# Dynamic Task Planning in Web-based Product Design and Manufacturing Environments

BY

CHULHO CHUNG

A Thesis

Submitted to the Faculty of Graduate Studies

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Manufacturing Engineering

University of Manitoba

Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

**Dynamic Task Planning in Web-based Product Design and Manufacturing Environments**

BY

Chulho Chung

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

OF

DOCTOR OF PHILOSOPHY

# Abstract

This dissertation presents efficient approaches to dynamic task planning, and their implementation in a Web-based product design and manufacturing (PD&M) environment. The involvement and sharing of timely available resources form a key to achieving this planning. The following task-planning activities are focused on: (1) process (fabrication) planning at the shop floor level; (2) assembly planning at the production level; and (3) disassembly planning at the product end-life level.

Standard database (DB) technologies including relational DB (RDB) and Java DB connectivity (JDBC) are extensively applied to appropriately model, manage, and search various knowledge and resources, both of which are related to manufacturing and de-manufacturing (DM). Efficient DB search algorithms are also developed to minimize a number of transactions over the Internet, and to retrieve a reduced amount of data. The data are then embedded into the process of resource-involved task planning. In particular, object-oriented DBs (OODBs) are developed to provide an easy and effective manipulation of the retrieved resource data on the client. The OODBs are defined by the Java classes developed, consisting of encapsulated attributes and methods. The resource data formed with DBs are set to the attributes in each OODB, and the developed

supporting methods for dynamic task planning are mainly interfaced to the methods of the OODB.

A fast assembly-tool reasoning method is proposed for considering tool feasibility in complete assembly or disassembly planning. It is based on a parameterized assembly tool and a non-homogeneous global accessibility cone that approximates the obstacles of a fastener being assembled or disassembled. It avoids the use of complex collision-detection methods. The method can also simply deal with complex variations in a fastener movement and a tool access angle. As another critical issue in determining the priority of part assembly or disassembly with the tool feasibility, part disassemblability is analyzed via a method developed in this research. Representing topological part accessibility in a geographic coordinate system, a disassembly directionality chart is formed by collision detection algorithms and two-dimensional (2D) mapping processes developed. This is advantageous in executing Boolean operations to find an accessible direction for a part assembly or disassembly.

Several optimization methods are developed to generate optimal process details for an intended task, including the best set of tools and machines. These methods are based on the object-oriented programming (OOP) technology, the genetic algorithm (GA) and a heuristic search. For instance, a two-tier GA is applied for tool selection-embedded optimal assembly planning. It is executed on the constructed fastener-based assembly structure of a product, which is constructed via the tool feasibility and part disassemblability analyses. In this research, it helps to simultaneously optimize fastener-based assembly sequences and assembly tool sequences with a reduced number of generations or iterations. Moreover, a matrix-based heuristic approach developed

supports near-optimal selective disassembly (SD) sequence planning based on a dynamic DM environment. It is executed with primitive matrices that are generated from a given product solid model. For randomly selected parts, the developed approach generates an efficient and practical SD plan of a product. The SD plan defines a partial and incomplete disassembly sequence, minimizing SD efforts via forming modularized removals that satisfy assembly-tool feasibility and part disassemblability.

Web-based systems are utilized for the integration of dynamic task planning developed. They are built on neutral and standardised Web and DB technologies. The implemented Web-based systems work on a three-tier Internet-based architecture consisting of a Web server, a DB server and clients. It is a basic model to form a multi-tier Internet-based environment that is widely adopted in establishing a Web-based PD&M environment. In particular, an Internet-based visualization is accomplished by Java and virtual reality modelling language-external authoring interface (VRML-EAI). It is based on VRML-based static and dynamic interactions developed. This is advantageous in implementing a realistic three-dimensional (3D) simulation and acquiring real time information from a virtual object. The acquired information is also used for other computations including the estimation of machining time and required distance to move a part being disassembled. In the Web-based environment, the dynamic task planning systems subsequently achieve collaborative, cost-effective, platform independent, and visualized information sharing over the Internet.

# Acknowledgements

*And we know that in all things God works for the good of those who love him, who have been called according to his purpose* (Romans 8:28). I would like to thank God for the Ph.D. degree given to me first of all.

It is hard to find proper expressions to thank my great advisor, Dr. Peng. He is the greatest person who I have ever known in my life. He is more than a thesis advisor. He is my academic father and role model. I again express my appreciation and deep gratitude for his unprecedented support, guidance, advice, and belief in me. It was my privilege to be his graduate student and now I am so proud to be one of his graduates.

I wish to thank the rest of my committee members, Dr. Balakrishnan and Professor Clayton. They have always encouraged me with their academic enthusiasm and sincerity, and provided me with invaluable guidance and insightful comments throughout the course of my research.

My appreciation is sincerely expressed to my current and former colleagues working together in the Virtual Manufacturing Centre (VMC), names of a few, Dr. Chunsheng Yu, Xiumei Kang, Qing Niu, Hector Sanchez, Heewan Lee, Mahmud Ahsan, and Tao Luan.

My gratitude also goes to my Korean friends, Sungjun Kim, Hanyong Lee, and Dr. Hundal Jung at the University of British Columbia. Without their support and encouragement, it would have been impossible for me to obtain the degree.

I would like to express my gratitude to the University of Manitoba and Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support upon my research. I also wish to express my sincerest thanks to generous people who provided the fellowships, Edward R. Toporeck Graduate Fellowship and TransCanada Pipelines Graduate Fellowship in Engineering. These supports enabled me to concentrate more on schoolwork as well as research without having to worry about financial difficulties.

Finally, I would like to thank my family. Without them, it would have been impossible for me to obtain the degree. I am ever grateful to my wife, Hannah Roh, who supported me without any complaints during last four years. I appreciate her patience and trust towards me. I would like to thank my parent-in-law who supported our family through hard times. They deserve to be proud of their contribution to my Ph.D. degree. I also wish to express sincere love to my son, David, daughter, Catherine, and brother-in-law living in the USA, Kyungun Roh.

# List of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| Abbreviation | Description |
|---|---|
| AI | Artificial intelligence |
| AM | Agile manufacturing |
| AOR | Annual operation requirement |
| ASP | Active server page |
| BIT | Built-in-test |
| BOM | Bill of materials |
| B-rep | Boundary representation |
| BS | Bounding sphere |
| CAAP | Computer-aided assembly planning |
| CAD | Computer-aided design |
| CAPP | Computer-aided process planning |
| CATS | Computer-aided tool selection |
| CCW | Counter clockwise |
| CE | Concurrent engineering |
| CGI | Common gateway interface |
| CIM | Computer integrated manufacturing |
| CMM | Coordinate measuring machine |
| CNC | Computerized numerical control |
| COM | Common object model |
| CORBA | Common object request broker architecture |
| CPD | collaborative product design |
| CPU | Central processing unit |
| C-space | Configuration space |
| CVD | Chemical vapour deposition |
| CW | Clockwise |
| DB | Database |
| DDL | Data definition language |
| DFA | Design for assembly |
| DFD | Design for disassembly |
| DFE | Design for environment |

| Abbreviation | Description |
|---|---|
| DFLC | Design for life cycle |
| DFM | Design for manufacture |
| DFMa | Design for maintainability |
| DFQ | Design for quality |
| DFR | Design for recyclability |
| DFS | Depth first search |
| DFX | Design for X |
| DM | De-manufacturing |
| DML | Data manipulation language |
| DOF | Degree of freedom |
| DXF | Data exchange file |
| $D^3C$ | Digitized disassembly directionality chart |
| EAI | External authoring interface |
| EIDL | End item design life |
| ERD | Entity-relationship diagram |
| ERP | Enterprise resource planning |
| FATs | Fastener-axis action tools |
| FBAS | Fastener-based assembly structure |
| FSIs | Functional significant items |
| GA | Genetic algorithm |
| GAC | Global accessibility cone |
| $GAC^d$ | Global accessibility cone with depth |
| GNU | General public licence (free software foundation) |
| GPL | GNU public license |
| GT | Group technology |
| GUIs | Graphic user interfaces |
| HTML | Hypertext mark-up language |
| HTTP | Hypertext transmission protocol |
| HTTPS | Hypertext transmission protocol, secure |
| IDE | Integrated drive electronics |
| IDVS | Interactive design visualization system |
| IGES | Initial graphics exchange specification |
| I/O | Input/output |
| IP | Internet protocol |
| ISO | International standard organization |
| IT | Information technology |
| IT-number | ISO tolerance number |
| ITSS | Incorporated tool selection system |
| JDBC | Java database connectivity |
| JSP | Java server page |
| JVM | Java virtual machine |
| LAC | Lowest average cost |
| LACP | Least aggregate cost and process time |
| LAP | Least average process time |

| Abbreviation | Description |
|---|---|
| LMEs | Large manufacturing enterprises |
| MIME | Multipurpose Internet mail extension |
| MTBF | Mean time between failures |
| MTTR | Mean time to repair for corrective maintenance |
| MV | Machining volume |
| NC | Numerical control |
| NP | Non-deterministic polynomial |
| OBB | Oriented bounding box |
| ODBC | Open database connectivity |
| OODBs | Object-oriented databases |
| OOMFs | Object-oriented manufacturing features |
| OOP | Object-oriented programming |
| PC | Personal computer |
| PDM | Product data management |
| PD&M | product design and manufacturing |
| PHP | Hypertext pre-processor |
| PLCC | Product life-cycle cost |
| PVD | Physical vapour deposition |
| RAM | Read access memory |
| RDB | Relational database |
| RDBMS | Relational database management system |
| SD | Selective disassembly |
| SMEs | Small to medium enterprises |
| SQL | Structured query language |
| SSIs | Structural significant items |
| STEP | Standard for the exchange of product data |
| TATs | Tool-axis action tools |
| UML | Unified modelling language |
| VEs | Virtual enterprises |
| VM | Virtual manufacturing |
| VP | Virtual prototyping |
| VR | Virtual reality |
| VRML | Virtual reality modelling language |
| VRML-EAI | Virtual reality modelling language-external authoring interface |
| WP | Wave propagation |
| WPMT | Web-based product modelling tool |
| WWW | World wide web |
| XML | Extensible mark-up language |
| X3D | Extensible 3D |
| 2D | Two-dimensional |
| 3D | Three-dimensional |

# List of Notations

| Variables and symbols used for dynamic process planning in Chapter 3 | |
|---|---|
| **Variable or Symbol** | **Description** |
| AOR | Annual operational requirement (hr) |
| $\gamma$ | Annual preventive maintenance cost including servicing, inspection, calibration |
| $C_o$ | Machine capital cost |
| $C_m$ | Normalized machine cost based on machine capital cost and maintenance cost |
| $d_{ij}$ | Dimensional tolerance between manufacturing faces $f_i$ and $f_j$ |
| EIDL | End item design life |
| $\mathrm{ET}_j$ | Elapsed time of the $j$-th preventive maintenance task for the $i$-th failure mode |
| $F_i$, $F_j$ | $i$-th and $j$-th manufacturing features |
| $f_i$, $f_j$ | $i$-th and $j$-th manufacturing faces |
| $m$ | Number of relative tolerances |
| MTBF | Mean time between failures |
| MTTR | Mean time to repair for corrective maintenance |
| $n$ | Number of manufacturing faces in a part to be machined |
| $P_{cm}$ | Corrective maintenance cost rate per hour, including payment and spare parts |
| $P_{pm}$ | preventive maintenance cost rate per hour, including payment and supplies |
| $q$ | Number of failure modes in a machine |
| $R_a$ | Roughness average |
| $r$ | Number of preventive maintenance tasks in a failure mode |
| $s_i$ | Surface roughness given to a manufacturing face $f_i$ |
| $\mathrm{TF}_j$ | Task frequency of the $j$-th preventive maintenance task for the $i$-th failure mode |

**Variables and symbols related to dynamic assembly and disassembly planning in Chapters 4, 5, 6 and 7**

| Variable or Symbol | Description |
|---|---|
| $\mathcal{DC}$ | Disassembly chain |
| $\mathcal{DF}$ | Family set of fasteners corresponding to a $\mathcal{DS}$ |
| $\mathcal{DS}$ | Selective-disassembly sequence (a family set of parts) |
| $e_a$ | Height of a tool head |
| $e_b$ | Height of a tool end |
| $e_e$ | Height of a tool extension |
| $e_f$ | Displacement in the fastener removal direction (y-axis) |
| $e_h$ | Displacement in the opposite direction of a tool access (xz-plane) |
| $e_n$ | Displacement in the normal direction of a tool access (xz-plane) |
| $f$ | Fitness function |
| $\boldsymbol{F}^a$ | Set of accessible fasteners |
| $F_i$ | Fastener $i$ in a product |
| $\boldsymbol{FP}$ | Fastener-part connectivity matrix ($m$×$n$) |
| $h$ | Height of a tool handle |
| $l_e$ | Effective length of a tool handle |
| $l_f$ | Total length of a fastener |
| $\boldsymbol{M}$ | Set of fasteners in a product, $M$={1, 2, …, $m$} |
| $m$ | Number of fasteners in a product |
| $N$ | Set of parts in a product, $N$={1, 2, …, $n$} |
| $n$ | Number of parts in a product |
| $\boldsymbol{n}$ | Normal vector on a triangle patch, $\boldsymbol{n} = n_x\hat{\boldsymbol{i}} + n_y\hat{\boldsymbol{j}} + n_z\hat{\boldsymbol{k}}$ |
| $\boldsymbol{n}_t$ | Tool rotation axis |
| $\boldsymbol{n}_f$ | Fastener rotation axis |
| $\boldsymbol{O}$ | Original point |
| $\boldsymbol{PA}$ | Part accessibility matrix ($n$×$n$) |
| $\boldsymbol{PA}^{i,j}$ | Part accessibility of part $i$ to its adjacent part $j$ |
| $\boldsymbol{PD}$ | Part disassembly-route matrix ($n$×$n$) |
| $P_i$ | Part $i$ in a product |
| $\boldsymbol{Ps}$ | Set of selected parts for separation |
| $R(\theta, \varphi)$ | Defined depth towards the direction at a pixel ($\theta$,$\varphi$) of a GAC[d] |
| $r_a$ | Radius of a tool head |
| $r_b$ | Radius of a tool end |
| $r_e$ | Radius of a tool extension |
| $T_k^i$ | A triangle patch of part $i$ |
| $\hat{T}(\theta, \varphi)$ | Unit vector on a pixel ($\theta$,$\varphi$) |
| $V_i$ | Vertex on a triangle patch, $V_i = x\hat{\boldsymbol{i}} + y\hat{\boldsymbol{j}} + z\hat{\boldsymbol{k}}$ |
| $w$ | Width of a tool handle |
| $\boldsymbol{X}$ | Intersection point on a triangle patch |
| $\alpha_{\max}$ | Maximum tool access angle |

| **Variable or Symbol** | **Description** |
|---|---|
| $\alpha_{\min}$ | Minimum tool access angle |
| $\beta$ | Minimum tool-application angle |
| $\varphi, \varphi^*$ | Colatitude and latitude angle |
| $\theta$ | Longitude angle |
| $\theta_S$ | Starting position to search a required feasible range along the longitude direction |
| $\Delta\theta$ | Converted minimum tool-application angle |
| $\Delta i$ | Topological disassemblability of a part $P_i$ |
| $\Lambda$ | Feasibility of an articulated device |

# Chapter 1

# Introduction

This chapter presents an introduction and background of the research on dynamic task planning in Web-based product design and manufacturing (PD&M) environments. The importance of task planning is first described, and the demand for dynamic task planning to support the Web-based PD&M is discussed. Research motivation, objectives and contributions are also presented in this chapter.

The reminder of this chapter is organized as follows. Section 1.1 presents the background of the research. Research motivation and objectives are presented in Sections 1.2 and 1.3, respectively. The research contributions are listed in Section 1.4. In Sections 1.5 and 1.6, the overview and special terms of the dissertation are respectively presented, and the summary is then presented at the last section of this chapter.

## 1.1 Background

Task planning defines the details of economical and optimal means for successfully accomplishing intended tasks of various product life-cycle disciplines. The disciplines may include manufacturing, distribution, service, and de-manufacturing (DM) such as product and material recycling, and disposal. For instance, the task planning for manufacturing includes process (fabrication) planning and assembly planning. It generates the manufacturing information of processes, procedures, and required resources including tools and equipment for product manufacture at the shop floor and assembly levels. The generated information is used for design of manufacturing facilities and systems, resource procurement and scheduling, and production planning and control.

Currently, engineers are facing emerging challenges. A fast-moving marketplace rapidly diminishes the value of new technologies or products while demanding a variety of highly customized products in small volumes. The success of a product depends on not only the cost, quality, design innovation, delivery and services, but also the environmental concerns such as recycling and disposal. It is more and more critical to shorten the *time-to-market* of high quality products at low cost while maintaining a customer-focused environment and attention to the product life cycle. *Agility* is regarded as one of the most important attributes of PD&M systems to help enterprises survive in the currently competitive marketplace.

Based on advances in computer technology and the Internet, Web-based PD&M has been identified as an enabling technology of agile manufacturing (AM) and its related activities. Much of the research has been oriented towards developing Web-based PD&M systems for: (1) supporting various disciplines at different product life-cycle stages including marketing, product development, production, product use and product end life, and (2) incorporating the product life-cycle disciplines into a globally networked environment where dispersed business partners cooperate. The main thrusts of Web-based PD&M are concurrency, attention to the product life cycle, collaboration, high-fidelity simulation and validation, and integration with other Internet-based business systems. They form the key to enhancing the agility in the $21^{st}$ century business environment (Davis *et al.*, 2004; Xu *et al.*, 2003). Two promising paradigms support Web-based PD&M effectively, which are *Web-based collaborative product design (CPD)* and *Web-based virtual manufacturing (VM)*. Both paradigms aim to shorten *time-to-market* by significantly minimizing uncertainty issues beforehand. The uncertainty issues are related to various disciplines occurring at the latter stages of the product life cycle.

In Web-based PD&M environments, the task planning helps to minimize the *uncertainty issues* and to shorten *time-to-market*. It works between design and other product life-cycle disciplines by reciprocally performing: (1) the reliable assessment of product design alternatives in a discipline point of view, and (2) the generation of vital information for realizing the disciplines. For instance, a product design alternative is reliably evaluated in terms of various *–abilities* including manufacturability, assemblability, serviceability and recyclability. A task plan generated for a product being designed can be used to either determine feasibility or calculate reliable time and cost in completing the intended task. A comparison study of several design alternatives can be executed based on the calculated time and cost. If the task plan is not feasible to be used in an existing product life-cycle environment, a huge amount of time and cost may have to be committed to the re-planning or even re-designing of a product. Therefore, the task planning is a crucial process in product development.

Similarly, the generated details from task planning can be further used for: (1) *marketing* to do exploration of business opportunities for new products; (2) *manufacturing* to do such activities as design of manufacturing facilities and systems, resource procurement and scheduling, outsourcing, and production planning and control; (3) *distribution* to develop resources and procedures for product packaging, handling, storage and transportation; (4) *service* to develop processes for procuring spare parts, and equipment for performing product maintenance; and (5) *DM* to do such activities as exploration of the most efficient recycling and disposal of a product, development of resources and procedures for DM, and design of DM facilities and systems.

## 1.2 Motivation

Major efforts have focused on developing efficient approaches to task planning, leading to various computerized systems for generating reasonable and consistent task plans. However, the task-planning systems developed are not fully satisfactory in supporting Web-based PD&M because of: (1) islands of automation or expertise syndrome, and (2) uncertainty caused by disregarding resources in a real manufacturing (RM) or DM environment. These issues lead to diminished organization agility and increased time-to-market.

There is a need for dynamic task planning that is performed based on real product life-cycle environments. The involvement of timely available resources including equipment and tools is critical in this planning. This works in examining a feasible and economical task plan for a product being designed. In addition, the resource-involved task planning should be performed in a Web-based PD&M environment that allows resource sharing within geographically dispersed business partners.

A feasible and economical task plan can be effectively generated by using the dynamic task planning systems employed in a Web-based PD&M environment. The details generated can be used for: (1) exploiting business opportunities in a virtually formed enterprise alignment; (2) executing reliable assessment of product design alternatives in a discipline point of view; and (3) realizing disciplines occurring at the latter stages of the product life cycle.

In particular, the reliable assessment of product design alternatives is efficiently achieved, as dynamic task planning supports *design for X* (DFX) systems for Web-based CPD. The currently developed DFX systems may include (Kuo *et al.*, 2001): design for manufacture (DFM), design for assembly (DFA), design for disassembly (DFD), design for recyclability (DFR), design for environment (DFE), design for life-cycle (DFLC), design for quality (DFQ), design for maintainability (DFMa), and design for reliability. This demand is an on-going issue because of the limitations of the DFX systems. The limitations (Dereli, 1998; Huang and Mak, 1999; La *et al.*, 2003; Zha, 2005; Zhao and Shah, 2005) can be summarized as follows: (1) re-design suggestions based on *rules-of-thumb*; (2) too much user-interaction to analyze the manufacturing time and cost; (3) unreliable time and cost estimation; and (4) disregard of manufacturing resources and processes.

