \wedge ds_0 \in DS$

The manufacturing counterpart is for instance, the shaping by casting of the manufacturing counterpart of the design state ds_0.

ShapingByCasting(ms_i, ms_p) → ms_0

$ms_i = F(ds_i) \wedge ms_0 = F(ds_0) \wedge ms_p = F(ds_p)$

The previous design states before the ds_0 are only needed to specify how the positive of the diecast can be manufactured.

Curriculum Vitae

Frank Delbressine was born on February 26th, 1957 in Beek, The Netherlands. He attended the Bisschoppelijk College in Echt, The Netherlands, where he obtained his Atheneum-B diploma before commencing his studies in Mechanical Engineering at the Higher Technical School in Heerlen, The Netherlands.

Immediately afterwards, he entered Eindhoven University of Technology, Eindhoven, The Netherlands, to continue his study in Mechanical Engineering. He received his Master's degree in April 1985. In the same month, he was engaged by the Faculty of Mechanical Engineering at Eindhoven University of Technology and he started his doctoral studies in the area of the integration of design and manufacturing.

STELLINGEN

behorende bij het proefschrift van

F.L.M. Delbressine

1 Zonder menselijke interventie is integratie van ontwerpen en fabriceren onmogelijk.

Dit proefschrift.

2 Een integratie van ontwerpen en fabriceren kan alleen dan gerealiseerd worden, indien de fabricage beperkingen al in de ontwerpfase gehanteerd kunnen en dienen te worden.

Dit proefschrift.

3 Programmeertalen voor machine besturingen zijn, vergeleken met de hogere programmeertalen, ongestructureerd te noemen.

Dit proefschrift.

4 De topologie dient uitgebreid te worden met fabriceerbaarheids criteria en fabriceerbare transformaties.

Dit proefschrift.

5 Een nieuwe type mens is aan het ontstaan: Homo Mus Computator, deze wordt gekenmerkt door een sterk ontwikkelde rechter- c.q. linker onderarm en door een goed ontwikkelde oog-hand coordinatie van alle muizenissen.

6 Robots zouden bij voorkeur dienen te bestaan uit translatorische elementen.

Hijink J.A.W.,
Modale analyse als experimentele techniek.
Intern rapport faculteit W, WPB nr. 0029.
Technische Universiteit Eindhoven, Eindhoven, 1983.

7 Het in het buitenland blijvend-verkerende werk van Nederlandse kunstenaars, zou beter toegankelijk moeten worden gemaakt voor Nederlanders.

8 Het is zeer verlokkend om alles in algorithmen te gieten als een van de weinige gereedschappen die ter beschikking staan een computer is.

Maslow A.H.,
The psychology of science.
Harper & Row, New York, NY (1966).

9 De Hoge Raad zou het recht moeten krijgen elke wet aan de grondwet te toetsen.

10 Aan een promotie-onderzoek werken altijd meer mensen mee dan de promovendus, de afschaffing van een dankwoord in het proefschrift is derhalve niet juist.