

Chapter 1

Introduction

Concurrent engineering and collaborative engineering are modern philosophies to reengineer the traditional product design and development processes leading to better quality, shorter lead-time, more competitive cost and higher customer satisfaction. A full-scale implementation of the philosophies to form an integrated and collaborative product development environment includes two aspects: (1) the upstream design and downstream manufacturing processes are seamlessly linked through the implementation of some intelligent reasoning and integration strategies for effective information exchange, and (2) collaborative design, fuelled by cooperation strategies and the Internet technologies, is realised through collocating a multi-disciplinary design team and emphasising interpersonal aspects of the group work.

In this chapter, the motivations, concepts and research issues for the establishment of an integrated and collaborative environment for product development are introduced. The enabling Artificial Intelligence (AI) and Internet technologies for system implementation are discussed.

1.1 Concurrent and Collaborative Engineering

In traditional design and manufacturing companies, the product development process is usually organised in a sequential manner, i.e., Sequential Engineering (SE). In SE, specialists from different domains that are involved in product design and the related manufacturing processes work in an isolated and independent way. A stage in product development can be basically thought of as being a black box, which inputs are the results of the activities from the previous stage, and which

outputs are taken as inputs for the next stage, as shown in Fig. 1.1. The shortcomings of SE include high developmental costs, expensive re-designs and re-work, poor communications and interactions between designers/process planners/manufacturing engineers, the lack of critical decision-making with respect to all of the product development, etc. [Ranky, 1994].

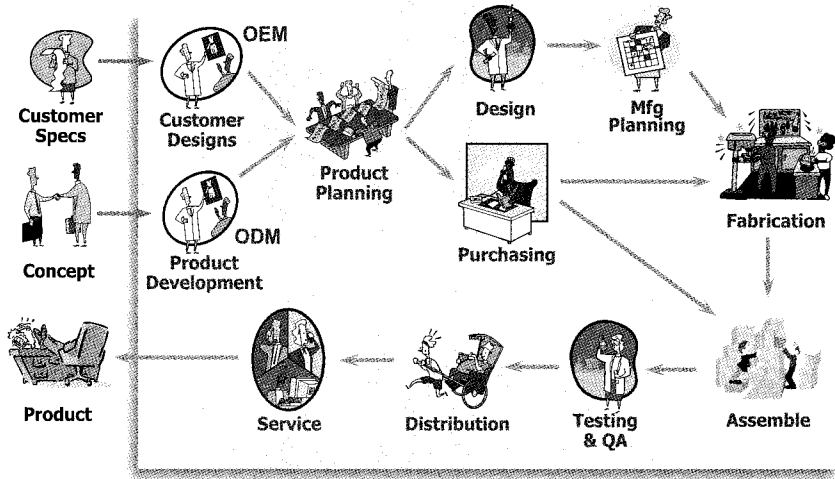


Fig. 1.1 The sequential development process of products.

In a product development process, design concepts and models usually require dynamic adjustments to achieve better product quality and processes. Changes made during the early design stage do not cause significant increase in cost, while during the production stage, sharp increase in cost will incur since many blueprints, design documents or components have been created and these would require re-work and re-design. Subsequently, the lead-time and cost of the product will dramatically increase. This situation is illustrated in Fig. 1.2, which reflects the cost trends of products with respect to the changes in their life-cycle development [Pallot and Sandoval, 1998]. It shows clearly that greater cost reduction opportunities are usually available during the early stages of the product development and therefore it is of paramount

importance to identify design and manufacturing problems as early as possible to reduce the risk and cost.

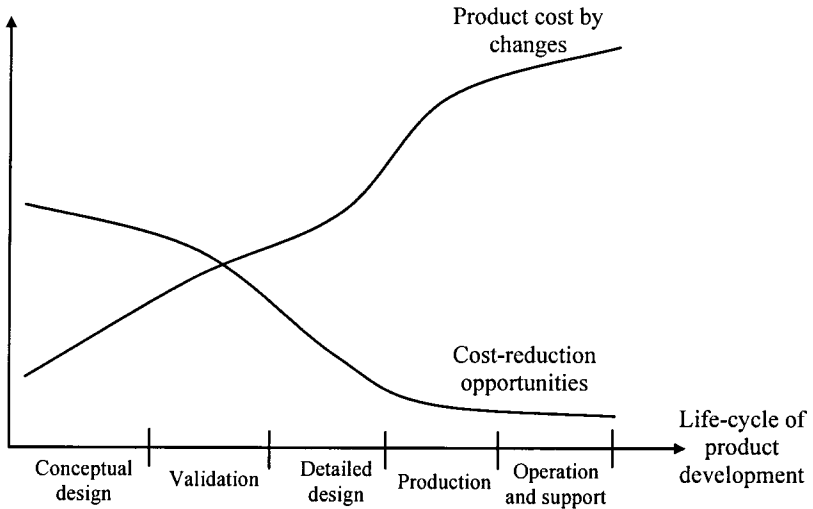


Fig. 1.2 Product cost and cost-reduction opportunities from the life-cycle viewpoint.

Modern product development practices are becoming more product-oriented, and aim at decreasing the lead-time, minimising work-in-process, just-in-time flow of materials, and high efficiency and flexibility of manufacturing capacity utilisation. With the advances in computers and computing technologies, a large number of software tools and philosophies have come into existence to facilitate the product development and realisation processes. From 1990s, the Concurrent Engineering (CE) concept has been adopted by design and manufacturing companies to systematically integrate the design of products with the related manufacturing processes using some software packages and computing technologies in an integrated computing environment. In a CE environment, techniques, algorithms and software tools are provided to allow designers and developers to interact with each other effectively and efficiently. With the implementation of CE, which usually causes more time and money to be spent in the initial design

stage to ensure that the concept selection is optimised, companies can reduce design changes at the later stages, leading to better engineered products with an advantage in total quality, time and cost competitiveness (shown in Fig. 1.3).