They result from the nature of the current DFX systems that rely on simplified approaches based on knowledge, experience, rules and guidelines. Although the approaches help in executing evaluations and re-design suggestions for design alternatives quickly and easily, there remain uncertainty issues with respect to such disciplines as manufacturing, assembly, service, and DM. In a Web-based PD&M environment, the remaining uncertainties lead to increased product life-cycle cost (PLCC) and product development-cycle time. Based on feasible and economical task plans generated by dynamic task planning, it is possible for DFX systems to execute the practical assessment of various *–abilities* for a product design alternative, and the reliable estimation of related time and cost. This allows for better decision-making with reduced uncertainties, and consequently reduced product development-cycle time.

Similarly, the discipline realization can be efficiently done by the aid of Web-based VM that is supported by dynamic task planning. Web-based VM is based on high fidelity simulation and validation to minimize the uncertainty issues associated with a product life cycle discipline. This process is required to deal with a huge number of *what-if* scenarios for a product being designed. A *what-if* scenario can be associated with such elements as business partners, manufacturing operations and systems, machines, tools and other resources. Moreover, the number of *what-if* scenarios will significantly increase when various product design alternatives and all the life-cycle disciplines are concurrently considered until a product design is finally frozen.

For examining every *what-if* scenario, Web-based VM systems execute computer-intensive modelling and simulation, which is a time-consuming process. This can lead to increased process time in examining all the possible *what-if* scenarios. In addition, if not given the detailed information (*i.e.*, required processes, machines, tools, *etc.*) of a *what-if* scenario, or no scenario is given to a new product being designed or considered, the simulation and validation process should rely on an examiner's knowledge, experience or preference. This may lead to remaining uncertainty issues to the product design that is eventually frozen. A reliable task plan generated by dynamic task planning helps to create a reduced number of feasible *what-if* scenarios. Only the workable ones are precisely examined via Web-based VM systems. Consequently, this can enhance the efficiency of the high fidelity simulation and validation, which solves complicated problems in order to prevent costly mistakes before a product design alternative is employed.

## 1.3 Research Objectives

The objective of this research is to develop efficient approaches to dynamic task planning, and to integrate them into a Web-based PD&M environment. Based on the sharing and involvement of timly available resources in task planning, these approaches aim to generate the details of feasible and optimal means for the intended tasks that belong to each discipline at different product life-cycle stages. The notion of dynamic task planning is simply illustrated in Figure 1.1.

As shown in Figure 1.1, a product design alternative is given to a dynamic task-planning system in a Web-based PD&M environment. This design alternative is considered to a specific discipline at product life-cycle stages. This discipline can be executed at various business partners geographically dispersed. Using the task-planning system, a planner chooses a business partner to generate optimal and feasible discipline details. The details are based on the chosen partner's resources that are used to complete the intended tasks economically and successfully. Subsequently, the detailed information including optimal procedures and the required resources serves to execute further processes via DFX, or VM systems in the Web-based PD&M environment.

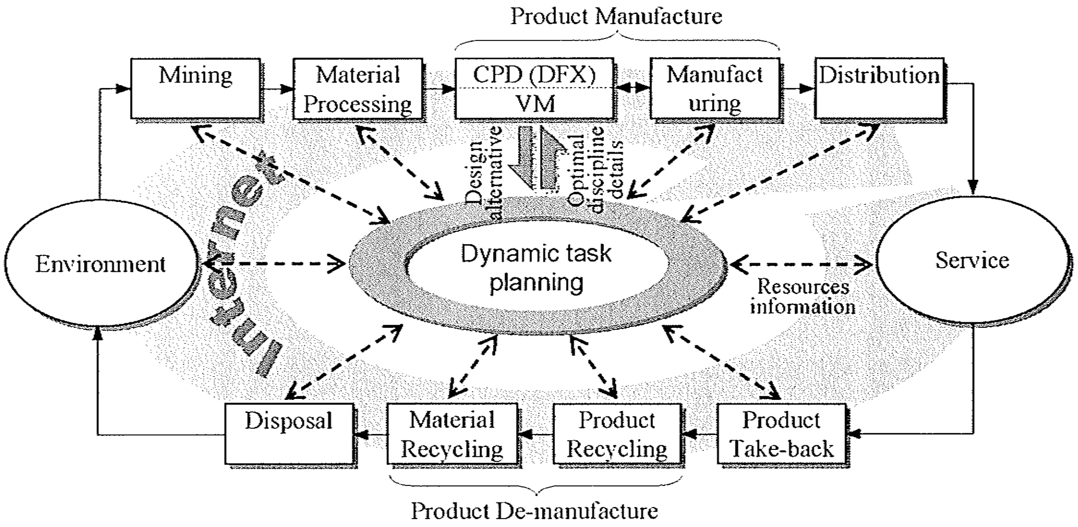


**Figure 1.1**: Notion applied in the research on dynamic task planning in a Web-based PD&M environment.

The requirements of the approaches to be developed are as follows: (1) *generic*, to deal with various products manufactured, from small to medium enterprises (SMEs) to large manufacturing enterprises (LMEs); (2) *dynamic*, to respond in a timely manner to a product life-cycle environment; (3) *rapid*, to synchronise with Web-based DFX or VM systems dealing with various *what-if* scenarios; and (4) *precise*, to generate the task plans that minimize potential errors with respect to product life cycle disciplines.

In addition, the developed approaches for dynamic task planning are implemented as Web-based systems that can be employed in a Web-based PD&M environment. It is required that the systems should be built on currently available neutral and standardised Web and database (DB) technologies. These technologies are used to integrate the developed systems into an existing Web-based PD&M environment, while providing the cost-effective, platform-independent and visualized information sharing over the Internet.

## 1.4 Research Contributions

This research contributes to the fields of task optimization, collision detection, geometric accessibility analysis, engineering knowledge and resource management, and three-dimensional (3D) visualization on the Internet. The following are the list of these contributions:

(1) A fast assembly-tool reasoning method based on a geometric accessibility analysis;

(2) A topological part disassemblability analysis via several triangle patch-based collision detection techniques;

(3) A two-tier genetic algorithm (GA)-based approach to dynamic assembly planning with a number of tool alternatives;

(4) A matrix-based heuristic approach to near-optimal selective-disassembly (SD) sequence planning;

(5) The extensive use of relational database (RDB) and Java database connectivity (JDBC) technologies in modeling, managing and searching knowledge and resources;

(6) The efficient use of the object-oriented programming (OOP) paradigm in defining object-oriented databases (OODBs) to appropriately manipulate data retrieved on the client; and

(7) Web-based Interactions of Java and virtual reality modelling language-external authoring interface (VRML-EAI).

## 1.5 Dissertation Overview

In this dissertation, each chapter commonly consists of an introduction, a main body and a summary. The introduction provides audiences with the overview of each chapter before discussing matters in detail. The discussed matters are later summarized at the end of each chapter.

The remainder of this dissertation is organized in the following manner. Chapter 2 reviews literature in the related fields that include process planning, assembly planning, and disassembly planning. Technologies related to Web-based PD&M are investigated to draw a technical direction by which task planning approaches to be developed are efficiently implemented in a Web-based environment. The problem statement, and research strategy and hypothesis are also presented in this chapter.

Chapter 3 presents an approach to dynamic process (fabrication) planning at the shop floor level. In addition to current research areas such as manufacturing feature recognition, set-up planning and sequencing, and machining process determination, the selection of tools and machines at the dynamic shop floor is embedded into process planning. DB search algorithms and selection knowledge are developed to efficiently retrieve tool and machine alternatives from a DB. Developed production cost and time models are then used to determine the best set of tools and machines via the OOP technology. To demonstrate the efficiency of the proposed approach, several application examples are also provided at the end of this chapter.

Chapter 4 presents a fast assembly-tool reasoning method based on a geometric accessibility analysis. Given solid models of product parts and fasteners, the goal is to quickly determine whether an assembly tool is usable in assembling or disassembling a fastener. This novel method is based on a parameterized assembly tool and a global accessibility cone with depth ($GAC^d$) that approximates the obstacles surrounding the fastener being assembled or disassembled. This method is advantageous in determining the assembly-tool feasibility that is a critical consideration in both dynamic assembly and disassembly planning. In addition, an assembly tool DB is modelled and implemented to link the developed method with a dynamic manufacturing environment. In this chapter, subsequently, application examples are provided to illustrate the performance of analyzing assembly-tool feasibility via the developed method.

Chapter 5 presents a topological disassemblability analysis of parts. Two primitive matrices are introduced for supporting the part disassemblability analysis. These matrices represent topological constraints of parts, also known as topological accessibility of parts, and parts-fasteners connectivity. A systematic method is also proposed to automatically construct these matrices from computer-aided design (CAD) solid models. The methods are based on various collision detection techniques that are developed in this research.

Chapter 6 presents a GA-based approach to dynamic assembly planning at the production level. This approach is based on the assembly-tool feasibility analysis and the part disassemblability analysis, which are presented in Chapters 4 and 5, respectively. A number of tool alternatives are embedded into the assembly planning process for a product. Outcomes of this process are the best set of assembly tools and an optimized assembly plan that satisfies both topological constraints of components and tool feasibility. At the end of this chapter, application examples are provided to show how this approach achieves dynamic assembly planning appropriately.

Chapter 7 presents a matrix-based heuristic approach to non-destructive SD planning at the product end-life level. The approach is also based on the assembly-tool feasibility analysis and the part disassemblability analysis, which are presented in Chapters 4 and 5, respectively. For supporting the design process at the early stage of product development, this chapter also discusses a de-manufacturability analysis based on the proposed approach. In particular, the approach generates a near-optimal SD sequence based on a dynamic DM environment. It is reasonable to find a *good* feasible solution obtained by less expensive computation because SD sequence planning is mathematically a non-deterministic polynomial (NP)-hard problem. A NP-hard problem is difficult to solve with a known efficient algorithm, and it is quite unlikely that one exists. To demonstrate the efficiency of the proposed approach, several application examples are also provided at the end of this chapter.

In Chapter 8, the proposed approaches for dynamic task planning are implemented as Web-based systems in a three-tier Internet environment. The systems are developed via VRML-EAI, Java and JDBC. In establishing a three-tier Internet environment, a Web server and a DB server adopt Apache hypertext transmission protocol (HTTP) server version 1.3, and MySQL DB server version 3.23, respectively. The use of the Web-based systems in a multi-tier Internet environment is advantageous for collaborative, distributed and globally dispersed PD&M. This chapter includes the methodology to implement the Web-based systems, their structures, functions and graphic user interfaces (GUIs) in detail. Subsequently, Chapter 9 concludes the dissertation. The summary of research contributions and possible extensions of this work are also discussed in this chapter.

## 1.6 Special Terms

Through out this dissertation, a variable or symbol in **bold** character-type represents either a set or a vector. The *italic* character-type is frequently used for the expressions of symbols and variables, equations, algorithms, emphasised terms in a context, and Latin origins including *et al.* This character-type is also used to express peculiar terms developed by the author. The terms mainly include Java classes and their contents developed through this research. In Chapter 8, the Century Gothic font is used for explaining the structure of developed Java program codes that are inserted in the context of the chapter.

## 1.7 Chapter Summary

This chapter presented the importance of Web-based PD&M in the 21st century business environment, the related problems of current task planning to support Web-based PD&M, and the notion of dynamic task planning. Dynamic task planning is efficient in not only solving the existing problems of current task planning, but also enhancing PD&M efficiency in Web-based PD&M environments. Resource sharing and involvement were emphasized to achieve dynamic task planning on the Internet. In the next chapter, a review of the related literature is provided, leading to a research hypothesis and strategy.

# Chapter 2

# Literature Review and Research Strategy

This chapter reviews literature in: (1) process (fabrication) planning at the shop floor level; (2) assembly planning at the production level; and (3) disassembly planning at the product end-life level. The three topics chosen mainly require the involvement of manufacturing and DM resources in executing task planning. They can constitute major planning activities at product life-cycle stages, appropriately corresponding to currently complex customers' demands (Bras, 2004; Cui and Forssberg, 2003; Kuo, 2000; Nagalingam and Lin, 1999; Singh, 1996). The demands for a product are described as low cost, high quality, high customization, low volume, right delivery and services, and environmental satisfactions regarding recycling and disposal.

The focus of the review is the involvement of resources such as machines and tools in task planning. In addition, various technologies employed in developing Web-based PD&M systems are examined with a view to finding ways to effectively achieve research objectives. Based on the literature review, the problems addressed are summarized in the problem statement, leading to the research strategy proposed at the latter section of this chapter. The research hypothesis is also presented.

## 2.1 Process Planning

Generally, the aim of process planning is to define details of processes required to fabricate a product according to its specifications and available manufacturing resources (Peng *et al*., 2000). Process planning has played an important role in the link between design and manufacturing. It includes a variety of activities such as interpretation of design data, selection and sequencing of machining processes, selection of machine tools and cutting tools, determination of cutting parameters, choice of jigs and fixtures, and calculation of production time and cost.

Due to the dependence on a process planner's expertise and knowledge in executing process planning, computer-aided process planning (CAPP) systems have been introduced in industry. The CAPP systems can not only generate consistent and reasonable process plans but also interface with CAD systems in a computer integrated manufacturing (CIM) environment. Investigation shows that an efficient CAPP system could result in a total reduction of the production cost by up to 30% and time in a manufacturing cycle could also be reduced by up to 50% (Ahmad *et al*., 2001). For this reason, CAPP has been a popular research area in the last two decades.

Hundreds of CAPP systems have been developed, and currently available CAPP systems are extensively examined by Ahmad *et al*. (2001). From the viewpoint of plan generation principles, CAPP systems can be classified into two categories: variant and generative CAPP. The variant approach is based on the concept that similar parts will have similar process plans. Group technology (GT) has been used to store and retrieve standard process plans that will be edited to suit the planning requirements for new similar parts. This approach is effective in most batch manufacturing industries, where similar parts are produced repetitively. However, with the increasing demands for customisation of products and the increasing use of automatic manufacturing systems, a fast detailed, flexible and automatic process planning system is required. The generative approach seems to be the most promising one.

Generative CAPP systems use manufacturing knowledge to generate process plans. This is advantageous for new product development, but it relies on the development of artificial intelligence (AI), fuzzy logic, knowledge base and neural network technologies. Most generative CAPP systems have various functions to aid process planning activities including selection of machining processes, sequencing of machining operations, selection of machine tools and cutting tools, selection of machining parameters, and estimation of time and cost.

Although tremendous efforts have been made in developing CAPP systems, the effectiveness of these systems is still not fully satisfactory. Some efforts lead to increased cost of CAPP systems and reduced system flexibility. These are not efficient to respond to rapid changes in manufacturing environments cost-effectively and instantly.

The CAPP systems requiring huge amount of cost in application development and infrastructure are financially beyond the reach of most SMEs. For instance, Koelsch (1994) believes that CAPP systems are available only for companies with revenues in excess of $50 million.

In addition, the reduced system flexibility results in a poor interface of CAPP systems with the dynamic nature of manufacturing environments (Ahmad *et al.*, 2001). Because of the poor responses of current static CAPP systems to real-time shop-floor circumstances, many manufacturing problems may occur. Investigations have shown that 20-30% of all process plans are not valid and have to be altered when production starts (Usher and Fernandes 1999). One of the related issues is the dynamic selection of tools and machines in process planning. It is frequently used to examine an economical and feasible process plan for both planning and scheduling with resources at a shop floor taken into account.

Since automated approaches to selecting tools in process planning appeared in the late 1970s and early 1980s, many computer-aided tool selection (CATS) systems have been developed for CAPP systems. The CATS systems were developed using various methods: analytical, heuristic and a combination of both. An approach, in particular, the object-oriented tools selection system, has been introduced in CATS. For instance, Soromaz and Khoshknevis (1997) developed a system that selects tools based on object-oriented knowledge. In their research, tool types are selected based upon general guideline rules. Constraints imposed by machines and operations eliminate a few of the tool types. They addressed the need for *ideal tool selection*, but failed to clearly explain how this is done.

Usher and Fernandes (1999) suggested a systematic method for identifying and ranking tool alternatives based upon production cost and time. The authors developed production time and cost factors to rank tools, and used the object-oriented approach to manipulate a large number of alternative tool sets. Although they suggested a promising guide to dynamic tools selection, they did not propose any method to efficiently manage knowledge or tool information. In dynamic tools selection, the efficient organization of knowledge and the tool management are essential in forming a dynamic manufacturing environment. They also constitute a key to combining other process planning activities including selection of machining processes, sequencing of machining operations, and selection of machines.

Fernandes and Raja (2000) developed the incorporated tool selection system (ITSS) including five steps: alternative tooling, compatibility, logic elimination, determination of tooling parameters and DB search. The authors presented a method to sequentially determine tool types via object-oriented knowledge and DB search. However, the topic discussed is not combined with machines selection. The authors also fail to clearly explain the structure of the DB used and ITSS, both of which should be incorporated into a CAPP system.

Subrahmanyam *et al.* (1999) proposed a systematic approach to the development of a tool management system for supporting dynamic tools selection. They estimated that the developed tool management system could significantly reduce the down time of machines because of unavailability of the appropriate tools. Moreover, the authors emphasized the relevance of RDB in not only modelling tools and related manufacturing knowledge, but also accessing and manipulating information. However, the constructed data model in their research was not structured and organized although a widely used relational database management system (RDBMS), ORACLE V7.1, was applied to implement their notion.

Compared to dynamic tools selection, very few researchers have addressed the aspect of dynamic machines selection involved in CAPP. Instead, machines selection has been a major activity of job scheduling that is defined as the allocation of resources over time to perform a collection of tasks (Cao *et al.*, 2005; Subramaniam *et al.*, 2000). The objective of job scheduling is to assign specific tasks to specific machines in order to balance load distribution among different machines so that the available machines can be effectively utilized (Kumar and Rajotia, 2003). Current scheduling is done separately after a CAPP system generates a static process plan via incorporated knowledge. The generated plan does not consider machine capacity and the status of a shop floor. It may lead to a process plan either not feasible or sub-optimal from the scheduling point of view (Kumar and Rajotia, 2003).

Currently, several attempts have been made towards integrating process planning and scheduling for efficient dynamic machines selection. As an approach, all available machines and possible operations can be involved in the process of machines selection for all part types (Moon *et al.*, 2002; Moon and Seo, 2005). The mathematical model used in their research represents machine visiting sequences for the part types so that the total production time for the production order is minimized and workloads among machines are balanced. GA was then applied in searching the best set of machines and process plans. However, this computation is expensive in dealing with all the combinations of the used machines and operations. In a Web-based PD&M environment, the issue such as the immediate manufacturability analysis for the modification of a design alternative is hardly supported by this approach. In addition, the authors assumed that all machines are always available so that it is unlikely to achieve machines selection based on a dynamic shop floor.

Subramaniam *et al.* (2000) undertook research into the machines selection at a dynamic shop floor. In their research, three selection rules were suggested to achieve economical production, which are lowest average cost (LAC), least average process time (LAP), and least aggregate cost and process time (LACP). As performance measures to select machines at a dynamic shop floor, LAC, LAP and LACP help to quickly select machines to be used for processing a part. However, they only focused on dynamic machines selection without considering process planning. In particular, the authors mentioned machine reliability and maintainability to characterize machine cost related to maintenance or machine downtime, but did not incorporate this notion into their criteria for the machines selection.

A more dedicated performance measure was suggested by Kumar and Rajotia (2003). This performance measure includes machine capital, operation cost and cycle time. The authors then suggested the efficient steps to select machines on a dynamic shop floor. This approach is advantageous in combining dynamic machines selection to process planning. However, a machine under breakdown or maintenance is not considered for job assignment, and the machine breakdown or maintenance was also not quantified into the performance measure the authors suggested.

## 2.2 Assembly Planning

Assembly planning defines sequences of product assembly, by which initially separated parts are gathered to form a complete product. An optimal sequence results in less fixturing, less tooling, and more reliable operations during the assembly of a product. The assembly sequence plays a major role in aiding production control, assembly line design, and scheduling activities because it can provide useful manufacturing information and impose constraints in the selection of production equipment and alternative routines (Yin *et al.*, 2003). Therefore, the assembly planning is a crucial process through all stages of product development under concurrent engineering (CE) or CPD (Choi *et al.*, 2002; Noh *et al.*, 2004).

In the last decade, many computer-aided assembly planning (CAAP) systems have been developed to efficiently generate optimal assembly sequences. The use of robotic assembly tools allows an automated and computerized assembly planning integrated with CAD systems (Yin *et al.*, 2003). Initial assembly planning attempted was based on the geometric constraints of parts in a product. However, this approach runs on exponential time with the number of assembly components. Moreover, its computation time dramatically increases in searching for an optimal assembly sequence, as a collision detection with complex parts is utilized to automatically determine the assembling priority among them. It subsequently leads to considering other non-geometric constraints in assembly planning. One of the attempts is the fastener-based approach to assembly planning.

The fastener-based approach allows for avoiding expensive computation for a complex product consisting of many components. It can be used for easily defining a subassembly that consists of several parts connected with the same fasteners. Based on this approach, a product is simply transformed to the hierarchical fastener-based assembly structure (FBAS) for assembly planning. However, it is necessary for the fastener-based approach to consider the feasibility of assembly tools during the assembly process. It includes evaluating assemblability in terms of assembly-tool feasibility, and selecting a proper tool for economical assembly.

There have been many attempts to embed tools selection into assembly planning. In the research performed by Homem De Mello and Sanderson (1991), attachments were included in a relational model of assemblies for fastening operations. Although the attachments can be used to plan an assembly sequence, the detailed tool applications required to remove these attachments were not modelled in their work. Miller and Hoffman (1989) described a system that requires access space available for a fastener removal. However, simple tests consisting of ray casting and box testing were used to roughly distinguish between feasible tool applications and those that are not feasible.

The most detailed work was examined by Wilson (1998). The author developed a tool representation that includes the tool use volume, the minimum space that should be free in an assembly operation to apply a tool. In his research, the author also mentioned the notion of fast tool reasoning that allows the selection of a tool from a set of possible tool alternatives to execute an assembly operation. However, this notion was not included in his approach. Moreover, the tool reasoning method suggested by the author requires the computation of the configuration space (C-space) representation of obstacles. The computation is too expensive to deal with a large number of possible tool sets for a fastener. The tool sets are retrieved from an assembly-tool DB. This makes it impossible to achieve tool selection-embedded assembly planning.

Gupta *et al.* (1998) proposed a method to avoid the expensive computation of the C-space. With the articulated tool representation, the method combines collision detection methods and randomized via-point path planners. However, the method was also based on the same assumption used by Wilson (1998), where tool *place-constraints* and *use-volumes* are not variable during a tool application. Because of the limitations of their methods to deal with variations in a tool application, the feasibility of an assembly tool with one degree of freedom (DOF) was only discussed in their research.

Recently, Lazzerini and Marcelloni (2000) discussed the importance of a number of assembly tools applied and tool changes in achieving optimal assembly planning. However, the authors did not incorporate this notion into their work. Researchers such as Kuo (2000), Tseng and Li (1999), and Yin *et al.* (2003) focused on fasteners in assembly or disassembly planning, but they also failed to take assembly tools into account in their work; the tools may or may not be appropriate for the fasteners in a given part configuration.

In the research performed by Léon *et al.* (2001), a fast sequence generator for assembly or disassembly was proposed to support the early design process of product development. Although the authors discussed the significance of assembly tooling in generating an efficient assembly or disassembly sequence, no tool feasibility test was executed in their work. Tseng *et al.* (2004) incorporated a number of tool changes into their objective function used to plan an optimal assembly sequence. In their research, however, the authors only used four types of generalized tools that are simply classified by the applied force magnitude.

To achieve dynamic assembly planning in this research, a modified global accessibility cone (GAC) rather than C-spaces or collision detections is adopted to make assembly-tool reasoning computationally inexpensive. The modified GAC is used to simply represent a geometric configuration generated by obstacles or components surrounding a fastener.

A GAC was originally used to represent available angles of attack towards an object. The concept of GACs has been extensively investigated, and led to promising results for the accessibility analysis of tools used in many other applications such as multi-axis machining, laser scanning and coordinate measuring machines (CMM). The notion of feature accessibility and technique for creating GACs were first proposed by Spyridi and Requicha (1990). In their research, GACs were used to represent likely angles of attack for a probe approaching an object. However, this method cannot generate GACs for a general surface because of the computation complexity of their algorithm called the Minkowski algorithm.

To eliminate the need for the complex computations of the Minkowski algorithm, approaches to create GACs at single points were suggested by many researchers (Jackman and Park, 1998; Lim and Menq, 1994; Limaiem and ElMaraghy, 1997). They adopted various methods to efficiently create GACs such as a ray casting technique and Boolean operations. By exploiting standard computer graphics hardware, recently, discrete approximation techniques have been used to rapidly create GACs (Spitz *et al.*, 1999; Spitz and Requicha, 2000).

## 2.3 Disassembly Planning

Disassembly planning plays a key role in DM. It aims to generate disassembly sequences with subsequent disassembly actions (Lambert, 2003). The initial attempt of disassembly planning was rooted in assembly planning. Using the various methods developed for assembly planning, a disassembly sequence is determined by the separation of an assembly into two or more subassemblies. It is called *reverse assembly planning*, which is based on the complete reverse assembly (or disassembly) analysis of a product. However, disassembly is usually not performed to its full extent, and incomplete disassembly is often preferred for DM. Although a partial disassembly for DM can be obtained from a complete disassembly analysis, it does not guarantee an optimal disassembly sequence for DM.

SD is one of the efficient approaches for planning partial and incomplete disassembly for DM. SD is defined as the disassembly of selected parts in a product. The SD aims to minimize efforts to separate parts selected for separation or replacement. Unlike an assembly or disassembly, SD allows a partial and non-sequential disassembly procedure regardless of the assembly indenture level of a product, with which a complete procedural assembly or disassembly sequence can be generated.

There are three strategies currently applied for SD, which are destructive SD, non-destructive SD or combinations. In materials recycling, destructive SD is regarded as a commonly used strategy, which includes shredding and other flexible destructive methods (Basdere and Seliger, 2003; Cui and Forssberg, 2003; Lambert, 2003; Pak and Sodhi, 2002). These methods provide an effective and economic solution for materials recycling without considering an assembly structure or indenture level. With these methods, however, only materials are persevered, and geometric details of the product may be lost (Bras, 2004).

Based on component-based sequence planning, non-destructive SD by unscrewing fasteners is often required for such activities as maintenance (repair), product reuse, product recycling or remanufacturing. Without damaging components, non-destructive SD aims to quickly isolate selected parts, not including any part of joining elements-fasteners. This enables resources (*i.e.*, material and energy) used for a product in manufacturing to be preserved. Thus, this method has the highest priority from the environmental point of view (Bras, 2004; Shu and Flowers, 1999).

Component-based sequence planning for SD has been addressed by several researchers. As a dominant approach to SD sequence planning, the wave propagation (WP) method was suggested by Srinivasan and Gadh (1998). This method defines disassembly waves to topologically arrange the parts associated with the disassembly of selected parts. It plans an optimal disassembly sequence based on the waves. The WP method uses two objectives in reducing the SD cost, minimizing the number of component removals and the total weight of components to disassemble (Srinivasan *et al.*, 1999).

Garcia *et al.* (2000) proposed an approach to reduce the computational complexity of the WP method. Instead of constructing waves for selected parts, the approach searches for an optimal disassembly sequence via a predefined precedence graph that describes the shortest path to all exterior nodes. Another effort to reduce the complexity was made by Mascle and Balasoiu (2003). The authors suggested an algorithmic approach to choose a minimal number of parts that form disassembly waves based on the WP method. However, these approaches based on topological disassemblability of parts ignore two important issues: batch disassembly of parts for minimizing efforts in SD, and tool feasibility in partial and non-sequential disassembly.

The batch disassembly of parts is based on modularization to simplify a disassembly problem by defining subassemblies that are relatively tightly connected to other components in an assembly (Lambert, 2003). This batch removal process in SD allows an optimal sequence to minimize efforts to access parts selected for separation or replacement. Moreover, the tool feasibility allows a reliable and practical sequence that ensures the proficiency of removing fasteners by a tool in partial and non-sequential disassembly. Therefore, it is unlikely that WP-based methods can generate an efficient and optimal sequence for SD.

A potential approach to SD integrating batch disassembly of parts and tool feasibility to fasteners can be developed from the fastener-based approaches used in assembly or disassembly sequence planning. Akagi *et al.* (1980) initially emphasized the significance of fasteners in assembly sequence planning. Gui and Mantyla (1994) used fasteners as a description of features to provide constraints for joint components. However, these investigations only focused on assembly modelling based on the nature of fasteners. In the research (Kuo, 2000), a method was proposed to decompose an assembly into subassemblies connected by fasteners. It starts with identifying cut-vertices, bi-connected graph and pendent vertices in a component-fastener graph via the depth first search (DFS) algorithm. Disassembly sequence is planned using an objective function and the disassembly precedence matrix based on the identified subassemblies. Tseng and Li

(1999) decomposed an assembly into a set of fastener-based subassemblies, and then generated assembly sequences by an interference-checking method based on CAD boundary representation (B-rep). Furthermore, Yin *et al.* (2003) decomposed an assembly into a set of fastener-based subassemblies with hierarchical structure. They used a relational model graph (Homem De Mello and Sanderson, 1991) instead of the CAD B-rep interference checking (Tseng and Li, 1999). However, the above approaches fail to notice the importance of tool feasibility because they mainly focus on the generation of a complete procedural assembly or disassembly sequence for a product with fastener-based structure. Although an SD sequence for selected parts may be obtained from a complete disassembly analysis by the above approaches, there is no guarantee that this will generate an efficient and optimal sequence for SD.

## 2.4 Technologies Related to Web-Based PD&M

Tremendous attempts have been made to implement systems for Web-based PD&M. A Web CAD, a part of an agent-based application-CyberCut, was introduced by Smith and Wright (1996). Built on the Java two-dimensional (2D) platform, modelling a part was performed through the destructive solid geometry the authors proposed in their research.

A more advanced system was suggested by Xu *et al.* (2003). In their research, the Web-based product modelling tool (WPMT) was implemented, which aims to enable a CPD through the Internet using Java 3D with geographically dispersed team members. As a feature-based 3D modeller, the WPMT provides designers with built-in-features and manipulation functions such as Boolean operations, scaling, rotations and translations. In particular, the authors addressed the importance of a multi-tier Internet environment for a

CPD in a geographically dispersed design environment. The multi-tier Internet environment is suitable for combining Web-based CAD systems and a Web-based product data management (PDM) system. The Web-based PDM system serves as the data sharing management module, provides search functions, and manages the product data information (Huang *et al.*, 2004; Xu and Liu, 2003). However, this notion was not accomplished in their research.

Lau *et al.* (2003) proposed an interactive design visualization system (IDVS) to facilitate an interactive product design at the conceptual design stage. It was implemented via Java, Visual Basic and virtual reality modelling language (VRML). As a Web-based virtual prototyping (VP) tool, the IDVS mainly focuses on concurrent visualization of a new product in order to communicate the design concept and to share the production data. In particular, the authors emphasized the notion of a global design environment where designers, customers and suppliers are concurrently involved.

A modelling and simulation system was proposed by Qin *et al.* (2004). Based on Java 2D and 3D, and VRML, this system aims to support a distributed machine design and control paradigm proposed in their research. Java 2D and 3D are used to implement a function to model 3D components, and to define their motions and connections. VRML is used for their assembly simulation via the easy assembly concept of *drag-and-drop* assembly on the Internet. Although the authors discussed the potentials of a distributed design and simulation in their research, no real test was executed on the Internet.

As decision supporting tools to CPD, various DFX systems including DFA and DFM systems have been adopted in Web-based PD&M environments. The potentials of various

Web-based DFX systems were examined by several researchers (Huang and Mak, 1999; Huang *et al.*, 1999; Huang and Mak, 2001). In particular, the researchers attempted to implement Web-based DFX systems using various Web technologies including active server page (ASP) and Active-X components to improve intractability for users. Because of the platform-dependant issue, however, hypertext mark-up language (HTML) and Java applets are widely used today for developing the client side of user interfaces (Xu *et al.*, 2003; Zhang *et al.*, 2004; Zeng *et al.*, 2003).

Web-based VM based on high fidelity simulation and validation has been extensively adopted in Web-based PD&M environments. It is capable of generating information about the structure, status, and behaviour of a manufacturing process and system, all of which can be observed in a manufacturing environment (Seo *et al.*, 2005). As an example, Qiu *et al.* (2001) proposed a Web-based virtual machining simulation. In their research, the Web-based system for the machining simulation was developed via Java and VRML-EAI. This system imports a predefined numerical control (NC) code, interprets it into tool motion data, and simulates a material removal process in real time. In particular, they used a scene description node of VRML, *ElevationGrid* node, to prevent performance deficiency in a client computer. Although this system provides users with more enhanced interactivity via a cost effective virtual-reality (VR) technology through the Internet, real manufacturing processes are not modelled into the developed system, which are essentially required for a VM system.

Extended research on the Web-based VM was carried out by Ong *et al.* (2002). In their research, a Web-based virtual computerized numerical control (CNC) milling system was proposed. This system was developed based on Java and VRML-EAI, which are the same technologies used in the research by Qiu *et al.* (2001). DBs including a machine DB, a tool DB and a work-piece material DB were also incorporated into the system so that a more practical solution was achieved in a networked environment. In particular, this system includes several manufacturing process models to achieve the virtual realization of real machining processes. The process models are based on collision detections and machining knowledge stored.

More advanced experiments for Web-based VM were investigated by several researchers. Kong *et al.* (2002) developed a common-object-request-broker-architecture (CORBA)-based virtual machining system that consists of a NC file analysis module, a simulation module, and a real machine-control module. Through the Internet, the simulated NC code via the developed system was downloaded to a real machining centre, the Benchman 4000. This is a promising attempt to link a VM system with a real manufacturing system over the Internet.

Interaction can be achieved by adding monitoring through the Internet that links a Web-based VM system and a real manufacturing system. For instance, Wang *et al.* (2004) proposed Web-based real-time monitoring. In their research, the monitoring function combines with remote machining that is based on virtual simulation via a developed Web-based VM system.

Some research addressed the extended role of Web-based PD&M within an enterprise. Lee (2003) discussed the need to combine Web-based PD&M systems and E-business systems including enterprise resource planning (ERP) systems. The ERP systems are the financial backbone of most enterprises. To achieve this notion, the author suggested several future directions: Internet-based monitoring and analyzing systems of machinery, information pipeline or platform for synchronization with manufacturing, scheduling systems, inventory systems, supply chain systems and ERP, and virtual design platforms for collaborative part, process, tooling design among suppliers, design and process engineers and customers for fast validation and decision-making.

In the research performed by Koc *et al.* (2003), the authors emphasized the PD&M operations in successfully integrating with the functional objectives of an enterprise using Internet, tether-free (*i.e.* wireless, Web, *etc.*) and predictive technologies. They believed that this new paradigm to integrate Internet-enabled and predictive intelligence for PD&M systems is becoming a new strategy for enterprises to achieve success in the currently competitive market.

Inter-enterprise resource sharing is recently identified as a crucial feature in Web-based PD&M environments (Cao and Dowlatshahi, 2005; Hao *et al.*, 2005; Webster and Sugden, 2003; Yang and Xue, 2003). It works among globally networked virtual enterprises (VEs) to explore business opportunities for a specific product. The resource sharing is effectively achieved by the use of information technology (IT), including Web and DB technologies (Cao and Dowlatshahi, 2005).

## 2.5 Problem Statement

The major problems found from the literature review are summarized as follows that define the research focus in the dissertation.

### 2.5.1 Islands of automation or expertise syndrome of current task-planning systems

The currently developed task-planning systems lag behind when it comes to Internet usage. They still suffer from disintegration, resulting in so-called *islands of automation* or *islands of expertise* syndrome (Huang and Mak, 1999), while other computerized PD&M systems have been successfully adapted to Web-based PD&M environments. This disintegration leads to insufficiency in supporting Web-based PD&M that ensures agility by collaboration, sharing of product data, process data, information and knowledge, and integration with other Internet-based business and engineering systems (Noh *et al.*, 2004).

Moreover, the failure to attach to the Internet may prevent SMEs from using high efficiency task-planning systems at low cost. Currently available task-planning systems require a huge amount of cost in application development, administration and maintenance. They are financially beyond the reach of most SMEs. The SMEs are currently facing shrinking budgets, limited computational capacity and diminishing manufacturing resources (Peng, 2004), and looking for cost-effective cutting-edge technology (*i.e.*, Web-based engineering tools).

### 2.5.2 Disregard for resource involvement in task planning

Although the current stand-alone task-planning systems help to generate consistent and reasonable task plans via incorporated knowledge, their effectiveness is still not fully satisfactory for the demands of Web-based PD&M. One of the issues is a poor response to the dynamic nature of product life-cycle environments including a manufacturing environment. For instance, a workshop in the environment often suffers from unexpected disruptions caused by bottleneck machines, non-availability of tools and personnel, or breakdown of machines and equipment. A readily generated task plan for manufacturing becomes invalid and has to be regenerated when production starts (Ahmad *et al.*, 2001; Fernandes and Raja, 2000; Kumara and Rajotia, 2003; Usher and Fernandes, 1999). In task planning, it is strongly required to consider timely available resources including cutting tools and machines.

Furthermore, the incorporated knowledge or methodology for the current task-planning systems is designed to hard support resource-involved task planning that is proposed in this research. For instance, tool feasibility and topological part accessibility are critical considerations in generating a complete assembly or disassembly sequence plan. However, they are still under-explored research areas. Although some methods are proposed to consider these aspects in automatic assembly or disassembly planning, they are so simplified, or complicated to carry out computationally intensive processes. This leads to inefficiency in achieving *rapid* and *precise* task planning with a number of available resource alternatives. New approaches should be developed to aid the resource-involved task planning efficiently.

### 2.5.3 Disregard for resource sharing in task planning

In particular, resource sharing within business partners plays an important role in Web-based PD&M environments. It results from the current marketplace that forces enterprises to form a temporary virtual alignment to respond to changing business environments. The alignment combines the specific core capabilities of its members in order to rapidly exploit manufacturing opportunities associated with a specific product or service (Dowlatshahi and Cao, 2005). Therefore, resource sharing in task planning are highly recommended not only for generating feasible and economical task plans, but also for quickly exploiting a manufacturing opportunity via an enterprise alignment virtually formed in a Web-based PD&M environment.

### 2.5.4 Technical direction for implementing Web-based task planning systems

With the expansion of the Internet and Web-based technologies in the past decade, many new concepts to achieve Web-based PD&M have been proposed, such as CPD and distributed VM. Visualization and product information sharing are the foundation for these concepts, which are accomplished by Web and DB technologies (Zhang *et al.*, 2004). For distributed processing in the client side, several attempts have been made, including Active-X and common object model (COM) without communication overheads with the server. Because of the platform dependant issue, however, HTML and Java applets are widely used for developing user interfaces of the client side.

For Web-based visualization, VRML is widely used as a neutral representation in various geometric models. The size of VRML files is smaller than most other 3D file formats, and they can be transferred easily through the Internet (Xu *et al.*, 2003). VRML is advantageous to be easily viewed via the standard Web browsers and to be converted from heterogeneous CAD tools, requiring no further developmental efforts. Moreover, the Java applet-VRML combination via external authoring interface (EAI) enables for implementing both a complicated processing and its visualization in the client side.

Web-based PD&M systems usually run on the three-tier architecture with JDBC techniques used for product information sharing over the Internet. This architecture avoids keeping a huge amount of product data in one site, which are used for CPD and distributed VM over the Internet. Therefore, it is feasible to implement applications for dynamic task planning to be developed, and to integrate them into a Web-based PD&M environment with geographically dispersed partners.

## 2.6 Research Strategy

A strategy is applied to achieve the objective of this research, which is to develop efficient approaches to dynamic task planning, and to integrate them into a Web-based PD&M environment. In particular, this dissertation focuses on the three dynamic task-planning activities closely reviewed, which are: (1) process (fabrication) planning at the shop floor level; (2) assembly planning at the production level; and (3) disassembly planning at the product end-life level. As for pursuing the notion of dynamic task planning, resource sharing and involvement on the Internet, the research strategy is to give attention to the following details: (1) modelling knowledge and resource DBs; (2) developing DB search algorithms; (3) developing operation methods; (4) defining OODBs; (5) developing task optimization methods; (6) developing 3D visualization methods; and (7) implementing Web-based integration. These issues work together to efficiently achieve the dynamic task planning of an intended task in a Web-based PD&M environment.

### 2.6.1 DB modelling

Various DBs including a cutting-tool DB, a machine DB, a material DB, an assembly tool DB, and a knowledge DB are modelled into a RDBMS. The DBs are designed to structure and store resources and knowledge in manufacturing and DM environments. Each DB in RDB is defined with tables and their relationships such as one-to-one, one-to-many, many-to-many, one-to-one recursive, and one-to-many recursive relationships. It is recommended that the DB modelling should be executed based on standardized specifications, for instance, international standard organization (ISO) standards. This is a generic approach so that the modelled DB can be used in industry without the need to modify it significantly.