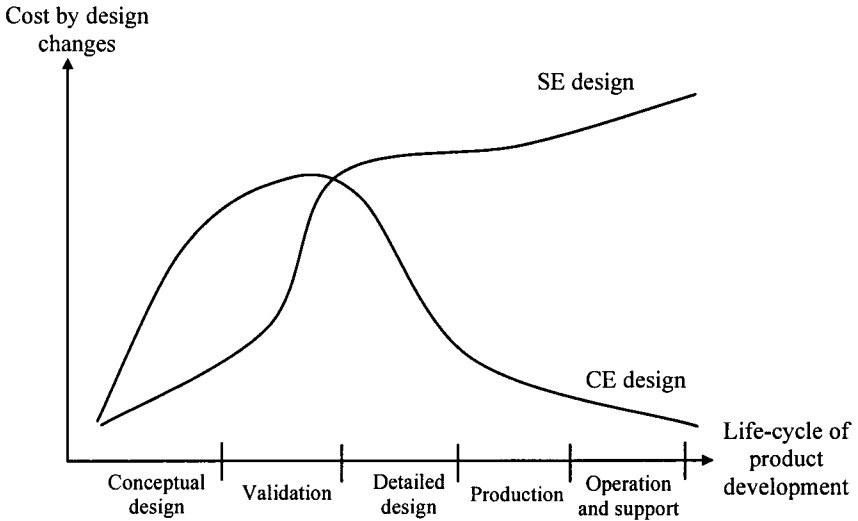


Fig. 1.3 Design changes and the cost incurred in industries.

The CE philosophy can be implemented through different strategies considering the diverse requirements of the users and the conditions of the companies. Pallot and Sandoval [1998] emphasised the communication and sharing of design data, integration and interoperability of the application systems in design and manufacturing, and the coordination between the upstream and downstream activities to support concurrent activities for multi-disciplinary teams. They also investigated some CE related projects that have been funded by the European information technology research program, ESPRIT. The ATLAS project allows the sharing of product models across functional departments based on an international data exchange standard, STEP (Standard for the Exchange of Product model data). Its initiative is to establish a generic data information sharing model for manufacturing enterprises to realise the seamless integration and interoperability of

design and manufacturing information. The FIRES project and the SCOPES project focused on the Design for Manufacturability and Assemblability (DFM/A) methodologies respectively to bring manufacturing engineers into the early design decision-making process to optimise the manufacturing and assembling processes when a product is being designed, thereby improving the overall performance of the product. The VEGA project and the MATES project aimed to establish distributed information infrastructures to support remote communication in line with the rapid industrial requirements and the emerging Internet technologies.

From an industrial viewpoint, Balamuralikrishna, *et al.* [2000] emphasised three T's for implementing CE: tools, training and time. Tools refer to the communication facilities between the personals in the multi-disciplinary departments to address the information exchange that is obstructed by the complexity and wide range of specialised disciplinary areas and interdependent activities. One of the greatest challenges in managing the simultaneous operation of inter-related tasks is getting the people to work together as a team. Training provides a mechanism for employees to work collaboratively and concurrently, making the best use of the company resources. Time means corporations need time to carefully investigate and plan CE as it involves many complex software tools and information infrastructures. Many reported cases have shown that a hurried implementation of CE usually brings high probability of backfiring. In summary, a successful implementation of CE needs to consider the following aspects (but not limited to):

- A communication strategy for a multi-disciplinary group of people from the design and manufacturing departments to share and exchange ideas and comments.
- An integration strategy to link heterogeneous software tools in product design, analysis, simulation and manufacturing optimisation to realise obstacle-free engineering information exchange and sharing.
- An interoperability strategy to manipulate downstream manufacturing applications as services to enable designers to evaluate manufacturability or assemblability as early as possible.

With the trend towards global competition and the rapid advances of the Internet technologies, nowadays, extensive research and development have been made towards supporting distributed applications to form a wider landscape, in which geographically dispersed users, systems, resources and services can be synthesised across enterprises in an Internet/Intranet environment beyond the traditional boundaries of physical and time zones. Face-to-face communication and cooperation is impossible in this situation. Some traditional communication methods that are used in a traditional CE environment, such as emails, discussion forums and net-meetings, are not fully satisfactory. To address this issue, Internet-enabled collaborative engineering and the related techniques are developing at a rapid pace since the end of the last century. A variety of commercial tools have been launched to support engineering distribution and collaboration, e.g., SmarTeam™ (www.smarteam.com), TeamCenter™ (www.ugs.com/products/teamcenter), ProjectLink™ (www.ptc.com/products/windchill/projectlink.htm), Onespace™ (www.onspace.net), Adaptive Media Envision3D™ (www.adaptivemedia.com), Streamline™ (www.autodesk.com), etc. As a successful industrial application example, Ford worked collaboratively with its acquired Volvo from their separate sites on car design based on a collaborative design platform. With the help of collaborative tools, small and medium enterprises or even individual designers with specific domain knowledge will be able to participate and collaborate in the design process with large manufacturing companies.

An Internet-based collaborative design environment, which is illustrated in Fig. 1.4, consists of two capabilities, namely, distribution and collaboration. Distribution means systems that are geographically dispersed can be linked to support remote design activities, while collaboration allows individual designers to be associated and coordinated to fulfil a global design target and objective. A collaboration mechanism requires the specific design of a distributed architecture of a system to meet the functional and performance requirements.

CE and Internet-based collaborative engineering are complementary in functions. The former emphasises on a vertically seamless integration of the upstream design and downstream manufacturing processes through leveraging on some intelligent strategies to realise information sharing

and flowing. The latter focuses on the horizontally interpersonal aspects of group work across the whole design chains. An Integrated and Collaborative Environment (ICE) for product development is actively being investigated to incorporate the two strategies to meet the following requirements:

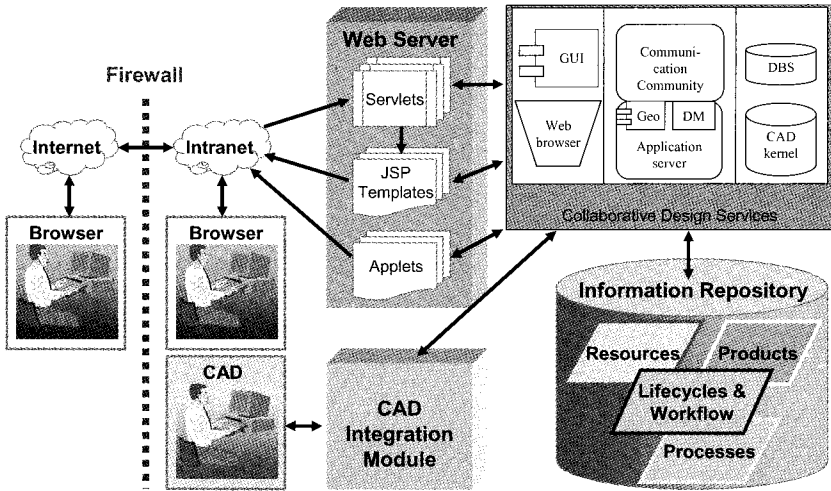


Fig. 1.4 A collaborative product development environment over the Internet/Intranet.

- Enterprise integration for the distributed organisations and systems. Manufacturing companies and enterprises can be integrated with their distributed systems and partners, such as customers and suppliers, via networks to establish an ICE for product development. For instance, the distributed design and manufacturing analysis systems can send the demands and requirements from the customers directly to the design department of a company to support global competitiveness and rapid market responsiveness.
- Heterogeneous environments and interoperability of software tools. An ICE will allow the integration and interoperability of heterogeneous software and hardware. Information environments and legacy systems in companies are usually based on different

programming languages, representation languages and models for product information, and computing platforms. To achieve an effective and efficient interoperation and interaction of sub-systems and software components in such heterogeneous environments, automatic information conversion and interpretation capabilities are necessary to realise obstacle-free information communication and workflow control.

- Open and scalable computing structure and services. There is a need to provide a possibility to dynamically integrate new sub-systems into or remove existing sub-systems from an ICE-enabled product development with high convenience, security, reliability and without stopping and re-initialising the entire environment. New kinds of service architectures to wrap software tools have to be developed to incorporate them into the environment as required, so as not to interrupt organisational links previously established.
- Cooperation between humans, and between systems and humans. People and software systems need to work at various levels of collaboration, and with rapid access to knowledge and information repositories. Bi-directional communication infrastructures are necessary to allow effective and quick communication between systems or between humans and systems to facilitate their interactions.

1.2 Enabling Technologies

Product design and manufacturing are semantically rich domains. Problems in these domains, such as reasoning, analysis, process planning, etc., are difficult to solve as a large amount of data and information are available, and complex decision-making processes are involved. It is possible to utilise AI techniques as an enabling technology to facilitate the relevant applications and to further form an integrated environment. Meanwhile, the advances of collaborative capabilities in systems is fully driven by the development and deployment of the Internet technologies, such as Java, .Net, Web, HTML (HyperText Markup Language), XML (eXtensible Markup Language) or Web

service techniques, for building up the information infrastructures. In the following content, the features of these enabling technologies, especially those used in the rest of the book, will be highlighted.

1.2.1 Artificial intelligence

AI is the branch of computer science dealing with the design of computer algorithms and systems that exhibit characteristics associated with intelligence in human behaviours, including reasoning, learning, self-improvement, goal-seeking, self-maintenance, problem solving and adaptability. With the increase of hardware speed and advances in AI methodologies and algorithms, AI techniques have been widely used in design and manufacturing activities to solve problems in an efficient and successful way [Dagli, 1994; Teti and Kumara, 1997; Rao, *et al.*, 1999]. For instance, the Artificial Neural Network (ANN), which can be functional in pattern recognition and pattern association, has been successfully applied to manufacturing feature recognition to convert a product design model to a downstream manufacturing model. The Genetic Algorithm (GA), the Simulated Annealing (SA) algorithm and the Tabu Search (TS) algorithm can be utilised in the optimisation of process plans for products to reduce their manufacturing cost and improve the quality of the solutions.

1.2.1.1 Artificial neural network

An ANN is an information-processing technique that has certain performance characteristics in accordance with a biological neural network. An ANN, which is a specific mathematical model and algorithm to simulate neural biology, can be applied to numerous problems, such as storing data and patterns, classifying patterns, performing general mapping from input to output, and finding optimal solutions to optimisation problems [Fausett, 1994]. Usually, an ANN consists of four parts: (1) a series of simple elements, namely neurons (or units, nodes, or short-term memory elements); (2) neurons are linked to form a network and each link is associated with a weight; (3) each

neuron has an input and an output, and is associated with an activation function to sum up the weighted input signals from its connected neurons to determine its output signal; and (4) each network has one input and one output, and signals are passed from neurons for input to neurons for output over connection links. A multi-layer neural network is illustrated in Fig. 1.5(a), and the computation process for a neuron y_l is shown in Fig. 1.5(b).