### 2.6.2 DB search algorithms

DB search algorithms are developed to efficiently retrieve tool and machine alternatives available for the completion of an intended task. Two issues are considered in this work. The first consideration is to reduce the number of transactions with a remote DB server through the Internet, and the second one is to reduce the amount of data to be retrieved, which responds to user request. The reduced amount of data helps to achieve resource-involved task planning efficiently. The DB search algorithms are based on dynamic structured query language (SQL) queries and search criteria defined in this research. The initially retrieved tools and machines are embedded in the task-planning process. This results in optimized process details, including the best set of tools and machines to be used.

### 2.6.3 Operation methods

Operation methods are developed to provide various computations demanded during dynamic task planning. These computations can include: (1) manufacturing feature recognition from a standard CAD format; (2) set-up planning and sequencing; (3) machining process determination; (4) tools and machines selection; (5) machining time and cost calculation; (6) assembly (or disassembly) time and cost calculation; (7) assembly-tool feasibility analysis; (8) topological part disassemblability analysis; (9) maintainability calculation; (10) vector and matrix calculation; and (11) 3D geometry transformations for scaling, translating and rotating.

### 2.6.4 OODBs

Various OODBs are utilized to efficiently manipulate information that is retrieved from DBs in a remote DB server, or newly generated during dynamic task planning. The OODBs in this research are defined as a set of instances. The instances are based on the Java classes developed. The Java classes include *AssyToolOODB*, *MachineOODB*, *VRMLGeometryData*, and *TurningToolOODB*. Based on the OOP paradigm, the OODBs consist of encapsulated attributes and methods. A set of data retrieved from DBs is set to the attributes in each OODB. In addition, the developed operation methods are mainly interfaced to the methods of the OODB. Therefore, the use of OODBs provides an easy and effective manipulation of various data used for dynamic task planning.

### 2.6.5 Task optimization

Several optimization methods are developed to generate optimal process details for an intended task, including the best set of tools and machines. These methods are based on the GA and several heuristics developed in this research. For a given task, in particular, the developed OODBs are incorporated into the optimization process for generating feasible and practical process details.

### 2.6.6 3D Visualization

3D visualization is executed via static and dynamic VRML-based interactions developed. The two interactions are implemented by the standardized programming languages, Java and VRML-EAI. They provide the interaction of human and virtual objects, the precise time-dependant 3D simulation of a generated task plan, and the real time acquisition of needed data during the simulation process. In particular, the acquired data from computerized objects is used to calculate other useful information including machining time, and elapse time for assembly or disassembly.

### 2.6.7 Web-based integration

Web-based systems are utilized for the integration of dynamic task planning developed. They are built on neutral and standardised technologies. The technologies include the Web technologies of Java, HTML and VRML-EAI, and the DB technologies of RDB and JDBC. Thus, the implemented Web-based systems work on a three-tier Internet-based architecture. This architecture consists of a Web server, a DB server and clients. It is a basic model to form a multi-tier Internet-based environment that is widely adopted in establishing a Web-based PD&M environment. In the Web-based environment, the dynamic task planning systems subsequently achieve collaborative, cost-effective, platform independent, and visualized information sharing over the Internet.

## 2.7 Research Hypothesis

In this research, a Web-based PD&M environment employs the dynamic task-planning systems developed. It is assumed that manufacturing and DM business partners are networked with this Web-based environment. The business partners can be geographically dispersed, and participate in a temporary alignment for a specific product being designed during its life cycle.

## 2.8 Chapter Summary

This chapter reviewed literature in process planning, assembly planning, and disassembly planning. Based on this review, inefficiencies of current task planning were described in achieving dynamic task planning based on resource involvement and sharing. Technologies related to current Web-based PD&M systems were investigated to indicate a technical direction by which dynamic task planning approaches to be developed can be efficiently implemented in a Web-based environment. This review subsequently resulted in the problem statement, and research strategy and hypothesis. In the next chapter, an approach to dynamic process planning is presented with sample applications.

# Chapter 3

# Dynamic Process Planning

This chapter presents an approach to dynamic process (fabrication) planning at the shop floor level. In addition to current research areas such as manufacturing feature recognition, set-up planning and sequencing, and machining process determination, the selection of tools and machines at a dynamic shop floor is embedded into process planning. DB search algorithms and selection knowledge are developed to efficiently determine tool and machine alternatives from DBs.

## 3.1 Proposed Approach

The proposed approach for dynamic process planning aims at generating a high-fidelity process plan to manufacture a part, and to provide tool and machine alternatives based on dynamic manufacturing environments. The main idea of this approach is to embed available manufacturing resources at the shop floor into the process planning of a part. Once specific resources including tools and machines are chosen for the process plan of a part, they are not available until the part is completely manufactured at the shop floor. In particular, a dynamic process planning for rotational parts is discussed in this chapter. A rotational part is represented by the symmetric geometry about its centre line. It is a widely used mechanical part in a product. The following are major functions included in this proposed approach.

***Manufacturing feature recognition***. This process converts design information into manufacturing features for process planning. Rule-based algorithms are proposed to convert design information into object-oriented manufacturing features (OOMFs). The OOMFs for rotational parts are represented by twenty hierarchical Java classes developed in this research.

***Set-up planning and sequencing***. A set-up planning and sequencing algorithm considering tolerance factors is applied to minimize the number of set-up sequences. As a result of this process, two or more set-up sequences are generated with datum information including a datum plane and a clamping position.

***Machining process determination***. A machining process is determined based on the limitation of surface roughness generally obtained from different machining processes. For a given dimensional or geometric tolerance instead of surface roughness, a machining process is determined by using typical ISO tolerance grades assigned for different machining processes. Based on users' interaction to input required surface roughness or tolerance limits for a part, the machining processes are automatically determined by the system developed in this research.

***Tools and machines selection***. DB search algorithms and knowledge are developed to retrieve available tools and machines at the shop floor. Based on weighing factors related to production cost and time, possible tool and machine alternatives are ranked to select the best set of tools and machines to manufacture a part.

In particular, generic DB technologies are used to efficiently support the above processes. The DB technologies including RDB and DB connectivity like JDBC help to efficiently define and transact a huge amount of tool- and machine-related information in a DB server. Moreover, OODBs based on the OOP technique are used to efficiently manipulate the retrieved tool- and machine-related information within the developed system for dynamic process planning. The OODBs are defined as a set of instances that are instantiated from the developed Java classes in this research. The OODBs include encapsulated attributes and methods to provide an efficient manipulation of the data for other modules developed for process planning.

## 3.2 Data Structure and Database

A well-organized DB structure is introduced to accomplish dynamic process planning in this research. Figure 3.1 shows an entity-relationship diagram (ERD) of the overall DB structure. As shown in Figure 3.1, each entity that is converted to a DB represents the fact in manufacturing environments. The entity is complicatedly associated with others in the environments. For instance, product data are organized based on the hierarchical structure, such as the bill of materials (BOM) used. As shown in Figure 3.1, the product data is associated with a set of part data. A part consists of a number of manufacturing features that may require different machining processes. A manufacturing feature is divided into a set of manufacturing faces used for tolerance assignment. The tolerance information and a chosen material type help the manufacturing feature determine a more suitable machining process with available tools and machines to be used. Subsequently, a process plan can be then assigned to a part, collecting all the process or fabrication information generated. Proper knowledge interacts with the entities in order to efficiently accomplish the process planning of a part.

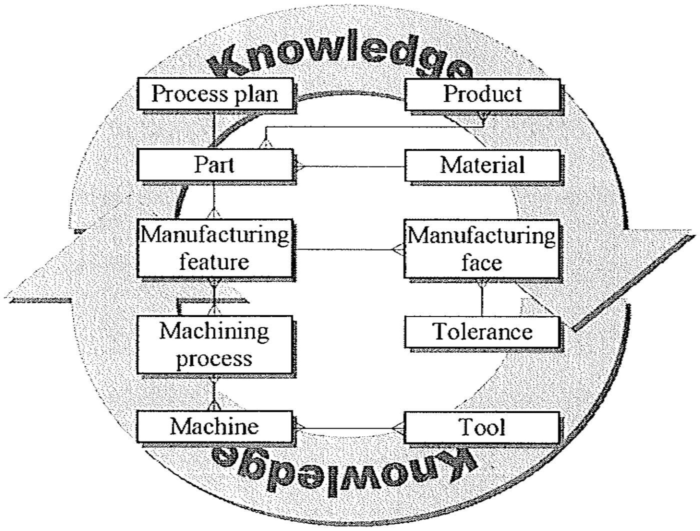
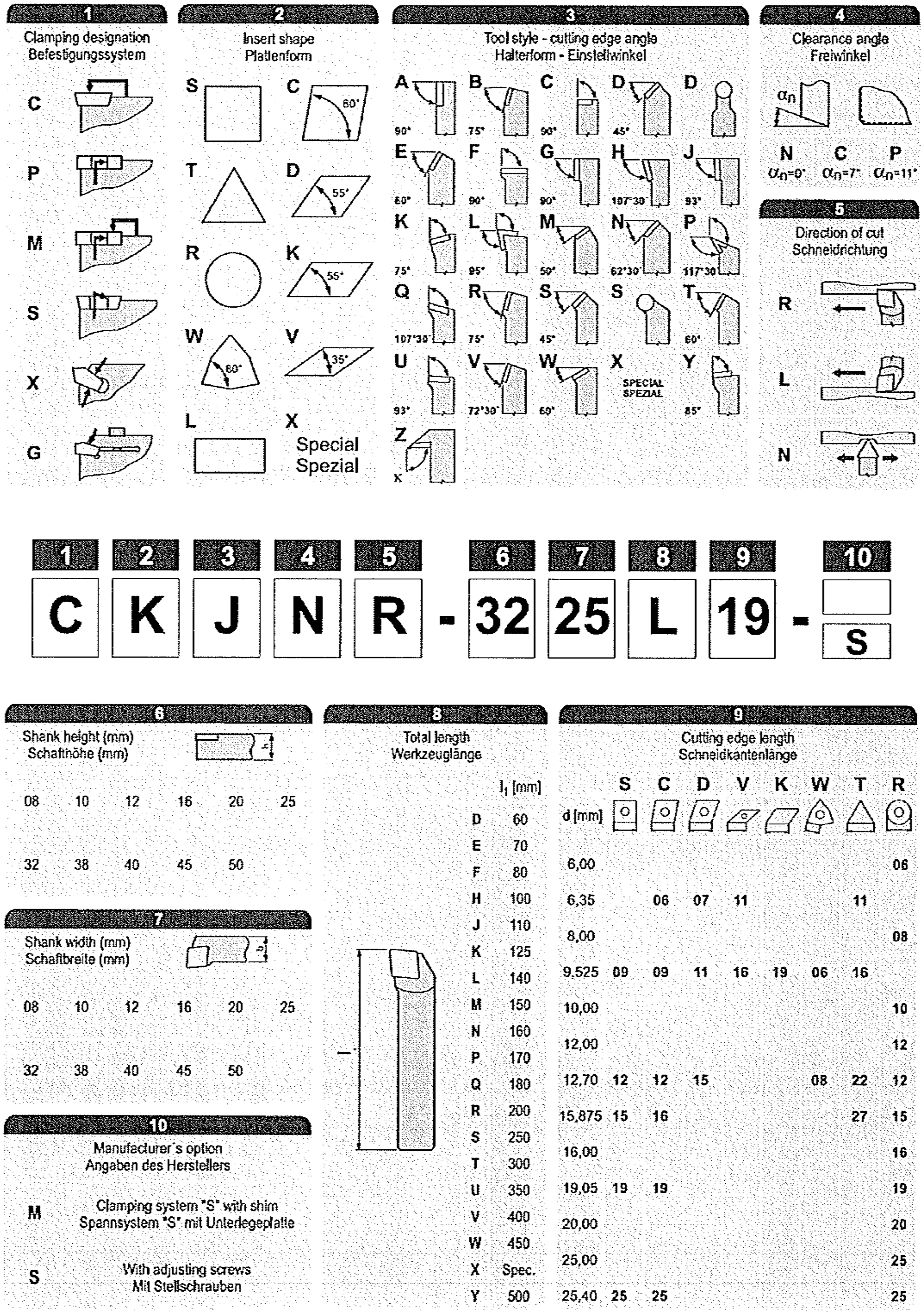


**Figure 3.1**: An ERD of the overall DB structure developed in this research.

MySQL server version 3.23, an RDBMS, is used to define and manage the DBs shown in Figure 3.1, which include a machine DB, a tool DB and a material DB. A DB in RDB is defined with tables and their relationships such as one-to-one, one-to-many, many-to-many, one-to-one recursive, and one-to-many recursive relationships (Watson, 1999). For instance, 36 tables with several types of relationships (Refer to Appendix A.1) are used to define the tool and machine DBs in this research. In particular, the tool DB is modelled via ISO standards. This DB is generic so that it can be commonly used in LMEs and SMEs using the tools designated by ISO. Figure 3.2 shows an example of the ISO code designation of external insert holders.



| | | | | | |
|---|---|---|---|---|---|
| 08 | 10 | 12 | 16 | 20 | 25 |
| 32 | 38 | 40 | 45 | 50 | |

| | | | | | |
|---|---|---|---|---|---|
| 08 | 10 | 12 | 16 | 20 | 25 |
| 32 | 38 | 40 | 45 | 50 | |

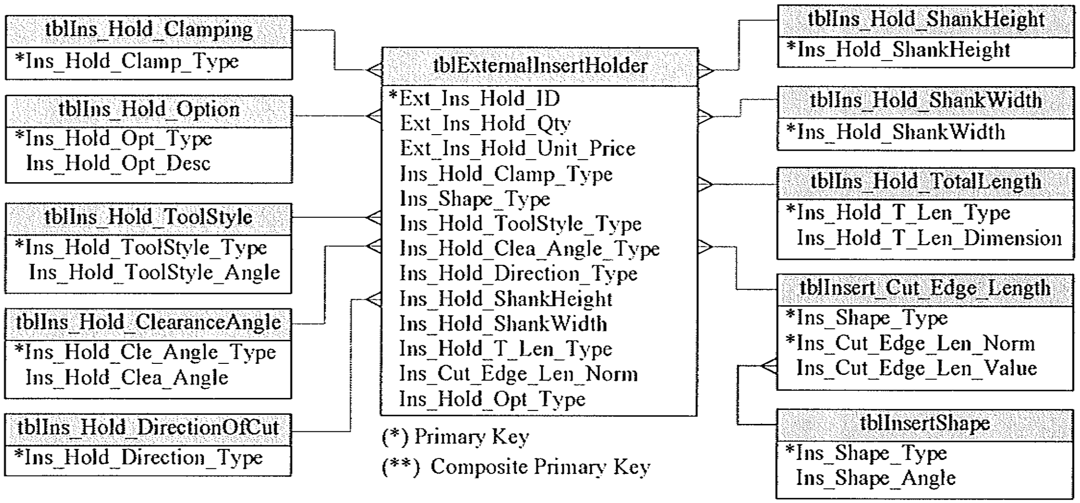
| | $l_1$ [mm] |
|---|---|
| D | 60 |
| E | 70 |
| F | 80 |
| H | 100 |
| J | 110 |
| K | 125 |
| L | 140 |
| M | 150 |
| N | 160 |
| P | 170 |
| Q | 180 |
| R | 200 |
| S | 250 |
| T | 300 |
| U | 350 |
| V | 400 |
| W | 450 |
| X | Spec. |
| Y | 500 |

| d [mm] | S | C | D | V | K | W | T | R |
|---|---|---|---|---|---|---|---|---|
| 6,00 | | | | | | | | 06 |
| 6,35 | | 06 | 07 | 11 | | | 11 | |
| 8,00 | | | | | | | | 08 |
| 9,525 | 09 | 09 | 11 | 16 | 19 | 06 | 16 | |
| 10,00 | | | | | | | | 10 |
| 12,00 | | | | | | | | 12 |
| 12,70 | 12 | 12 | 15 | | | 08 | 22 | 12 |
| 15,875 | 15 | 16 | | | | | 27 | 15 |
| 16,00 | | | | | | | | 16 |
| 19,05 | 19 | 19 | | | | | | 19 |
| 20,00 | | | | | | | | 20 |
| 25,00 | | | | | | | | 25 |
| 25,40 | 25 | 25 | | | | | | 25 |

| | |
|---|---|
| M | Clamping system "S" with shim<br>Spannsystem "S" mit Unterlegeplatte |
| S | With adjusting screws<br>Mit Stellschrauben |

**Figure 3.2**: An example of the ISO code designation-external insert holders (Pramet Co., 2001).

As shown in Figure 3.2, an external insert holder is represented by 10 pre-designated codes, and the description of each code is described in detail. Figure 3.3 shows an example of data modelling for the external insert holders shown in Figure 3.2. A table is defined via the data definition language (DDL) syntax of SQL, which is presented in Figure 3.3. Based on this syntax, a table named *tblExternalInsertHolder* is defined, consisting of appropriate fields and data types. Every external insert holder data including the unit price and number of quantities at the shop floor is additionally stored in this table.



DDL syntax of SQL for creating a table

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
tbl_name [(create_definition, ...)] [table_options]
[select_statement]
create_definition:
col_name type [NOT NULL | NULL] [DEFAULT
default_value] [AUTO_INCREMENT]
      [PRIMARY KEY] [reference_definition]
  or  PRIMARY KEY (index_col_name,...)
  or  KEY [index_name] (index_col_name,...)
  or  INDEX [index_name] (index_col_name,...)
  or  UNIQUE [INDEX] [index_name]
(index_col_name,...)
                    ⋮
```

The SQL statement to create the table named *tblExternalInsertHolder*