There are three design considerations of an ANN: (1) its architecture for organising neurons and their links, (2) a learning method to determine the weights on the links, and (3) an activation function of the neurons.

An ANN can provide a high degree of robustness due to the massive parallelism in its design. It is often used in situations where only a few decisions are required from a massive amount of data, or a complex non-linear mapping needs to be learned. Intelligent manufacturing is one of the most important and successful application domains for ANNs. Two ANNs commonly used in manufacturing feature recognition are: the Multiple-Layer Feed-Forward (MLFF) net trained using a Back-Propagation (BP) training algorithm, and the Adaptive Resonance Theory (ART) net.

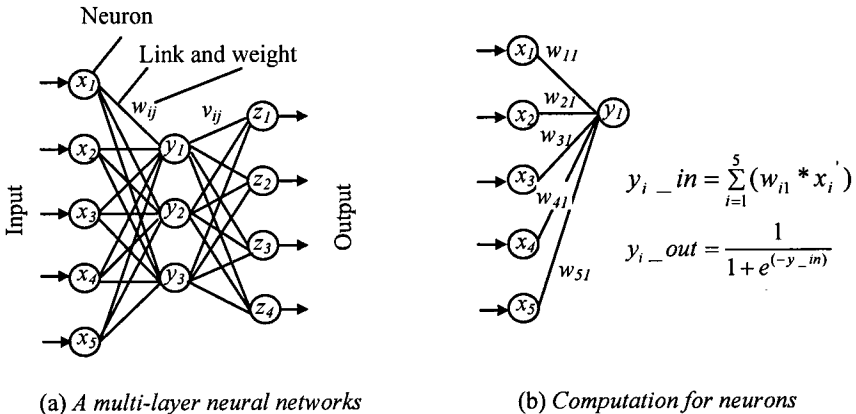


Fig. 1.5 A multi-layer neural networks and the computation process.

(1) MLFF net with BP

An MLFF net with BP is a gradient descent method to minimise the total squared error of the output computed by the net. An MLFF net with BP is a supervised-learning net, i.e., the net can be trained to map a given vector of inputs to a specified vector of target outputs. After training, it can give reasonable or good responses to an input that is similar to that used in training. The process of computing signals in an MLFF net is forward from the input layer to the output layer, whereas the training process (BP algorithm) of the net is in reverse. This process iterates until the weights of the net are adjusted (trained) to generate the same or similar signals for the input signals. The computation processes are illustrated in Fig. 1.6.

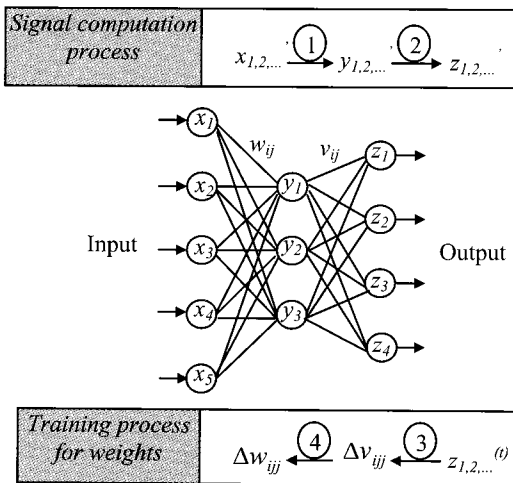


Fig. 1.6 Signal computation process and training process.

The main parameters to be determined for an MLFF with BP include the number of hidden layers, the number of nodes in the hidden layers, the choice of the initial weights and biases, and certain control parameters, such as the activation functions and the learning rate of the net. Since an MLFF net with BP can perform a stochastic gradient

descent in a weighted space and can be easily realised, it has emerged as the most popular algorithm in the category of supervised training algorithms. However, there are some serious drawbacks of the net that limit its potential applications, namely:

- There is no specific mathematical method to determine the main parameters of the net, such as the number of hidden layers, the number of nodes of the hidden layer(s), etc. The determination of these parameters is quite time-consuming and tedious as it requires many trials;
- Since the mechanism of the net is a gradient descent method, results are liable to be trapped in some local minimum points. Whether a global minimum can be achieved or not depends much upon human experience and trials.

(2) ART net

An ART net is an unsupervised learning net, i.e., through adjusting the vigilance parameter in the net, the similarities among the input patterns can be controlled and these patterns can be categorised into different families.

An ART net, as shown in Fig.1.7, usually involves three groups of neurons, namely, input processing units (F_1 layer), cluster units (F_2 layer) and reset units. The F_1 layer consists of n numbers of each type of units (where n is the dimension of an input pattern). In the F_2 layer, the signals of the input units are combined and the similarities of the input vectors are compared with the weights of the cluster units. There are two sets of weights between the F_1 and F_2 layers. The bottom-up weights from F_1 to F_2 are denoted as b_{ij} , and the top-down weights from F_2 to F_1 are t_{kj} . Based on a “vigilance parameter” (ρ) in the reset units, the cluster units can decide whether to learn an input vector or not. Compared with an MLFF net with BP, an ART net has two advantages:

- An ART net has the ability to remain stable to preserve significant past learning while it remains adaptable enough to incorporate new information, whenever it might appear;
- The stability time for an ART net is much shorter than the training time of an MLFF net with BP. There are also fewer control

parameters in the ART net. Compared with an MLFF net, the ART net is easier to control and manipulate.

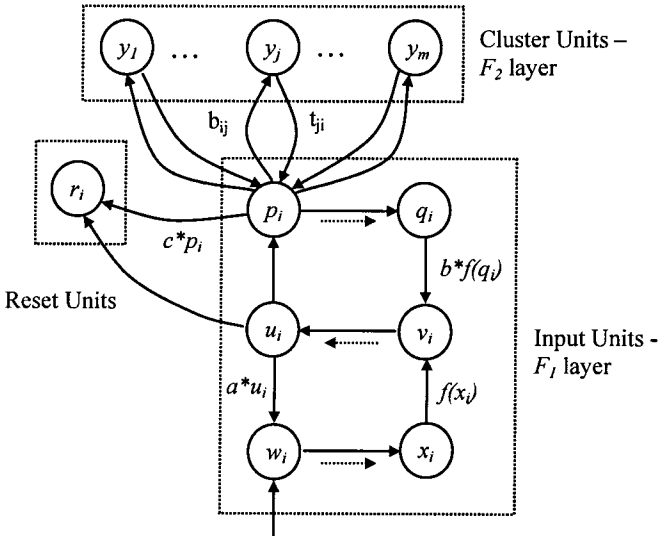


Fig. 1.7 A typical ART architecture.

There are two kinds of ART nets, namely, the ART 1 net and ART 2 net [Fausett, 1994; Haykin, 1999]. The architectures and basic mechanisms of the ART 1 and ART 2 nets are similar. However, ART 1 is designed to cluster binary input vectors and ART 2 handles continuous-valued input vectors. Therefore, an ART 2 net can be applied to more complex situations and wider application domains than an ART 1 net, benefiting from its more flexible input representation scheme. On the other hand, the input field of ART 2 is more complex since it handles continuous-valued input vectors and needs to avoid vectors that are arbitrarily close together. The architecture and computing process of an ART 2 will be explained in Chapter 3, in which a hybrid AI technique, including heuristic reasoning, graph manipulation, and an ART 2 net, has been designed for recognising interacting manufacturing features from a design part. A comparison of applying ART 2 and MLFF nets for this problem is made to show their characteristics.

1.2.1.2 Heuristic optimisation techniques

Conventional optimisation algorithms are often incapable of optimising non-linear multi-modal functions. A random search method might be helpful to search for an optimal solution. However, an undirected search is inefficient for large domains. To address this problem effectively, some heuristic optimisation techniques, such as GA, SA and TS, have been proposed to quickly find a solution in a large searching space through some embedded intelligent heuristic strategies [Pham and Karaboga, 2000]. Some basic concepts of these techniques are as follows:

- **Representation.** A heuristic algorithm does not use much knowledge about an optimisation problem and deals directly with the parameters of the problem. It can work with chromosomes (for GA) or solutions (for SA and TS) that represent the parameters. The optimisation problem can be represented as a string, which includes two popular schemes: binary string representation and integer/real number representation. The scheme to use is determined by the ease of modelling the problem itself as well as the performance of the algorithm in terms of accuracy and computation time.
- **Evaluation.** A fitness (objective) function is modelled based on certain criteria to evaluate the chromosomes or solutions. The role of the fitness function is twofold: (1) it is used as a selection operation to determine the individual chromosome or solution to be reproduced to drive the optimisation procedure; and (2) it is used as a stopping condition to determine whether a chromosome or solution has been satisfied, so as to stop or continue the search procedure. A suitable fitness function can be a mathematical equation. Where this method cannot be used, a rule-based procedure can be constructed.
- **Optimisation strategies.** Based on the chosen representation scheme, a GA will form a group of initial chromosomes (i.e., a population) for certain genetic operations, such as selection, crossover and mutation, to be executed on to achieve an optimal solution. For SA and TS, starting from a single solution, some neighbourhood strategies are applied to generate a group of solutions and a solution

is selected according to certain criteria as a starting point to continue the optimisation process.

(1) Genetic algorithm

A GA is modelled based on natural evolution in that the operators a GA employs are inspired by a natural evolution process. These genetic operators, including selection, crossover and mutation, can be used to manipulate the chromosomes (solutions to a problem) in a population over several generations to improve its fitness function gradually. The procedure of a GA is shown in Fig. 1.8.

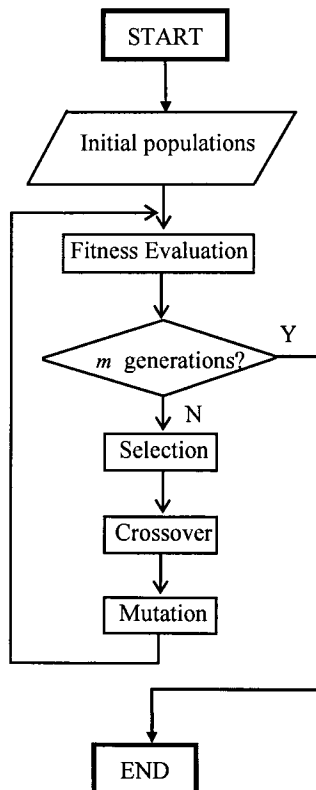


Fig. 1.8 The flowchart of a GA.

A GA works with chromosomes that represent the parameters. Several important issues to be considered when applying a GA to an application problem include:

- Representation schemes of the chromosomes, and the definition of an evaluation function.
- The selection of a suitable procedure for each genetic operator to improve the efficiency and quality of the search. For instance, for the selection operator, there are mainly two alternative procedures: proportional selection (“roulette wheel”) and ranking-based selection. In the crossover operator, some common alternative strategies include the one-point crossover, two-point crossover and cycle crossover.
- In the designed chromosomes, there are usually some precedence constraints in them. The crossover and mutation operations employed in a GA might cause the precedence constraints to be violated. The method to handle constraints and search the feasible space is a major difficulty in applying GA.
- The choice of the GA control parameters depends on the problem and the representation schemes employed. Important parameters include the population size, the crossover rate and the mutation rate. These parameters should be designed for a general condition for the problem instead of being specific for a certain case study.