```
CREATE TABLE tblExternalInsertHolder (
    ext_ins_hold_ID          int(11)     auto_increment,
    ext_ins_hold_qty         int(11),
    ext_ins_hold_unit_price  float,
    ins_hold_clamp_Type      char(1)     NOT NULL,
                             ⋮
    ins_hold_opt_type        char(1)     NOT NULL,
    PRIMARY KEY (ext_ins_hold_ID),
    FOREIGN KEY(ins_hold_clamp_Type)
    REFERENCES
    tblins_hold_clamping (ins_hold_clamp_Type) ON
    DELETE CASCADE,
                             ⋮
```

**Figure 3.3**: An example of data modelling and table definition for external insert holders shown in Figure 3.2.

As shown in Figure 3.3, the *tblExternalInsertHolder* table is modelled in the one-to-many relationships with the tables for 10 pre-designated codes. This structure is advantageous to maintain data integrity and to efficiently manage data because each table stores its own information that is neither kept in other tables nor managed by them. Instead, relationships based on primary and foreign keys efficiently make connections among the tables. Moreover, primary and foreign key interactions enable to reduce the searching speed to find or form specified information, allowing an index-based searching scheme. The developed system for dynamic process planning can interact with DBs via data manipulation language (DML) of SQL frequently and quickly.

## 3.3 Manufacturing Feature Recognition

In this research, a manufacturing feature is defined as the element that can be formed by one manufacturing process. The feature parameters consist of design geometric data, surface roughness, dimensional and geometric tolerances, and methods to form the manufacturing feature. A feature is not directly available from a product design, such as a CAD file. Therefore, feature recognition is required to convert the design data into manufacturing features that are used to generate a process plan.

Data exchange standards have been established between heterogeneous CAD systems. Among standard data-exchange formats developed, data exchange file (DXF), initial graphics exchange specification (IGES) and standard for the exchange of product data (STEP) are the most widely accepted. The DXF format is selected as the input of design data for manufacturing feature recognition. This selection is based on the current manufacturing practice where a DXF formatted part drawing is usually given to a planner to perform process planning. A feature management module is developed here for this purpose, which also supports the feature-based modelling of a part for process planning. A part design is represented by geometric entities such as lines, curves, and surfaces. From a DXF format file, the feature recognition starts with obtaining geometric entities that will be used to form manufacturing features. Figure 3.4 shows brief steps to convert the DXF representations into the geometric entities defined in this research. The detailed flow chart of these steps is also illustrated In Appendix A.2.

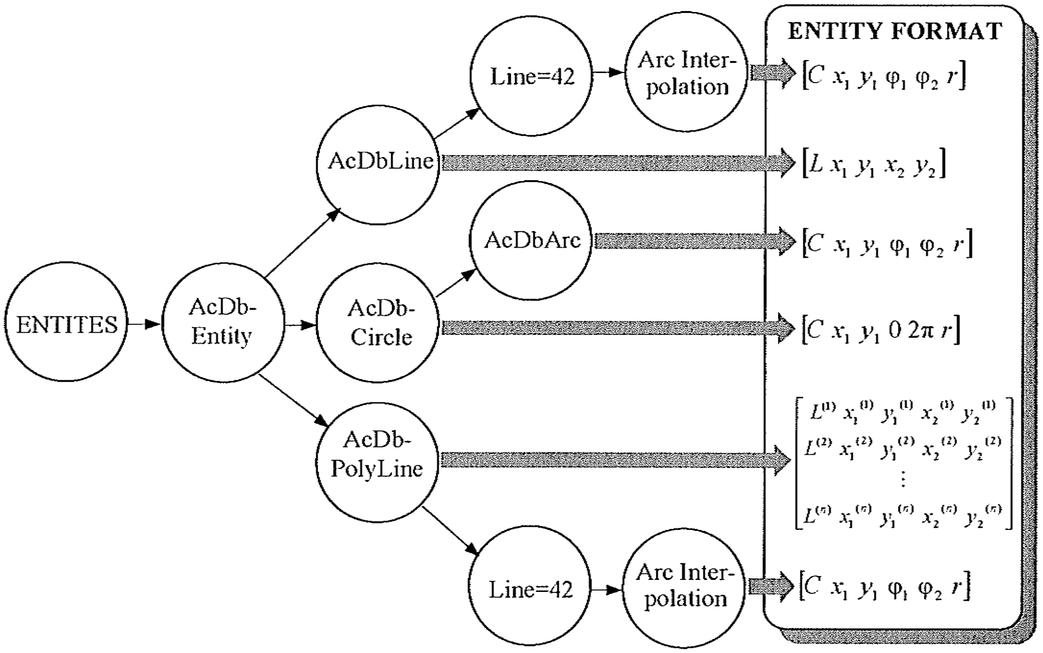

ENTITY FORMAT
Line=42
Arc Inter-polation
$[C\ x_1\ y_1\ \varphi_1\ \varphi_2\ r]$
AcDbLine
$[L\ x_1\ y_1\ x_2\ y_2]$
AcDbArc
$[C\ x_1\ y_1\ \varphi_1\ \varphi_2\ r]$
ENTITES
AcDb-Entity
AcDb-Circle
$[C\ x_1\ y_1\ 0\ 2\pi\ r]$
AcDb-PolyLine
$\begin{bmatrix} L^{(1)}\ x_1^{(1)}\ y_1^{(1)}\ x_2^{(1)}\ y_2^{(1)} \\ L^{(2)}\ x_1^{(2)}\ y_1^{(2)}\ x_2^{(2)}\ y_2^{(2)} \\ \vdots \\ L^{(n)}\ x_1^{(n)}\ y_1^{(n)}\ x_2^{(n)}\ y_2^{(n)} \end{bmatrix}$
Line=42
Arc Inter-polation
$[C\ x_1\ y_1\ \varphi_1\ \varphi_2\ r]$

**Figure 3.4**: Steps of converting geometric entities from DXF representations.

In Figure 3.4, for instance, a line entity is converted from the DXF representations such as *AcDbLine* or *AcDbPolyline*, and stored in a defined entity format, [*Line_type*, *Point1*, *Point2*]. *Line_type* defines either a straight line or a tangential line. In particular, if a line entity from a DXF file includes the number *42*, it will need a circular interpolation in the following line because the DXF format does not provide information with respect to the circle between two lines. The circular entity interpolated here is also stored in a defined circular entity format, [*Circular_direction, Centre_point, Start_angle, End_angle, Radius*]. In the above entity format, *Circular_direction* is defined as one of the two directions, counter clockwise (CCW) and clockwise (CW) directions. Figure 3.5 and the following Equations (3.1), (3.2) and (3.3) are used to interpolate a circle between two lines from a DXF file.

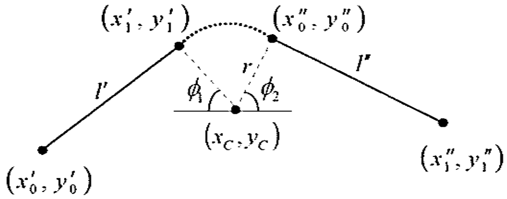


**Figure 3.5**: Circle interpolation between two arbitrary lines.

Determination of the centre point $x_c$ and $y_c$;

$$x_c = \left(\frac{b_2 - b_1}{a_1 - a_2}\right), \quad y_c = a_1 x_c + b_1 \qquad (3.1)$$

Where, $a_1 = -\left(\dfrac{x_1' - x_0'}{y_1' - y_0'}\right)$, $b_1 = y_1' - a_1 x_1'$ and $a_2 = -\left(\dfrac{x_1'' - x_0''}{y_1'' - y_0''}\right)$, $b_2 = y_0'' - a_2 x_0''$.

Determination of the radius $r$;

$$r = \sqrt{(x_c - x_1')^2 + (y_c - y_1')^2} \qquad (3.2)$$

Determination of two angles $\phi_1$ and $\phi_2$;

$$\phi_1 = \left|\tan^{-1}\left(\frac{y_c - y_1'}{x_c - x_1'}\right)\cdot\frac{180}{\pi}\right| \text{ and } \phi_2 = \left|\tan^{-1}\left(\frac{y_c - y_0''}{x_c - x_0''}\right)\cdot\frac{180}{\pi}\right| \qquad (3.3)$$

Once geometric entities are obtained and stored in the developed system, the next procedure converts these entities into manufacturing features. This process uses decision-making diagrams to determine the type of a feature. There are three types of decision-making diagrams (Refer to Appendix A.3) developed in this research; one of them is illustrated in Figure 3.6.

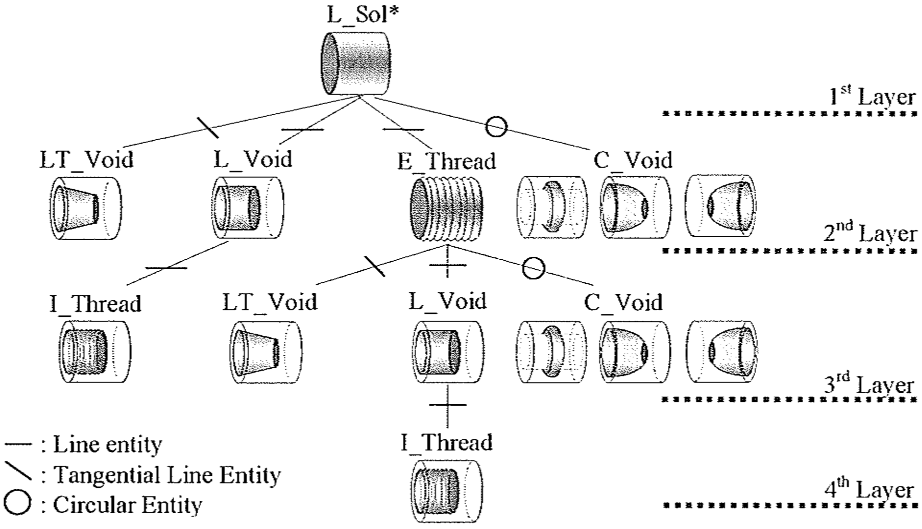


**Figure 3.6**: A decision making diagram for the feature definition based on obtained geometric entities.

In Figure 3.6, if the second layer entity is the same line entity with the first one, it should be an *L_Void* or *E_Thread* feature depending on the type of the next layer entity. In this case, the type of the next layer entity (*i.e.*, the third layer entity) is examined to determine the filling type of the feature for the second layer entity. Based on these rules, a feature can be formed from obtained geometric entities. In particular, an OOMF based on OOP is introduced here to efficiently store and manipulate the defined feature. Figure 3.7 shows a hierarchical class diagram of the OOMFs representing a groove, a tap, an internal and an external cylinder, and a thread, which are inherited from the super class named *ManufFeature* class.

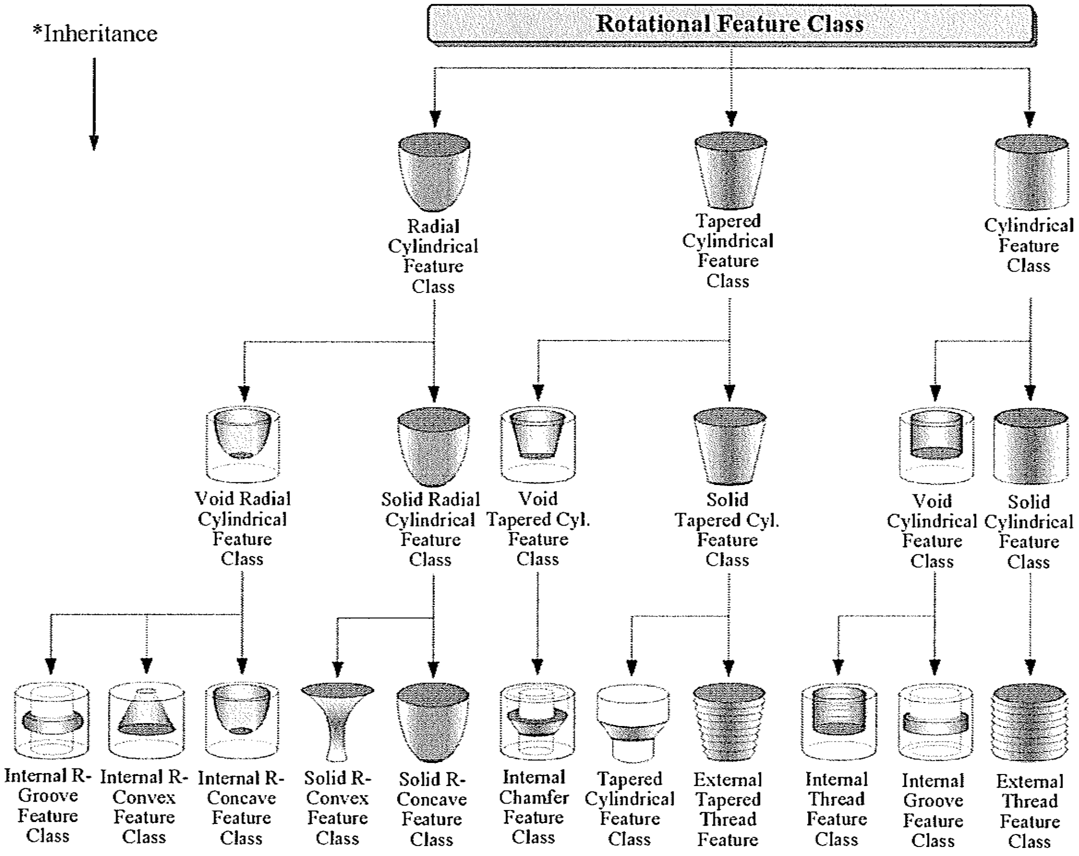

*Inheritance
Rotational Feature Class
Radial Cylindrical Feature Class
Tapered Cylindrical Feature Class
Cylindrical Feature Class
Void Radial Cylindrical Feature Class
Solid Radial Cylindrical Feature Class
Void Tapered Cyl. Feature Class
Solid Tapered Cyl. Feature Class
Void Cylindrical Feature Class
Solid Cylindrical Feature Class
Internal R-Groove Feature Class
Internal R-Convex Feature Class
Internal R-Concave Feature Class
Solid R-Convex Feature Class
Solid R-Concave Feature Class
Internal Chamfer Feature Class
Tapered Cylindrical Feature Class
External Tapered Thread Feature
Internal Thread Feature Class
Internal Groove Feature Class
External Thread Feature Class

**Figure 3.7**: Hierarchical class diagram of the OOMFs defined in this research.

The *ManufFeature* Java class is also described in Appendix A.4. This class in Appendix A.4 is interfaced with the *FeatureManagement* class, which imports the DXF file of a part, and converts the obtained geometric entities to OOMFs via methods previously presented in this section. As shown in Appendix A.4, in particular, a defined feature has the information about its manufacturing faces that are used for tolerancing, or set-up planning and sequencing. The manufacturing faces here are simply calculated from defined features. For instance, each feature initially generates four manufacturing faces: right, left, upper and lower faces. Then, the related manufacturing faces in adjacent features are merged or redefined via simple rules developed. These rules are formulated by using the normal vectors (±*X*, ±*Y*) of adjacent manufacturing faces that are examined by such properties as *prevFeatureID* and *ingredientList* in Appendix A.4.

## 3.4 Set-up Planning and Sequencing

The set-up planning and sequencing defines a set of manufacturing faces that can be manufactured at one set-up on a machine tool. The features defined in Figure 3.7 are divided into a set of manufacturing faces to efficiently define tolerances including dimensional and geometric tolerances. These tolerances can be here categorized into local tolerances (*i.e.*, tolerance of a diameter of a hole) and relative tolerances (*i.e.*, geometric tolerance like concentricity, or tolerance of a dimension between two faces). Depending on the number of faces used to define a tolerance, the notion of this categorization is applied for calculating tolerance factors proposed by Huang (1998).

In this research, the set-up planning and sequencing algorithm via the tolerance factors is adopted to minimize the number of set-up sequences. This algorithm was suggested by Huang and Zhang (Huang, 1998; Huang and Zhang, 1996). A tolerance defined in a single manufacturing face, or more than one manufacturing faces can be converted into a tolerance factor by dividing the tolerance value by the representative length. This is relative tolerance information representing which manufacturing faces combination has a tighter tolerance relationship in a part to manufacture. Based on the definition by Huang (1998), several examples (Refer to Appendix A.5) of calculating tolerance factors are introduced in this dissertation.

The set-up planning and sequencing algorithm suggested by Huang and Zhang (Huang, 1998; Huang and Zhang, 1996) is also summarised in Appendix A.6. This algorithm starts with grouping manufacturing faces with tight tolerance relationships, which should be machined at the same set-up based on tool approach directions. Then, the selection of set-up datums is executed to facilitate tolerance control, namely, manufacturing faces that have tight tolerance relationship needed to be mutually clamped and referenced. These set-up datums consist of a datum plane and a clamping position for a lathe machine. Subsequently, the machining priority that determines a set-up sequence is checked with the availability of set-up datums and a set of manufacturing faces with tight tolerance relationships.

## 3.5 Machining Process Determination

In this research, the machining process of a manufacturing face is determined by: (1) the limitation of the surface roughness that is generally obtained from different machining processes (Chang *et al.*, 1997), and (2) the ISO tolerance number (*IT-number* or ISO tolerance grade) assigned for different machining processes (Wilson, 2005). The former is applied when surface roughness is given to a manufacturing face while the latter is applied for a manufacturing face with a dimensional and geometric tolerance. Moreover, the most precise one will be chosen if two or more tolerances given to a manufacturing face lead to different machining processes. Figure 3.8 shows a decision-making diagram to determine a machining process when surface roughness is given to a manufacturing face. In Figure 3.8, a turning process based on the given surface roughness is chosen among the machining processes such as roughing, semi-roughing, finishing, fine turning, and external and internal threading. Grinding and honing are not included in this research. In particular, each referenced roughness value (*i.e.*, 63$\mu$in in Figure 3.8) means the roughness average ($R_a$) that is defined as the average deviation from the mean surface or arithmetical average.

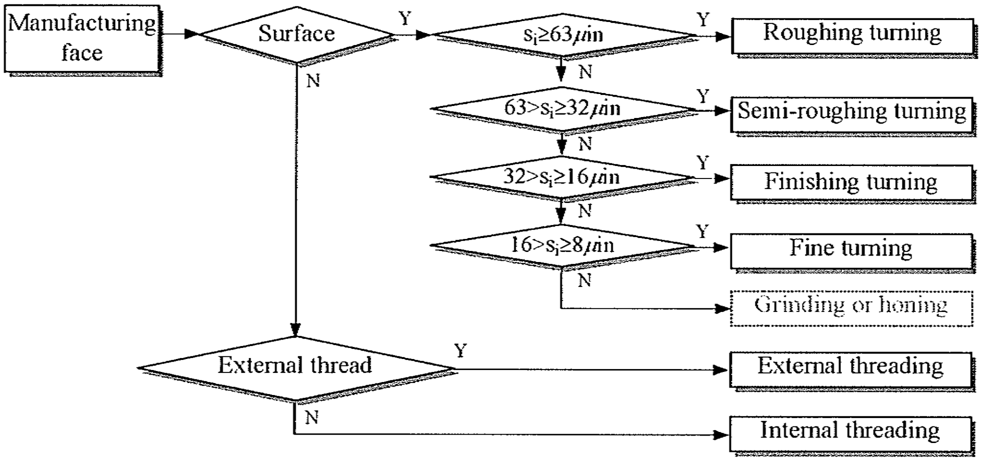


**Figure 3.8**: A decision-making diagram to determine a turning process for a given surface roughness.

For a dimensional or geometric tolerance to be given to a manufacturing face, it is necessary to determine the *IT-number* that is based on both the basic size of the face and its tolerance limitation. For instance, a manufacturing face shown in Figure 3.9 is assigned a dimensional tolerance, $d$=0.2 mm. The basic size of this face is $D$=124.9 mm. Then, the tolerance coefficient for this face can be first calculated as follows (Wilson, 2005).

$$\textit{tolerance coefficient} = \frac{d \cdot 10^{3}}{\left(0.45D^{\frac{1}{3}} + 0.001D\right)} \tag{3.4}$$

By Equation (3.4), the resulting coefficient≈84, and it corresponds to the *IT-number* 10 (*i.e.*, IT10) via the tolerance-grade conversion table (Refer to Appendix A.7). By this *IT-number*, the semi-roughing turning process is subsequently chosen in the machining process allocation table in Appendix A.7.

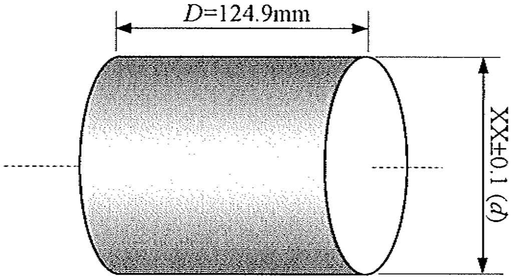


**Figure 3.9**: A manufacturing face with a dimensional tolerance to calculate the *IT-number*.

## 3.6 Tools and Machines Selection

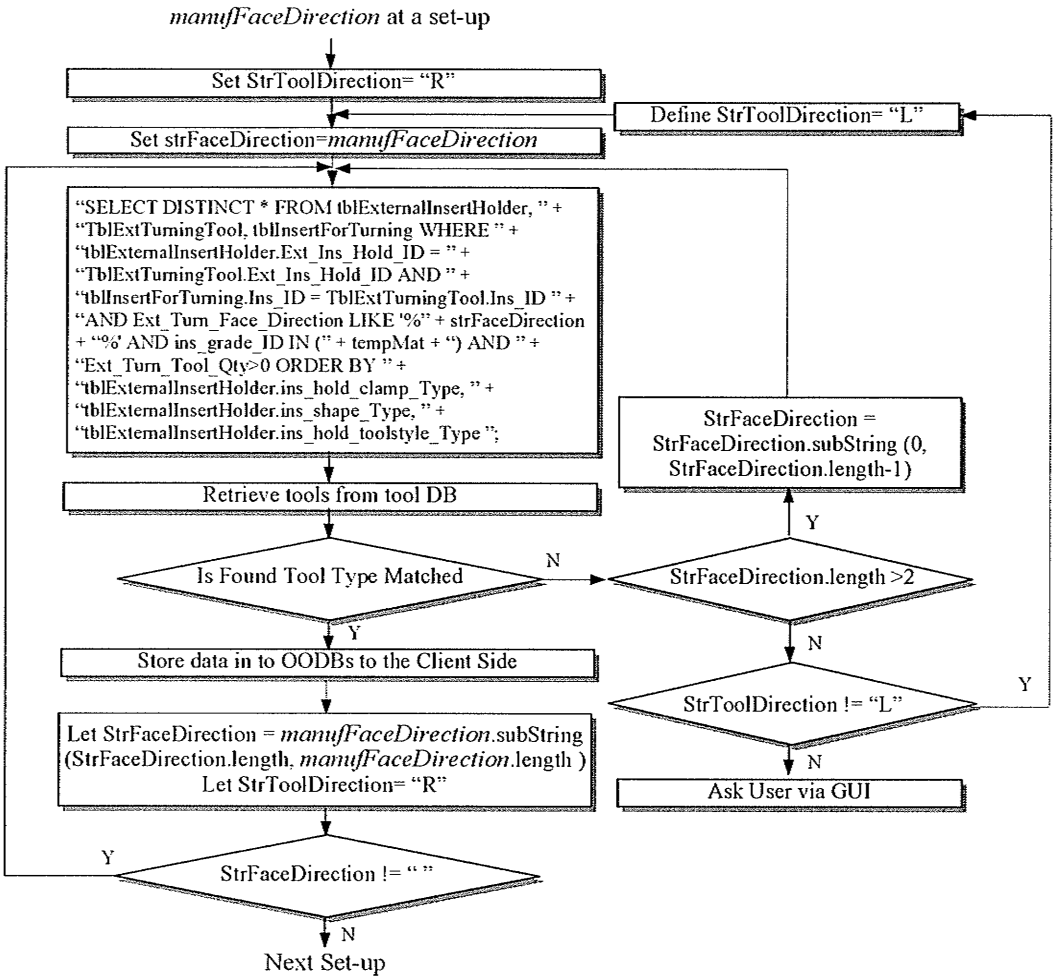
Tools and machines can be selected under static and dynamic manufacturing environments. Manufacturing resources including tools, machines and personnel at a shop floor are always available in a static environment, which was assumed by traditional CAPP systems. In a dynamic manufacturing environment, however, it is essential to consider resource availability and time schedules for a feasible process plan. Following dynamic selection of tools and machines is suggested to enable process planning to generate timely alternative plans for the workable and economical production.

## 3.6.1 Dynamic tools selection

### *3.6.1.1. Tool alternative retrieval from tool DB*

In dynamic tools selection, it is important to initially determine all possible tools to machine the manufacturing faces of a part. These tools are retrieved from the tool DB at a dynamic shop floor. However, the determined tools and their combinations may actually create a huge amount of complete tool sets to machine the whole part. In this research, two issues are considered to reduce the number of tools retrieved from the tool DB. The first consideration is to reduce the number of transactions with the DB server through the Internet, and the second one is to reduce the amount of data to be retrieved, which responds to user request. An efficient DB search strategy is required in order to satisfy these considerations.

DB search algorithms are developed to efficiently search the tool DB in retrieving tool alternatives for the manufacturing faces of a part. These algorithms are based on dynamic SQL queries and searching criteria defined in this research. Figure 3.10 shows a developed DB search algorithm to retrieve the tool alternatives for external manufacturing faces of a part. As shown in Figure 3.10, this algorithm starts with forming a manufacturing face direction (*i.e.*, *manufFaceDirection* in Figure 3.10) based on a set of external manufacturing faces in a part, which are expected to process with an external turning tool at one set-up. A formed manufacturing face direction is a set of direction vectors, and is represented by a set of the predefined numerals in this research. Moreover, this set of geometry-based vectors can be easily converted to the required directional constraints of an external turning tool that should be moved to machine the set of the external manufacturing faces.


manufFaceDirection at a set-up
Set StrToolDirection= "R"
Define StrToolDirection= "L"
Set strFaceDirection=manufFaceDirection
Retrieve tools from tool DB
Is Found Tool Type Matched
N
Y
StrFaceDirection = StrFaceDirection.subString (0, StrFaceDirection.length-1)
Y
StrFaceDirection.length >2
N
Store data in to OODBs to the Client Side
Y
StrToolDirection != "L"
N
Let StrFaceDirection = manufFaceDirection.subString (StrFaceDirection.length, manufFaceDirection.length )
Let StrToolDirection= "R"
Ask User via GUI
Y
StrFaceDirection != " "
N
Next Set-up