(2) Simulated annealing algorithm

An SA is a stochastic searching algorithm based on the principle of the real annealing process in metallurgy. It comes from an algorithmic analogy with the annealing of materials where the purpose is to lead the material to a state corresponding to a global minimum of its internal energy. This process is attained by searching the global minimum of an objective function in a space of solutions. An SA uses a control parameter called “temperature” that is decreased through iterations until it becomes close to zero [Kirkpatrick 1983]. The solution is sought by iterating and evaluating the energy at each stage. This method allows the algorithm to generate random points in which the fitness function has a greater value, i.e., the solution has a higher energy. When the

temperature is high, the algorithm will be likely to accept a higher energy solution, while at a very low temperature the algorithm will almost always only accept solutions of lower energy. Solutions are accepted according to the Boltzman probability [Aarts, 1989], and new solutions are randomly chosen in a feasible domain to satisfy constraints. The temperature begins at a high level and is cooled until an equilibrium is reached by allowing the initial temperature to seek a global optimum. Without this feature, it is possible to be trapped in a local minimum. By allowing the function to move to a higher value, it is able to climb over the hill and find the global minimum, as illustrated in Fig. 1.9. Several important issues to be considered when applying a SA include:

- Representation schemes of the solutions.
- Definition of an evaluation function.
- Definition of a neighbourhood mechanism for the generation of temporary solutions. Various neighbourhood mechanisms for the generation of temporary solutions can be developed while some of them could be adopted from a GA, for instance, crossover and mutation operators.
- Design of a cooling schedule. The parameters in the cooling schedule to be determined include an initial temperature, a temperature update rule, the number of iterations to be performed at each temperature step and a stopping criterion for the search.

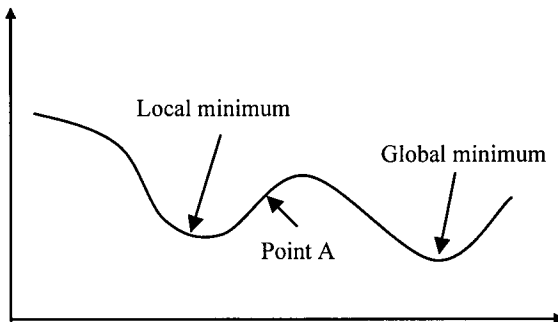


Fig. 1.9 Hill-climbing analogy in an SA.

It has been recognised that GA is not well-suited to perform finely-tuned local search [Rogers, 1991; Ishibuchi, *et al.*, 1994; Mathias, *et al.*, 1994; Yen, *et al.*, 1998]. Compared with a GA, an SA is more efficient in searching for a global or near-global optimisation solution, but inefficient in obtaining a group of solutions with good performance. Meanwhile, it has been shown that to control and adjust the parameters of an SA in an entire search space is not easy. Therefore, in the case of searching for a group of solutions for a global or near-global optimisation solution, it is meaningful to make an attempt to combine the strengths of a GA and an SA. Once the high performance regions of a search space have been identified using a GA, an SA should be invoked as a local search routine to optimise the members of the final population. In Chapter 5, a hybrid GA-SA method has been designed to optimise a process planning problem for optimal or near-optimal solutions.

(3) Tabu search algorithm

TS, which utilises some selected concepts of GA and SA and is characterised by the use of a flexible memory strategy, is a meta-heuristic algorithm that can guide search processes to overcome local optimal solutions in combinatorial optimisation problems. The fundamental of TS is to avoid entrapping in cycles by forbidding moves that take the solution to points in the solution space previously visited (hence “taboo”) [Glover, 1997]. Although TS is still in its infancy stage, during the last few years, it has been reported as a satisfactory solution approach for a variety of problems, such as scheduling, parallel computing, transportation, routing and network design.

In a TS algorithm, there are three main strategies, namely, the forbidding strategy, the freeing strategy and the aspiration strategy. During a search process, a Tabu list for recording the recently past moves is established and dynamically maintained. The strategies are briefly explained below.

- The forbidding strategy controls the solution that enters the Tabu list. It can avoid cycling and local minimums by forbidding certain moves during the most recent computational iterations.

- The freeing strategy is used to manage the solution that exits the Tabu list and when this occurs.
- The aspiration strategy is the interplay between the forbidding and freeing strategies for selecting trial solutions. A solution that has been forbidden by the Tabu list can become acceptable if it satisfies a certain criterion.

In Chapter 7, a TS-based approach is used to optimise a process planning problem for optimal or near-optimal solutions, and the relevant results are compared with the GA and SA approaches.

1.2.2 Internet technologies

The Internet technologies are evolving quickly recently. Many competing or complementary technologies have been launched by large software vendors, such as Microsoft, Sun and IBM. A few popular technologies are briefly introduced here, and information for applying the relevant Internet technologies to establish a collaborative system is surveyed in Chapter 6.

1.2.2.1 Client/server architecture

In Internet applications, the client/server architecture is the most commonly used paradigm to separate business logic, data processing and presentation of data. Clients are graphical user interfaces to present and render data, and servers serve as data repositories or executive processors for the data according to certain logics and programs. The original architecture of a client/server system is two-tier, and later migrates to three- or even n-tier (a three-tier architecture is shown in Fig. 1.10).