```
"SELECT DISTINCT * FROM tblExternalInsertHolder, " +
"TblExtTurningTool, tblInsertForTurning WHERE " +
"tblExternalInsertHolder.Ext_Ins_Hold_ID = " +
"TblExtTurningTool.Ext_Ins_Hold_ID AND " +
"tblInsertForTurning.Ins_ID = TblExtTurningTool.Ins_ID " +
"AND Ext_Turn_Face_Direction LIKE '%" + strFaceDirection
+ "%' AND ins_grade_ID IN (" + tempMat + ") AND " +
"Ext_Turn_Tool_Qty>0 ORDER BY " +
"tblExternalInsertHolder.ins_hold_clamp_Type, " +
"tblExternalInsertHolder.ins_shape_Type, " +
"tblExternalInsertHolder.ins_hold_toolstyle_Type ";
```

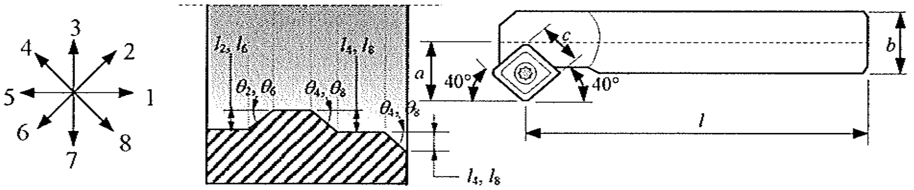
**Figure 3.10**: DB search algorithm to retrieve possible tool alternatives for external manufacturing faces of a part.

To retrieve the least number of tool alternatives from the tool DB, the manufacturing face direction is initially formed via all the external manufacturing faces of a part at one set-up. Then, this manufacturing face direction is inserted into a dynamic SQL query with other searching criteria including required tool material types and tool availability. The determination of the required tool material types is based on the material type of work-piece, hardness and required machining processes. The material type of work-piece and hardness are based on design requirements of a part while machining processes are determined by given tolerances discussed in previous Section 3.5. For efficiently supporting the determination process of a tool material, a tool manufacturer's handbook (Pramet Co., 2001) based on ISO 513 is used to collect related knowledge. The summary of knowledge tables are illustrated in Appendix A.8.

The availability of a turning tool is examined by checking the remaining quantity of the tool in the tool DB. As shown in Figure 3.10, a searching criterion like the *Ext_Turn_Tool_Qty>0* is simply inserted for this check. When a tool is reserved for a part to be manufactured, its quantity information in the tool DB can be reduced by one. The DML used for defining the dynamic SQL query shown in Figure 3.10 is also used for this update process (*i.e.*, UPDATE *TblExtTurningTool* SET *Ext_Turn_Tool_Qty* =*Ext_Turn_Tool_Qty*-1 WHERE *Ext_Turn_Tool_ID* = ...). On the contrary, this process can increase the quantity of a tool in the tool DB when the tool returns to an available state.

The dynamically generated SQL query shown in Figure 3.10 is sent to the DB server, and used to search appropriate external turning tools. If there is no tool matched in terms of the tool direction vector, particularly, the manufacturing face direction represented by numerals is reduced by a string operation. In Figure 3.10, this string operation is described by the Java expression, *StrFaceDirection.subString (0, StrFaceDirection. length-1)*. Thus, this reduction process is repeatedly executed until finding at least a matched tool. In this research, the tool direction vectors can be represented as a set of directions with which the tool moves in a machine. They are also pre-stored in the knowledge DB with constraints. The used tool direction vectors and constraints are based on tool holder types and insert shapes according to ISO. Therefore, all tools designated by the ISO standard can use this knowledge without any further modification.

Figure 3.11 shows a type of an internal turning tool with tool direction vectors and constraints in the knowledge DB. As shown in Figure 3.11, the tool type is *SCXC* classified by ISO. It has the tool direction vectors of *454565* and *121818* depending on the insert-holder direction types. Four constrains for this tool type are also listed in Figure 3.11. For instance, if a formed manufacturing face direction is represented as *4545*, it will be matched with the tool direction vectors of *454565* (*i.e.*, $4545 \subset 454565$). Then, a checking process to avoid a tool collision is executed for the four constraints of this tool shown in Figure 3.11, namely, the *Tool_Height* (*i.e.*, $a$+0.5$b$)<*Hole_Radius*, $\theta_4$, $\theta_6$= 40°, and $l_6$<$c$sin 40°.



| Tool type | Tool direction vectors | | Constraints |
|---|---|---|---|
| | R-type | L-type | |
| SCXC | 454565 | 121818 | $\begin{cases} \frac{hole\ Dia.}{2} > \left(a+\frac{b}{2}\right) \\ \theta_\xi, \theta_\zeta = 40^\circ; \sum l_\zeta < c\sin 40^\circ \end{cases}$<br>$Where, (\xi,\zeta) = \begin{cases} (4,6)\ for\ R-type \\ (2,8)\ for\ L-type \end{cases}$ |

**Figure 3.11**: An internal turning tool type with face direction patterns and constraints.

The notion applied in this research is expected to reduce the number of transactions and huge amounts of possible tool alternatives retrieved from the tool DB. This can also prevent a possible tool-path from interference between adjacent features, which may occur when a tool is recommended from only one manufacturing feature. Subsequently, recommended tools to machine a whole part consisting of manufacturing faces are stored in OODBs, named *TurningToolOODBs*. In particular, the *TurningToolOODBs* (Refer to Appendix A.9) store tool information and manufacturing faces to be machined at a set-up. These tool alternatives are further used to create complete tool sets to machine a whole part. Thus, the complete tool sets are a basis of performing the tools and machines selection process via weighting factors of production time and cost, which we will discuss later in this chapter.

### *3.6.1.2. Creation of complete tool sets based on the tool alternatives*

The creation of possible complete tool sets is based on all the tool alternatives retrieved for a whole part. These tool alternatives are stored in *TurningToolOODBs*. Because a huge amount of possible tool sets can be created, an algorithm is developed to efficiently generate complete tool sets using *TurningToolOODBs*. For instance, Figure 3.12 shows manufacturing faces to be machined at one set-up. As shown Figure 3.12, the manufacturing faces are grouped into *p*-sets of manufacturing faces. Each face set (*i.e.*, $i$=1, 2, ..., $p$) is machined by a tool type but it can have a number of tool alternatives with different types. In Figure 3.12, the number of tool alternatives for each face set is represented as $Nt(i)$. A total number of possible tool sets created is $Nt(1)\times Nt(2)\times\cdots\times Nt(p)$.

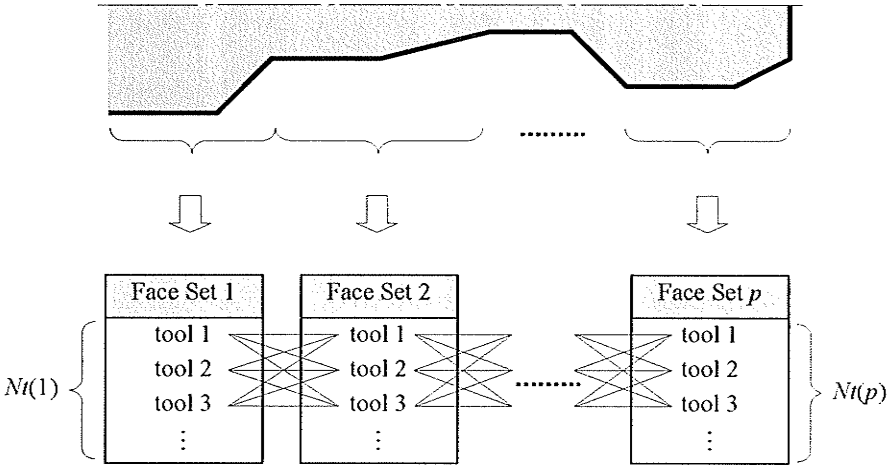


**Figure 3.12**: Creation of possible tool sets based on the tool alternatives.

Algorithm 3.1 is used to create possible complete tool sets that are based on the retrieved tool alternatives for manufacturing faces in one set-up. In Algorithm 3.1, the *TurningToolOODB*[][] represents the array of the *TurningToolOODB* classes to store tools, and possible tool sets (*i.e.*, *toolSet*[][]) store indexes of the *TurningToolOODB*[][] array. For the rest of the set-ups, this algorithm is repeatedly applied in the order of the set-up sequences discussed in Section 3.4. The *toolSet*[][] subsequently stores all the possible tool sets for a part.

**Algorithm 3.1**: Creation of possible tool sets

```
Input $TurningToolOODB[p][t_m]$ where $t_m = \max \forall Nt(i), i = 1, 2, \ldots, p$
Let $t_{all} = Nt(1) \times Nt(2) \times \cdots \times Nt(p)$
Let $toolSet\,[t_{all}][p]$
For $i = 0, \ldots, t_{all} - 1 \Rightarrow$
    Let $k = i$
    For $j = 0, \ldots, p - 1 \Rightarrow$
        $toolSet\,[i+1][j+1] = k \,\mathbf{mod}\, Nt(j+1) + 1$    /* mod : Integer Modulus */
        $k = k \setminus Nt(j+1)$                                  /* \ : Integer Division */
```

### 3.6.1.3. Tools selection

Once possible tool sets (*i.e.*, *toolSet*[][] in Algorithm 3.1) are created, the next step is to rank each tool set via weighting factors related to production time and cost. An optimized equation (Refer to Appendix A.10) is used for ranking, which is suggested by Usher and Fernandes (1999). In Appendix A.10, Equations (A.2) and (A.3), and Equation (A.4) are respectively used to calculate production time and cost, and to calculate the final score of a tool set. Later, the best tool set to machine a whole part is chosen by selecting one with the minimum score by Equation (A.4).

In particular, a total cutting length ($L$) is required to calculate machining time $\left(i.e., T_m = L/\left(f_r * N\right)\right)$ in Equation (A.2). The total cutting length is determined as a machining volume (MV) divided by the required cutting depth for a machining process. In this research, several algorithms are developed to build 2D MVs. As shown in Figure 3.13(a), there is a set of manufacturing faces to be machined at a set-up.

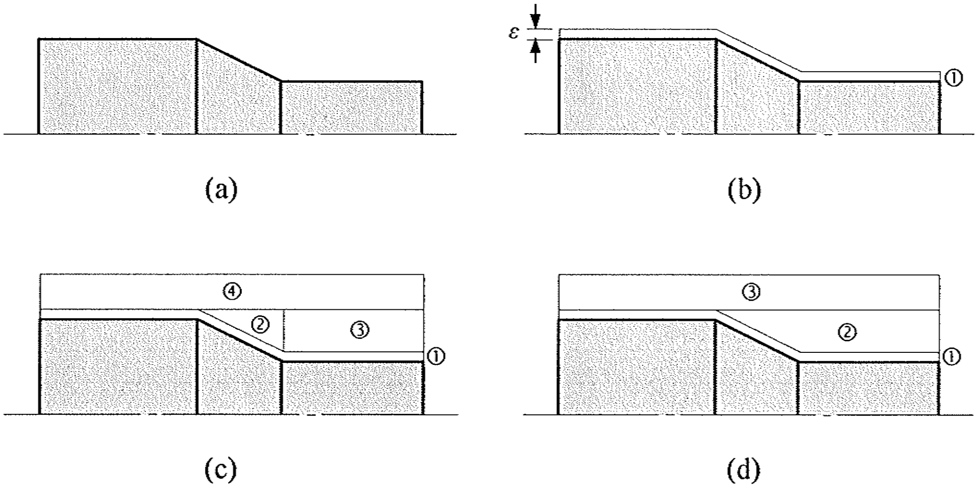


**Figure 3.13**: Machining volume generation in this research.

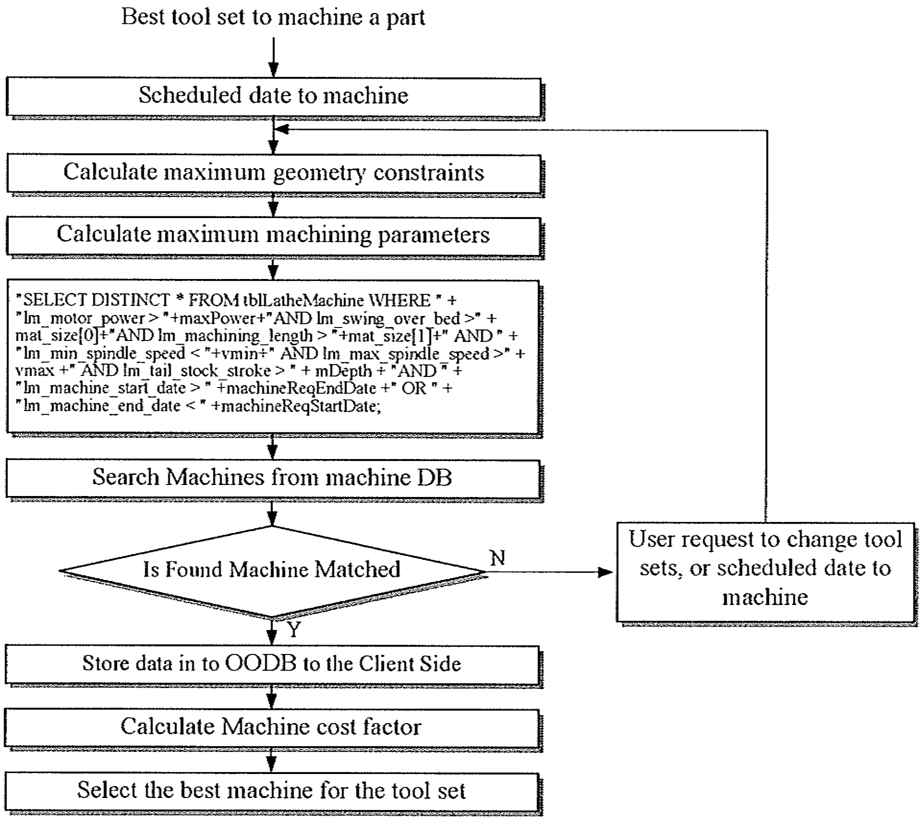
Contouring the geometry defined by the manufacturing faces, an MV is first generated for a fine or finishing process as shown in Figure 3.13(b). The depth ($\varepsilon$) in Figure 3.13(b) is determined here to allow three cuts for the generated MV. As shown in Figure 3.13(c) and (d), other MVs for rough cutting are subsequently generated and merged via the algorithms developed in this research.

In calculating the machining time in Equation (A.2) (Refer to Appendix A.10), the most rigid tool in a recommended tool set is first assigned to machine the generated MVs for rough cutting. Then, each tool in the tool set is used to machine the MV for the finishing process. The machining parameters such as depth of cut, feed rate, cutting speed and tool life are collected from a tool manufacturer's catalogues (Pramet Co., 2001), and stored in the knowledge DB. Appendix A.8 summarises knowledge tables used for calculating machining parameters, and presents an equation to calculate the optimized cutting speed using the knowledge tables.

### 3.6.2 Dynamic machines selection

The machine selection for a particular part is similar to the process of dynamic tools selection. A DB search algorithm is presented to efficiently search the machine DB using a dynamic SQL query and searching criteria defined in this research. It is also advantageous to reduce the number of available machines retrieved from the machine DB. Constraints used to select a machine include machine availability, maximum turning diameter and length, maximum spindle speed, maximum motor power, and tail stock stroke. These constraints are inserted into a dynamic SQL query to search appropriate machine alternatives for a tool set.

Figure 3.14 shows the developed DB search algorithm for dynamic machine selection.


Best tool set to machine a part
Scheduled date to machine
Calculate maximum geometry constraints
Calculate maximum machining parameters
Search Machines from machine DB
Is Found Machine Matched
N
User request to change tool sets, or scheduled date to machine
Y
Store data in to OODB to the Client Side
Calculate Machine cost factor
Select the best machine for the tool set