The motivation behind an n-tier architecture is to separate the application logic (in servers) from the user interface (in clients), which brings flexibility to the design of an Internet application. For instance, multiple user interfaces can be built without changes to the application logic and the relevant programs. Meanwhile, administration functions can be deployed in a server for effectively coordinating and monitoring the clients as a team. Recently, there have been much research activities

on a peer-to-peer architecture to allow a group of computers with equivalent responsibilities to be connected so as to pool their resources and decentralise the management, which is suitable for multi-agent systems, parallel computing and grid computing. To choose a suitable architecture and arrange the functional components in different computers in a network requires careful investigations according to the practical requirements and conditions.

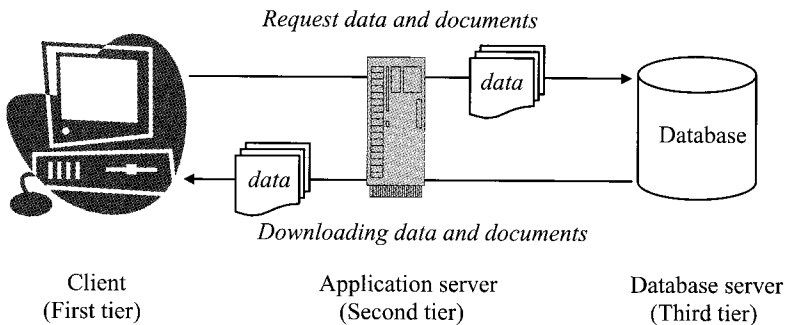


Fig. 1.10 A three-tier client/server architecture.

1.2.2.2 Representations for documents and design models

Along with HTTP (HyperText Transfer Protocol), which enables the cross-platform and cross-enterprise multi-media exchange of information from a server to a client or an end-user, HTML is a scripting language to organise texts, pictures, data and documents and render them in a Web browser in certain formats. HTML information is saved in plain text (e.g., ASCII), and the representation formats of the data and information are controlled by the HTML commands (“tags”), attributes and values. A drawback of HTML is that it does not allow users to create their own tags or attributes to parameterise or semantically specify their data. Another drawback is that it lacks a complex structure definition such that it cannot support the specification of structures needed to present database schemas or object-oriented hierarchies. Hence, design and

manufacturing information loses its structure when translated into a HTML file, and the embodied information is difficult to process and be extended.

To eliminate the drawbacks of HTML, XML was created to support distributed structured data and documents over the Web. XML offers several potential advantages that can be used in a collaborative environment. For instance, XML can define data in a document-oriented presentation format to separate a document's logical structure from its document-oriented presentation. Based on this characteristic, a user can specify several styles for the same XML document, and the same document structure can be defined as several output formats for different applications in the design and manufacture domains. XML can build documents from heterogeneous data and information. Hence, design and manufacturing applications, which might include heterogeneous information sources, such as documents, knowledge bases, relational or object-oriented databases, and case bases, can be integrated through utilising XML documents.

Research on the sharing of multi-media information, especially 3D modelling geometrical information on the Internet/intranet in real-time has been carried out recently. Such a capability can support heterogeneous software tools in an ICE, shorten a product development process as well as improve its quality. VRML (Virtual Reality Mark-up Language) is a language for describing interactive simulation of 3D models to be included in a HTML file for rendering in a Web browser. Essentially, VRML files describe a 3D scene in terms of objects, operations, and properties of the scene. Triangle, quadrangle and hexagon are several popular schemes to represent the meshes of a VRML model. Fig. 1.11 shows an example of the VRML syntax and its corresponding visual effect as seen in a VRML viewer. The basic geometry information is represented by the contents in *point* (providing the 3D coordinates of vertices) and *coordIndex* (providing the vertex indices in a certain sequence to compose the triangular meshes).

Other standards, such as X3D (eXtensible 3D) (www.x3d.com) and MPEG-4, which are functionally equivalent to VRML, extend the support of XML-based representation and video/audio application in compressed binary formats, respectively. OpenHSF (www.openhsf.org)

and XGL/ZGL (www.xglspec.org) enhance the capability of VRML for effective 3D streaming transmission of large-volume data over the Internet through data compression, mesh simplification and object prioritising to facilitate real-time collaboration.

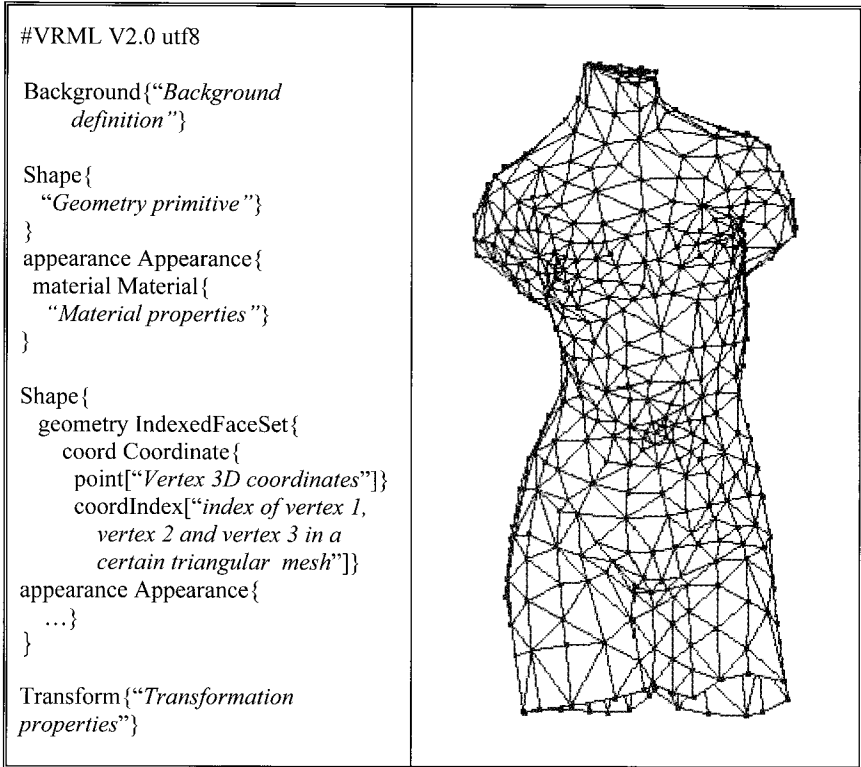


Fig. 1.11 The VRML syntax and its corresponding visual object.