```
"SELECT DISTINCT * FROM tblLatheMachine WHERE " +
"lm_motor_power > "+maxPower+"AND lm_swing_over_bed >" +
mat_size[0]+"AND lm_machining_length > "+mat_size[1]+" AND " +
"lm_min_spindle_speed < "+vmin+" AND lm_max_spindle_speed >" +
vmax +" AND lm_tail_stock_stroke > " + mDepth + "AND " +
"lm_machine_start_date > " +machineReqEndDate +" OR " +
"lm_machine_end_date < " +machineReqStartDate;
```

**Figure 3.14**: DB search algorithm to retrieve possible machine alternatives.

For a given tool set, such constraints as the required maximum spindle speed and maximum motor power are determined via the knowledge tables and equations in Appendix A.8 and Appendix A.11. The knowledge tables and equation in Appendix A.11 are also summarised based on the tool manufacturer's handbook (Pramet Co., 2001). As a searching criterion, the machine availability is simply examined by checking the two time-related fields used in the machine DB, such as *lm_machine_start_date* and *lm_machine_end_date* of the *tblLatheMachine* table (Refer to Appendix A.1). These two fields are updated when a lathe machine is reserved for a part to be manufactured. The DML of SQL is used for this update process (*i.e.*, a UPDATE statement).

As shown in Figure 3.14, the best tool set ranking in Section 3.6.1.3 is first applied in the procedure of retrieving possible machine alternatives. If there is no machine matched, a user should change the time schedule of machine usage, or the second best tool set can be applied by the user. When the process is complete, machine alternatives selected for a tool set are stored in OODBs, named *MachineOODBs* (Refer to Appendix A.12).

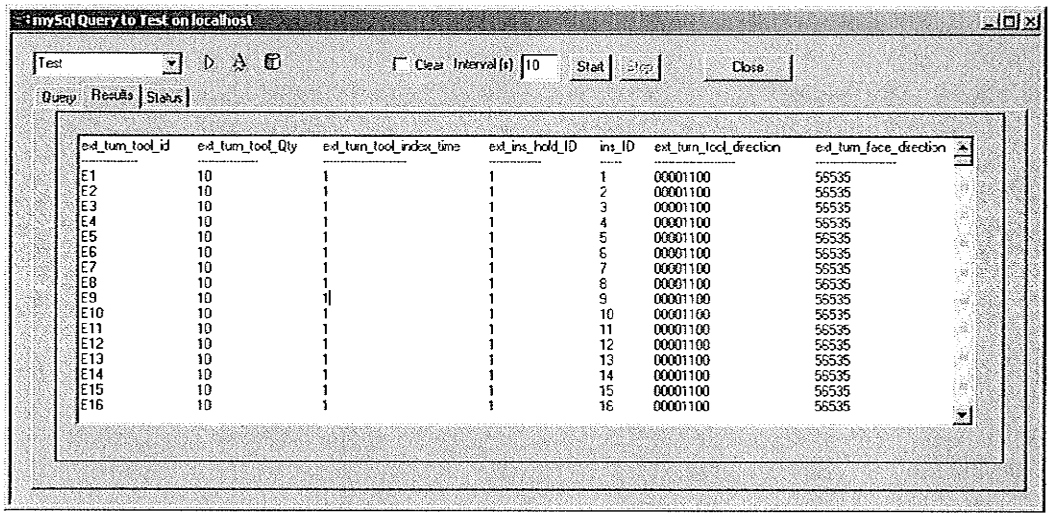
To select the best machine in *MachineOODBs* for a tool set, a machine cost model is developed. The machine cost model consists of two parts: machine capital cost and maintenance cost. Thus, the developed cost model for a single machine selection is described by Equation (3.5).

$$\begin{aligned} C_m &= \sum_i^q \left( \left( \frac{C_o}{\mathrm{EIDL}} + \frac{\mathrm{MTTR}}{\mathrm{MTBF}} \times P_{cm} \right) \cdot \mathrm{AOR} \right)_i + \sum_i^q \left( \sum_j^r \left( \mathrm{TF}_j \cdot \mathrm{ET}_j \right) \times P_{pm} \right)_i \\ &= \sum_i^q \left( \left( \frac{C_o}{\mathrm{EIDL}} + \frac{\mathrm{MTTR}}{\mathrm{MTBF}} \times P_{cm} \right) \cdot \mathrm{AOR} + \gamma \right)_i \\ &\cong \left( \frac{C_o}{\mathrm{EIDL}} + \frac{\mathrm{MTTR}}{\mathrm{MTBF}} \times P_{cm} \right) \cdot \mathrm{AOR} + \gamma \end{aligned} \tag{3.5}$$

In the above equation, the used symbols are listed at the beginning of this dissertation, *List of Notations*. Two system-effectiveness measures are used, mean time between failures (MTBF) and mean time to repair (MTTR). MTBF represents how often a machine is failing while MTTR represents the mean of the distributions of the time-intervals needed to repair an item. In Equation (3.5), the value of a machine is reduced annually, and it will eventually become zero when the machine life equals its design life called end item design life (EIDL). Therefore, this value can be normalized based on annual operation requirements (AOR). The maintenance cost is divided into preventive and corrective maintenance cost. With respect to scheduled preventive maintenance intervals, preventive maintenance cost can be defined as the annual cost to perform tasks such as adjustment, servicing, calibration and inspection without including any machine failure. On the other hand, corrective maintenance cost with respect to MTBF and MTTR for corrective maintenance can be defined as the cost to perform tasks such as repair, replacement, operational and functional tests, as well as the same tasks of preventive maintenance done with machine failure. Equation (3.5) is the result normalized by AOR to make the two categories have the same dimension.

## 3.7 Sample Applications

To evaluate the approach developed in this research, two examples of rotational parts are tested in this section. Tooling data for process planning were collected from the product catalogues (Pramet Co., 2001), which are stored in the tool DB. The tool DB contains 4,200 tools including external, internal, grooving and threading tools. For instance, external turning tools consisting of various external holders and inserts are listed in the MySQL query test window shown in Figure 3.15. This transaction result shown in Figure 3.15 is generated via a SELECT query based on the DML of SQL. From the tool DB, this query retrieves only the tool information satisfying its selection criteria. The data from five CNC machines are stored in the machine DB. However, most of information related to the machines is assumed in Table 3.1 because of the difficulty to get cost and design data such as EIDL, MTBF and MTTR.



```
ext_turn_tool_id  ext_turn_tool_Qty  ext_turn_tool_index_time  ext_ins_hold_ID  ins_ID  ext_turn_tool_direction  ext_turn_face_direction
----------------  -----------------  ------------------------  ---------------  ------  -----------------------  -----------------------
E1                10                 1                         1                1       00001100                 56535
E2                10                 1                         1                2       00001100                 56535
E3                10                 1                         1                3       00001100                 56535
E4                10                 1                         1                4       00001100                 56535
E5                10                 1                         1                5       00001100                 56535
E6                10                 1                         1                6       00001100                 56535
E7                10                 1                         1                7       00001100                 56535
E8                10                 1                         1                8       00001100                 56535
E9                10                 1                         1                9       00001100                 56535
E10               10                 1                         1                10      00001100                 56535
E11               10                 1                         1                11      00001100                 56535
E12               10                 1                         1                12      00001100                 56535
E13               10                 1                         1                13      00001100                 56535
E14               10                 1                         1                14      00001100                 56535
E15               10                 1                         1                15      00001100                 56535
E16               10                 1                         1                16      00001100                 56535
```

**Figure 3.15**: Transacted result of external turning tools in the tool DB.

**Table 3.1**: Assumed machine data.

| Machine model | Machine cost (C$) | EIDL (Hr) | MTBF (Hr) | MTTR (Hr) | Preventive maint. cost per Yr (C$/Yr) | Corrective maint. cost rate per hour (C$/Hr) |
|---|---|---|---|---|---|---|
| 10HC DAEWOO Horizontal Lathe | 80000 | 60000 | 100000 | 20 | 400 | 800 |
| TUR-630M | 30000 | 60000 | 30000 | 85 | 200 | 400 |
| CJK0620 NC Lathe | 40000 | 60000 | 20000 | 100 | 200 | 400 |
| CJK0632 NC Lathe | 50000 | 60000 | 20000 | 100 | 200 | 500 |
| CJK6125 NC Lathe | 55000 | 60000 | 20000 | 100 | 200 | 550 |

As shown in Table 3.1, EIDL of 60000 operation hours is given to all the machines equally, and machine cost is assumed in accordance with the complexity of the machines. The MTBF used in this research represents the mission MTBF (MIL-STD-756B, 1981). Based on this notion, it is assumed that the more complex machines are the more reliable, in accomplishing their missions although they require much more maintenance cost. This assumption is supported by the fundamental concepts of existing engineering design, such as redundancy, fail safe or damage tolerance design.

A similar assumption is used for MTTR, mean time to repair for corrective maintenance. The more expensive machines are normally set up for repair so that MTTR can be reduced. For instance, the self-diagnosis function, the built-in-test (BIT) and modularization for easy replacement can support this assumption. In spite of lower MTTR, the maintenance cost rate of the more expensive machines is much higher. The maintenance cost including preventive and corrective maintenance cost can consist of cost for diagnosis, spare parts and pay rate for skilled persons.

### 3.7.1 Sample application 1

As shown in Figure 3.16, an example part is tested to evaluate the approach developed. The part is made of the heat-resistant and creep resistant steel that is classified in the M1 group (Refer to Appendix A.8) based on ISO 513, and has a 300 HB of hardness. As shown in Figure 3.16, this part has four specified tolerance limits consisting of one geometric and three dimensional tolerance limits. In particular, the concentricity with the ∅0.05 mm tolerance is applied for the geometric tolerance in this example part. Based on the geometric entities obtained from the DXF file, OOMFs are defined via the developed algorithms in Section 3.3. As shown in Figure 3.17, 17 OOMFs are defined for this example, and 25 manufacturing faces are derived from the OOMFs.

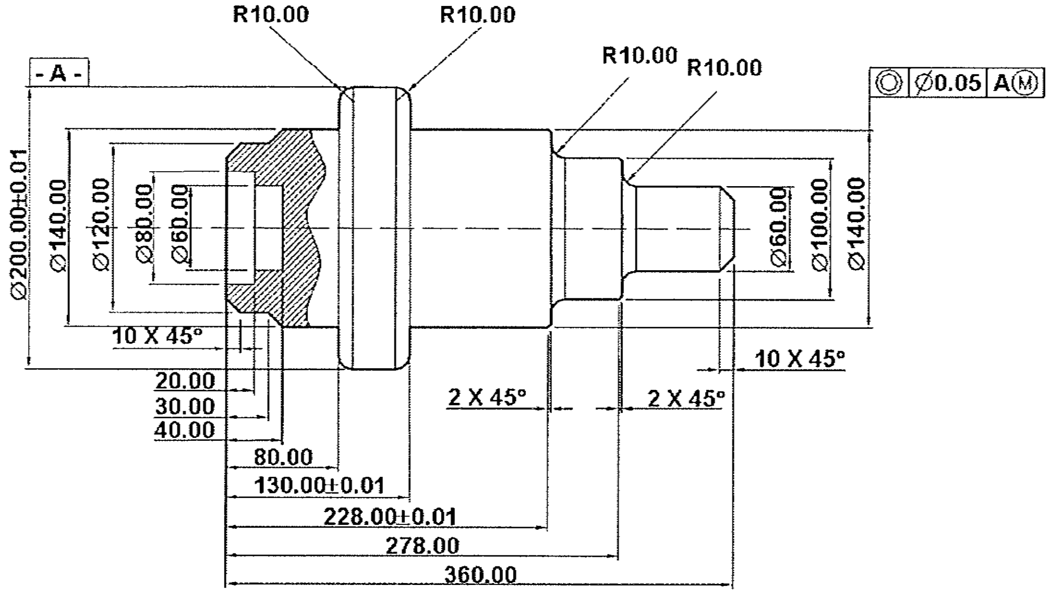
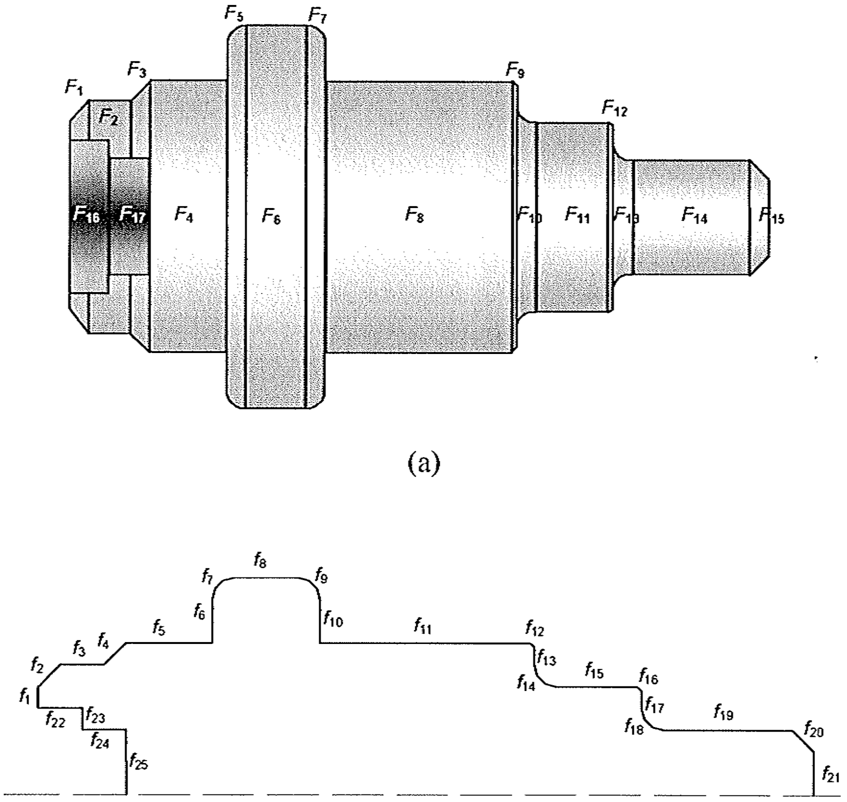


**Figure 3.16**: An example part.

The derived 25 manufacturing faces shown in Figure 3.17(b) are used to assign the geometric and dimensional tolerance limits given in Figure 3.16. Then, tolerance factors for the manufacturing faces are calculated to carry the out set-up planning and sequencing presented in Section 3.4. Thus, Figure 3.18 shows the tolerance factor graph based on the tolerance limits that are given to the example part. This graph is used to make its adjacency matrix *T* (Refer to Appendix A.6). The tolerance factor graph is a weighted, undirected graph ***G*** = (***V***, ***E***) representing the relative tolerance information of a part. The manufacturing faces within the part are represented as the vertices $V = \{f_1, f_2, \cdots, f_n\}$. The relative tolerance information is represented using the weighted edges $E = \{e_1, e_2, \cdots, e_m\}$. If the value of a weighted edge between two manufacturing faces $f_i$ and $f_j$ is small, it might be better if these two faces belong to the same set of sequences to be machined. Thus, the machining of the two manufacturing faces at one set-up is likely to satisfy the given tight tolerance requirement.



(a)

(b)

**Figure 3.17**: Definition of OOMFs and manufacturing faces for the example part: (a) 17 OOMFs generated from the DXF file, and (b) 25 manufacturing faces derived from the OOMFs.

As shown in Figure 3.18, the manufacturing faces $f_1$ and $f_{10}$ have the small value of the weighted edge. However, the two manufacturing faces cannot be machined at one set-up because of their different tool approach directions. As shown in Figure 3.17(b), the manufacturing face $f_1$ can be machined by a left-approaching tool while the manufacturing face $f_{10}$ requires a right-approaching tool. Similarly, the manufacturing faces $f_1$ and $f_{13}$ in Figure 3.18 cannot also belong to the same set-up sequence. On the other hand, it is highly recommended that the manufacturing faces $f_8$ and $f_{11}$ be put into the same set-up sequence.

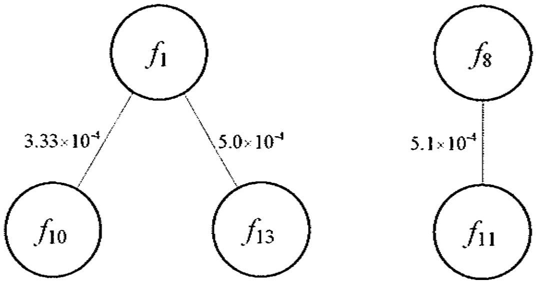


**Figure 3.18**: Tolerance factor graph based on tolerance limits that are assigned to the example part.

Based on the set-up planning and sequencing algorithm used, two set-up sequences are generated with datum information. Table 3.2 describes the generated set-up sequences for the example part shown in Figure 3.16.

**Table 3.2**: Generated set-up sequences for the example part.

| Set-up ID | Datum | Manufacturing faces |
|---|---|---|
| 1 | $f_8, f_{21}$ | $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_{22}, f_{23}, f_{24}, f_{25}$ |
| 2 | $f_1, f_5$ | $f_{21}, f_{20}, f_{19}, f_{18}, f_{17}, f_{16}, f_{15}, f_{14}, f_{13}, f_{12}, f_{11}, f_{10}, f_9, f_8$ |

In Table 3.2, the first set-up sequence includes both external and internal turning processes while the second set-up sequence includes only an external turning process. As discussed before, the two manufacturing faces $f_8$ and $f_{11}$ in Table 3.2 belong to the second set-up sequence of the set-up planning and sequencing process.

In addition, manufacturing faces at each set-up in Table 3.2 are used to form a manufacturing face direction to search tool alternatives via the DB search algorithm, which is discussed in Section 3.6.1.1. For instance, an external manufacturing face direction is initially generated with manufacturing faces $f_1$, $f_2$, $f_3$, $f_4$, $f_5$, $f_6$ and $f_7$, and its numeral values become *345453C* in the right-approaching tool direction, where *C* represents CCW. This implies that a tool cuts all the external faces at the first set-up. Then, the face direction is inserted in a dynamic SQL query with other searching criteria including the required tool material type. This SQL query is sent to the DB server, and used to search the matched turning tool alternatives. If there is no tool matched, the number of manufacturing faces is reduced, and a new SQL query is formed to search the matched tool alternatives for the reduced manufacturing faces.

Possible tool sets are created with the tool alternatives that are retrieved via the developed DB search algorithm. These tool sets are used for the ranking process (Refer to Appendix A.10) including the creation of MVs based on the set-up sequences in Table 3.2. In particular, four user-defined parameters are used to this ranking process, which are: (1) tool set-up time: 12 min; (2) pay rate: 0.2 C$/min; (3) weighting factor for production cost: 0.5; and (4) weighting factor for production time: 0.5. The calculation of production time for the ranking process uses standard tool-indexing time that is pre-stored in tool DB. This indexing time depends on types of tool inserts and tool holders. Figure 3.19 shows the best tool set selected to machine the example part via the ranking process.

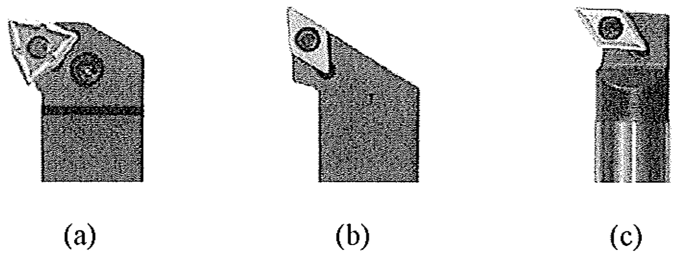
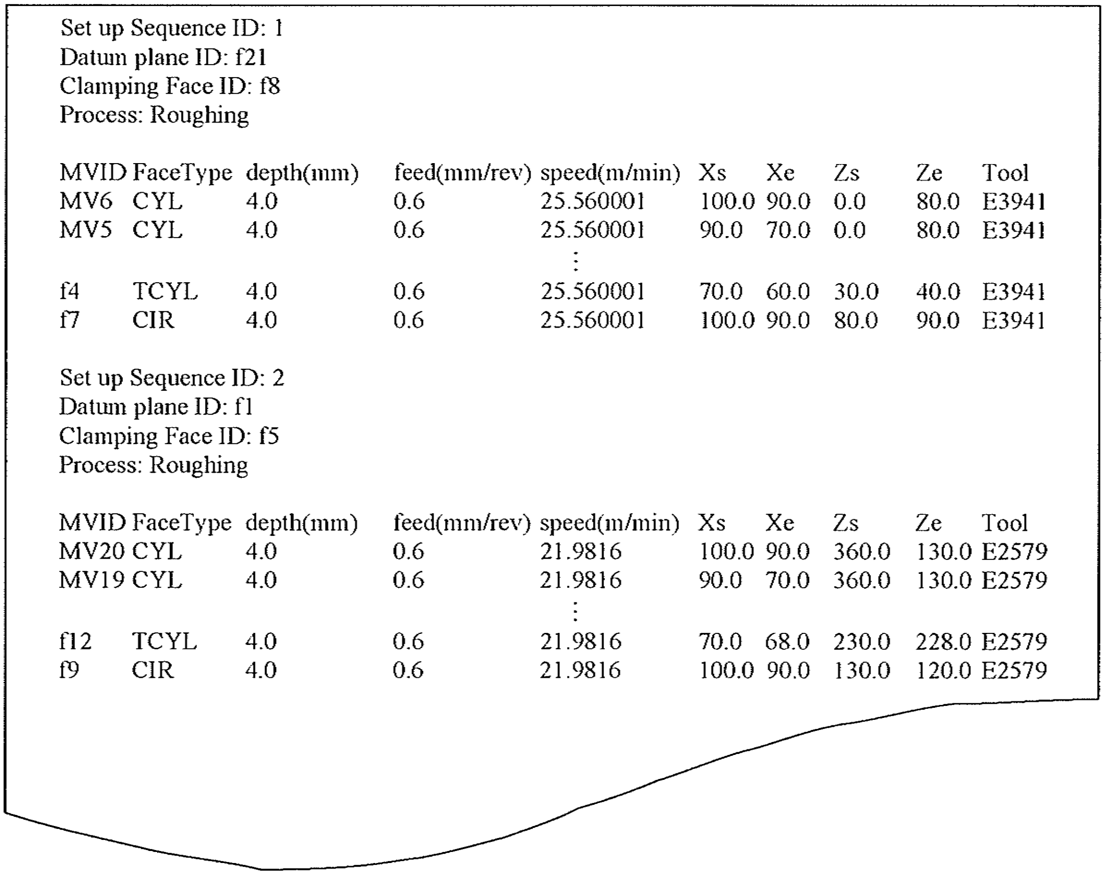


**Figure 3.19**: Best tool set selected for the example part (a) *E3941*: PWLNR-1616H06-S, (b) *E2579*: SDJCR-1616H11-S, and (c) *I164*: SDUCR07-S.

In Figure 3.19, there is a tool set consisting of three turning tools. This tool set ranks first in possible tool sets. As an external turning tool, the tool *E3941* shown in Figure 3.19(a) is selected based on the manufacturing faces $f_1$, $f_2$, $f_3$, $f_4$, $f_5$, $f_6$ and $f_7$ in Figure 3.17(b). At the first set-up in Table 3.2, this tool is combined with the internal turning tool *I164* shown in Figure 3.19(c). The tool *E2579* shown in Figure 3.19(b) is selected based on all the manufacturing faces at the second set-up in Table 3.2.

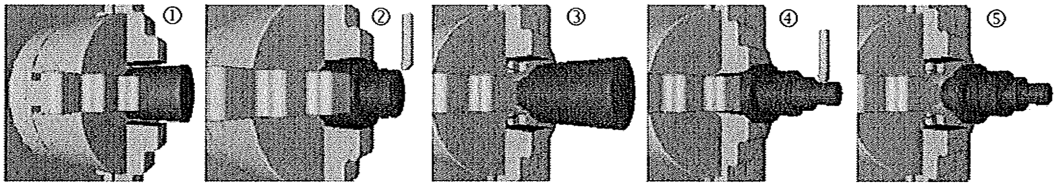
All the selected tools in Figure 3.19 result in the use of the same insert grade of 836, TiN multi-layer coated via the physical vapour deposition (PVD) process. Their tool-clamping types result in the use of big rigidity tools such as *P*- and *S*- types, which allow a stable machining process. The used inserts, *W* and *D* types, are versatile and economical so that they can reduce production time and cost by minimizing tool set-up time.

In the machines selection with the tool set shown in Figure 3.19, the 10HC DAEWOO horizontal lathe is suggested as the most economical machine despite the fact that its machine capital cost and maintenance cost are the most expensive among the machines in the machine DB. This results from the highest MTBF and the lowest MTTR that 10HC DAEWOO horizontal lathe has among the machines. However, the appropriateness of assumed data in machines selection should be reviewed and updated. Based on detailed design and cost data, this can be analyzed by traditional procedures such as failure modes, effects and criticality analysis, reliability and maintainability analysis, and maintenance task analysis. Subsequently, Figure 3.20 shows a generated process plan for this example part, and its simulation process. In particular, the simulation is executed using the system developed in this research, and its architecture and implementation are later discussed in Chapter 8.