1.2.2.3 Distributed enterprise system integration paradigms

Microsoft's .Net, OMG (Object Management Group)'s CORBA (Common Object Request Broker Architecture) and Sun Microsystems' J2EE (Java 2 platform Enterprise Edition) are the most popular distributed enterprise system integration paradigms used to establish a

more complex collaborative product development environment on the Internet/Intranet. Each strategy has its characteristics. .Net is now heavily used on the Microsoft Windows platform, while CORBA and J2EE can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines.

J2EE is chosen as the implementation paradigm to realise two prototype systems in Chapters 7 and 8. They are namely a Web-based prototype system for users to carry out visualisation-based design and manufacturing analysis, and a distributed co-design prototype system based on a J2EE infrastructure to enable a dispersed team to accomplish a collaborative feature-based design task, respectively. The underlying techniques for the J2EE paradigm, especially those used in this book, are briefly introduced here.

The J2EE, CORBA and .Net paradigms can be used to:

- Encapsulate and integrate existing legacy systems in product design, planning, simulation, execution, and distribution into an open, distributed intelligent environment via networks.
- Provide interfaces for designers and software systems to realise human/system interactions and interoperability of systems.
- Model special services in a distributed application, such as registration service, administration services, and communication facilitator, to coordinate product development processes and manage information and data.

A J2EE paradigm is composed of a series of disparate technologies that can be categorised as the component technologies, service technologies and communication technologies [Allamaraju, *et al.*, 2001].

Servlet and JSP (JavaServer Pages) are two Web-based component technologies that provide the server-side programming means to extend the functionality of a Web server to provide dynamic contents in HTML or XML. This is illustrated in Fig. 1.12.

Service technologies include the JDBC (Java DataBase Connectivity), JMS (Java Message Service), JCA (Java Connector Architecture), etc. These services are used to manage the access of databases, send and receive messages, integrate legacy applications, etc.

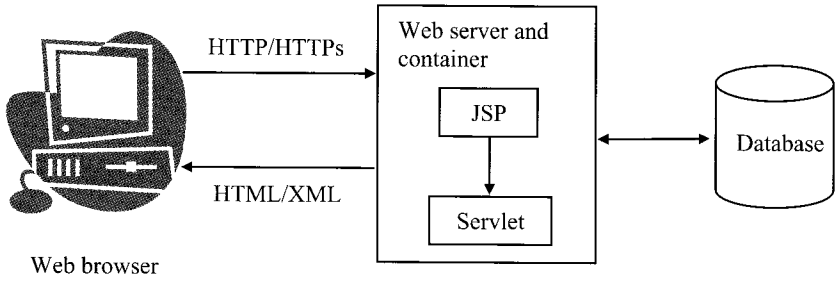


Fig. 1.12 Servlet- and JSP-based system architecture and the communication.

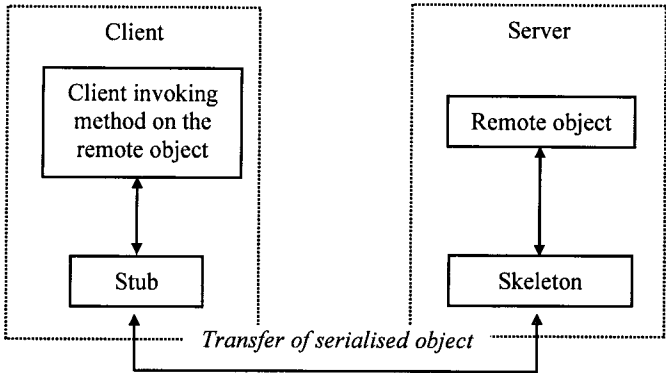


Fig. 1.13 A communication process based on RMI.

Communication components include the fundamental Internet protocols, such as HTTP and TCP/IP, and remote object protocols, such as RMI (Remote Method Invocation) and RMI-IIOP (RMI over Inter-ORB Protocol). TCP and IP are two fundamental transmission and Internet protocols, and they work together as a single entity to move data around the Internet. HTTP is a generic, stateless and application-level protocol that enables the multi-media exchange of information from a server to a client. Based on TCP/IP, RMI, which is one of the primary mechanisms in distributed object applications, allows object-to-object

communications between different systems. Based on RMI, applications can be enabled to call object methods located remotely and share data across systems. RMI defines interfaces of remote objects, and methods on these remote objects can be called through stub and skeleton mechanisms as if they are local. A RMI process is illustrated in Fig. 1.13. RMI-IIOP extends RMI to support CORBA mapping and invocation. With RMI-IIOP, a remote interface to any remote object implemented in any language can be defined and invoked.

1.3 Summary

With the extensive global competition and the rapid evolution of the Internet technologies, product development systems are moving towards supporting integrated and collaborative applications. ICE systems have been actively investigated to provide solutions to integrate geographically dispersed users, systems, resources and services, and enable them to cooperate and interoperate in an Internet/Intranet environment beyond the physical and temporal boundaries. This chapter presents an introduction of the motivations, concepts and objectives of establishing an ICE for product development. Some enabling technologies and their features, especially those used for system and methodology implementation in this book, are highlighted.