```
Set up Sequence ID: 1
Datum plane ID: f21
Clamping Face ID: f8
Process: Roughing

MVID FaceType depth(mm)   feed(mm/rev) speed(m/min) Xs    Xe   Zs    Ze   Tool
MV6  CYL      4.0         0.6          25.560001    100.0 90.0 0.0   80.0 E3941
MV5  CYL      4.0         0.6          25.560001    90.0  70.0 0.0   80.0 E3941
                                           ⋮
f4   TCYL     4.0         0.6          25.560001    70.0  60.0 30.0  40.0 E3941
f7   CIR      4.0         0.6          25.560001    100.0 90.0 80.0  90.0 E3941

Set up Sequence ID: 2
Datum plane ID: f1
Clamping Face ID: f5
Process: Roughing

MVID FaceType depth(mm)   feed(mm/rev) speed(m/min) Xs    Xe   Zs    Ze    Tool
MV20 CYL      4.0         0.6          21.9816      100.0 90.0 360.0 130.0 E2579
MV19 CYL      4.0         0.6          21.9816      90.0  70.0 360.0 130.0 E2579
                                           ⋮
f12  TCYL     4.0         0.6          21.9816      70.0  68.0 230.0 228.0 E2579
f9   CIR      4.0         0.6          21.9816      100.0 90.0 130.0 120.0 E2579
```

(a)



(b)

**Figure 3.20**: Generated process plan and its virtual simulation: (a) process plan, and (b) its virtual simulation (① the 1st set-up ② rough-turning process at the 1st set-up ③ the 2nd set-up ④ rough-turning process at the 2nd set-up, and ⑤ completion of the simulation).

### 3.7.2 Sample application 2

As shown in Figure 3.21, another example part is tested to evaluate the approach developed. The part is made of carbon steel non-alloyed that is classified as the P1 group (Refer to Appendix A.8) based on ISO 513, and has 300 HB of hardness.

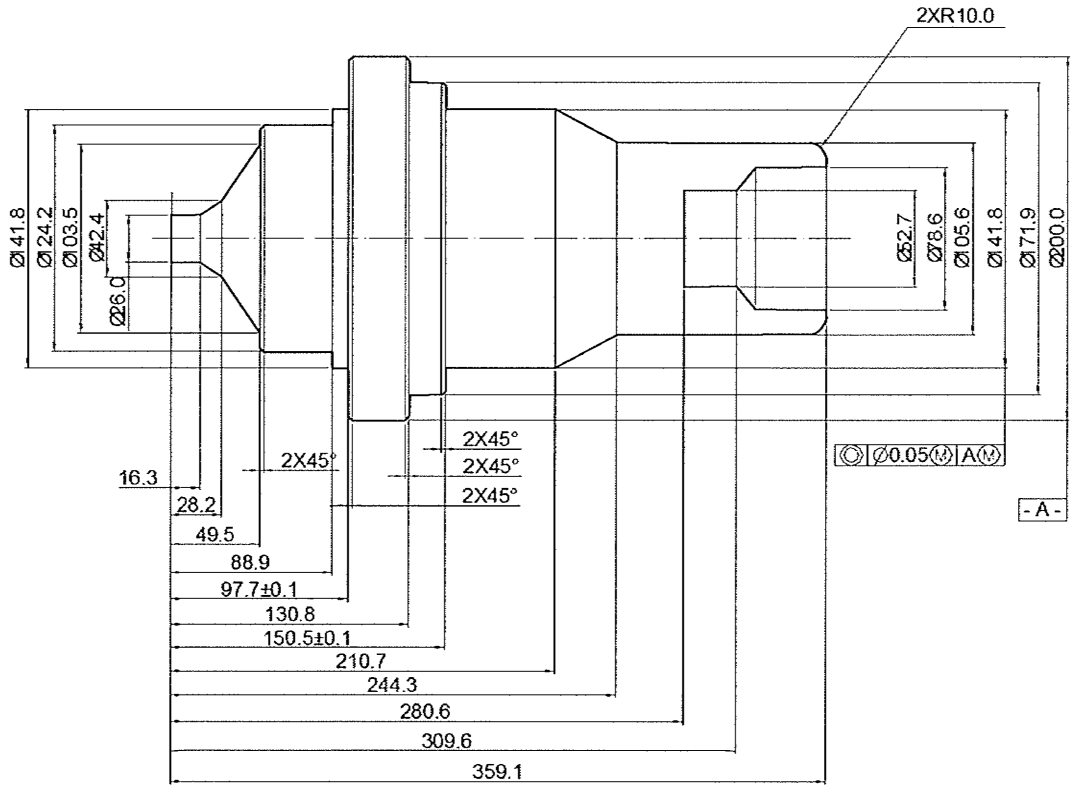
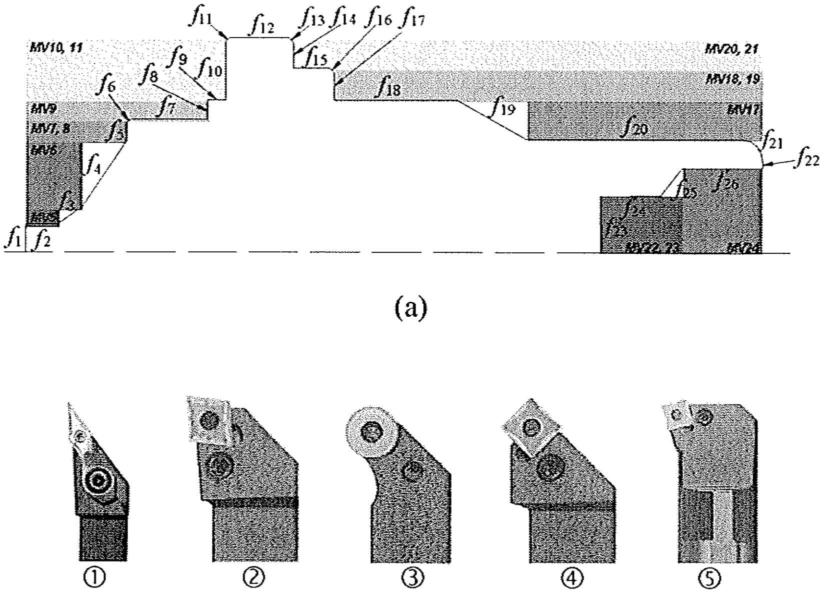


**Figure 3.21**: An example part.

This part is converted to 18 OOMFs, and 26 manufacturing faces are derived from the OOMFs. The 26 manufacturing faces are used to assign the geometric and dimensional tolerance limits given in Figure 3.21. After the set-up planning and sequencing processing, and tools and machines selections, a process plan is generated with a recommended tool set and a machine. The same resources and parameters as those used in the sample application 1 are applied in the example. Figure 3.22 shows the summary of the process planning for the part illustrated in Figure 3.21. As shown in Figure 3.22(b), in particular, five tools with the first rank in the created possible tool sets are selected, and their inserts are *V*-, *C*- and *S*- types that are versatile and economical to reduce production time and cost. The used insert grades of these tools are 320P or 525P, TiC/TiCN/TiN multi-layer coated through the chemical vapour deposition (CVD) process. In the machines selection, the 10HC DAEWOO horizontal lathe is also suggested as the most economical machine because of the minimal maintenance cost.



(a)

(b)

| Setup ID | Clamp | Datum | Sequence of operations | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Seq. ID | Machining Volume | Tool ID | Speed (m/min) | Feed Rate (mm) | Cut-depth (mm) |
| 1 | $f_{12}$ | $f_{22}$ | 1-1 | MV 10, 11 | E760 | 138 | 0.6 | 4.0 |
| | | | 1-2 | MV 9 | | | | |
| | | | 1-3 | MV 7, 8 | | | | |
| | | | 1-4 | MV 6 | | | | |
| | | | 1-5 | MV 5 | | | | |
| | | | 1-6 | $f_3, f_4$ | | | | |
| | | | 1-7 | $f_6, f_{11}$ | E2382 | 138 | 0.6 | 4.0 |
| 2 | $f_9$ | $F_1$ | 2-1 | MV 20, 21 | E1450 | 138 | 0.6 | 4.0 |
| | | | 2-2 | MV 18, 19 | | | | |
| | | | 2-3 | MV 17 | | | | |
| | | | 2-4 | $f_{19}$ | | | | |
| | | | 2-5 | $f_{16}, f_{13}$ | E2382 | 138 | 0.6 | 4.0 |
| | | | 2-6 | $f_{21}$ | E2148 | 109 | 0.6 | 4.0 |
| | | | 2-7 | MV 24 | I59 | 138 | 0.6 | 4.0 |
| | | | 2-8 | MV 22, 23 | | | | |
| | | | 2-9 | $f_{25}$ | | | | |

(c)

**Figure 3.22**: Summary of the process planning for the part illustrated in Figure 3.21: (a) generated MVs, (b) selected tool set (①*E760*: MVJNR-2525M16-S, ②*E1450*: PCLNR-2020K12-S, ③*E2148*: PRSCR-3225P16-S, ④*E2382*: PSSNR-2020K12-S, and ⑤*I59*: A32S-PSKNR12-S), and (c) detailed process plan.

## 3.8 Chapter Summary

This chapter presented an approach to dynamic process planning, which proposes to generate a high-fidelity process plan, and tool and machine alternatives based on a dynamic manufacturing environment. To achieve the objective, *dynamic tools and machines selection* was introduced to existing issues in static CAPP systems, which are namely *manufacturing feature recognition*, *set-up planning and sequencing*, and *machining process determination*.

For the *dynamic tools and machines selection* discussed in this chapter, DB search algorithms and organized knowledge were developed and implemented via RDB technologies. The RDB technologies helped to efficiently define and transact a huge amount of tools- and machines-related information in the DB server. Moreover, various OODBs based on the OOP technique were used to efficiently manipulate the retrieved tools- and machines-related information. The OODBs were subsequently applied in the tools and machines selection process that is based on weighing factors of production cost and time. Once tools and machines are selected for the process plan of a part, they are no longer available until the part is completely manufactured at the shop floor. A UPDATE statement based on the DML of SQL was used to change the availability state of the selected tools and machines in DBs. It simply set DB fields related to tool quantity and date for machine usage.

Based on the approach developed in this chapter, it is likely to achieve dynamic process planning in a Web-based PD&M environment. A high-fidelity process plan can be generated with its feasible tool and machine alternatives in a networked collaborative environment. The generated process plan can be used for quickly exploiting manufacturing opportunities for a specific product at the inter-enterprise level, and for realizing fabrication at the shop floor level. In addition, the process plan can provide VP or VM systems with a reduced number of feasible *what-if* scenarios for fabrication, and serve at supporting DFM to execute the practical assessment of manufacturability, and the reliable estimation of manufacturing time and cost. The reliable estimation is efficiently achieved by using the selected tools and machines, which are used to calculate machining cycle data such as the depth of cut, speed, feed rate and cutting time.

With respect to the research on the dynamic process planning, published papers in international journals and conferences are as follows.

Peng, Q. and **Chung, C.** (2006). A visualised CAPP System for Agile Manufacturing, Special Issue on Manufacturing Under Changing Environment, International Journal of Manufacturing Technology and Management (In Press).

**Chung, C.** and Peng, Q. (2004). The selection of tools and machines on Web-based manufacturing environments. International Journal of Machine Tools and Manufacture (IJMM) 44(2-3): 315-324.

**Chung, C.** and Peng, Q. (2003). The development of a Web-centred VR-CAPP system. Proceedings of the International Conference on Agile Manufacturing, Beijing, China, pp.369-376, ICAM2003, ISBN 7-111-13395-1, China Machine Press.

# Chapter 4

# Fast Assembly Tool Reasoning

This chapter presents a novel approach to geometric accessibility analysis for fast assembly tool reasoning. The approach is based on a parameterized assembly tool and a GAC$^d$ that approximates the obstacles surrounding a fastener being assembled or disassembled. It also works as a means for assembly tool feasibility analysis. The analysis will be further applied to dynamic assembly and disassembly planning, which are presented in Chapters 6 and 7, respectively.

## 4.1 Proposed Approach

The selection of proper tools to be used is an important factor in planning a complete assembly or disassembly sequence. An incomplete plan generated may lead to re-tooling, special tooling, or design changing at the production stage. A key tooling consideration in assembly or disassembly planning is the providing of available space for tool applications during a product assembly or disassembly. Compared to other reasoning methods developed for tools in machining or measurement, assembly tool reasoning about available space is a relatively under-explored issue. Some existing methods are computationally expensive to implement. Current assembly tool reasoning about space is mainly executed via physical simulation, virtual simulation, or interactive user-knowledge. These approaches are not efficient in dealing with various *what-if* scenarios regarding assembly or disassembly in a rapid product development. They also tend to depend on users' expertise or experience.

This chapter deals with the geometric accessibility analysis for fast assembly tool reasoning. Given solid models of a product consisting of parts and fasteners, the goal is to quickly determine whether an assembly tool is feasible or not. Moreover, a system linked with a DB is implemented to achieve a fast assembly tool reasoning in a dynamic manufacturing environment.

Feasible assembly tools are chosen based on the constraints of a given part configuration, the fastener type and tool availability at the dynamic shop floor. Then, the system developed in this research simulates the tool applications in a computer generated 3D environment. The simulation can clearly verify whether the tools are appropriate or not in assembling or disassembling a fastener. The assembly-tool reasoning method also works as a means for assembly-tool feasibility analysis. The analysis will be further applied to dynamic assembly and disassembly planning, which are presented in Chapters 6 and 7, respectively.

## 4.2 Global Accessibility Cone with Depth

Based on the concept of discrete GACs (Spitz *et al.*, 1999; Spitz and Requicha, 2000), a modified GAC is proposed in this research, which is named GAC$^d$ shown in Figure 4.1.

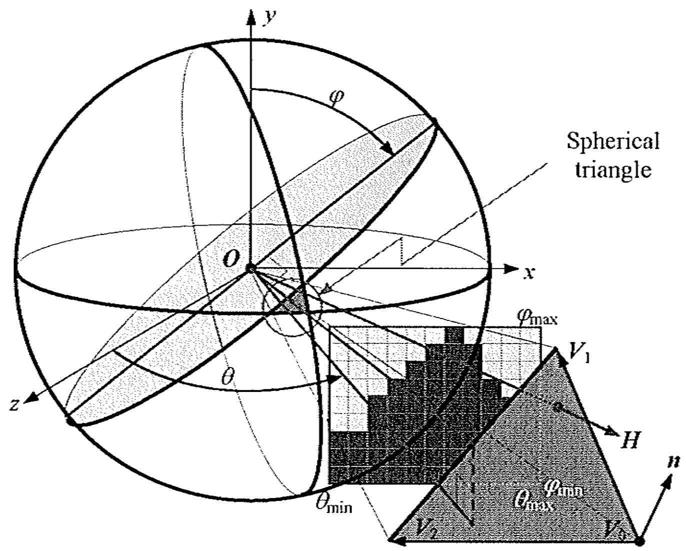


**Figure 4.1**: Construction of a GAC$^d$.

A GAC$^d$ represents approximated surroundings around a fastener. The surroundings are based on the geometric configuration of parts in a product. The GAC$^d$ consists of 180×360 pixels to make a total of 64,800 directions on a discrete unit sphere. The number of pixels is exactly matched with 180 colatitude angles ($\varphi$) and 360 longitude angles ($\theta$) in the Spherical coordinate system. Therefore, there is a one-to-one mapping between directions in the GAC$^d$ and unit vectors in the Spherical coordinate system. Thus, $\varphi$ and $\theta$ in this research are used to calculate a unit vector and to indicate the direction at an equivalent pixel ($\theta,\varphi$). In the Spherical coordinate system, a unit vector based on a pixel ($\theta,\varphi$) is simply determined as follows.

$$\hat{T}(\theta,\varphi) = (\sin\varphi\sin\theta)\hat{i} + (\cos\varphi)\hat{j} + (\sin\varphi\cos\theta)\hat{k} \tag{4.1}$$

To form a GAC$^d$, the initial position of a fastener becomes the centre point of a sphere, and the fastener removal direction is aligned to its $y$-axis. Then, depth information is determined for each direction on a GAC$^d$ by projecting a unit vector onto a face of a surrounding part or obstacle. However, this simple projection is a time-consuming process in a complex product with many parts. For ease to calculate the depth, a part is referred to as a set of triangle patches, which is widely used for solid representation in computer graphics.

The closest triangle patch to the centre point of a GAC$^d$ is first considered to calculate the depths towards directions on the GAC$^d$. The triangle patch is mapped as a spherical triangle (Harris and Stocker, 1998) shown in Figure 4.1 if its normal vector meets the projected vector onto it in an angle greater than 90 degrees. The spherical triangle is formed by connecting three points mapped on the surface of the GAC$^d$ with great circles (Harris and Stocker, 1998) passing through the points.

Based on six points including three vertices and midpoints at the three sides of the formed spherical triangle, four pixel boundaries, $\theta_{\min}$, $\theta_{\max}$, $\varphi_{\min}$ and $\varphi_{\max}$ on a GAC$^d$ are defined by the general equation in the Spherical coordinate system. Then, every unit vector $\hat{T}(\theta, \varphi)$ within the pixel boundaries is projected onto the triangle patch in 3D space, and used to find an intersection point. The intersection point $X$ on a triangle patch can be determined as follows.

$$X = \frac{[\{(\boldsymbol{O} - \boldsymbol{V_0}) \cdot \boldsymbol{n}\}\boldsymbol{H} - \{(\boldsymbol{H} - \boldsymbol{V_0}) \cdot \boldsymbol{n}\}\boldsymbol{O}]}{[(\boldsymbol{O} - \boldsymbol{V_0}) \cdot \boldsymbol{n} - (\boldsymbol{H} - \boldsymbol{V_0}) \cdot \boldsymbol{n}]} \tag{4.2}$$

Where, $X = x_1\hat{i} + x_2\hat{j} + x_3\hat{k}$, $\boldsymbol{O} = 0\hat{i} + 0\hat{j} + 0\hat{k}$, $\boldsymbol{H} = \infty\hat{T}(\theta, \varphi)$ and $\boldsymbol{n}$ is the normal vector of the triangle patch as shown in Figure 4.1. Subsequently, the determined intersection point $X$ above is tested as to whether it is inside on the triangle patch or not. This test is executed for three vertices $V_0$, $V_1$ and $V_2$ on the triangle patch; one of them is expressed as follows (Vince, 1995).

$$\begin{cases} 0 & \textit{if } (X - V_0) \cdot \{(V_2 - V_0) \times \boldsymbol{n}\} \geq 0 \\ 1 & \textit{otherwise} \end{cases} \tag{4.3}$$

If the intersection point $X$ is inside the triangle patch, which means that a return value is equal to zero by Equation (4.3), the direction at a pixel $(\theta, \varphi)$ within the pixel boundaries is not accessible from the outside. In this case, the depth between the point $X$ and centre point is calculated and stored at the pixel $(\theta, \varphi)$ of a GAC$^d$. Once directions defined at pixels in a GAC$^d$ are given depth information, the pixels are not used in the above process for another triangle patch. A direction defined at a pixel $(\theta, \varphi)$ here keeps only the shortest depth to the centre point of the GAC$^d$ from an obstacle. This reduces the computation time to form a GAC$^d$ for a complex product. Subsequently, all the triangle patches of parts in a product are repeatedly used to form the GAC$^d$ for a fastener. Figure 4.2 shows an example of a GAC$^d$ constructed around a fastener and its feasibility map with the same pixel size of the GAC$^d$.

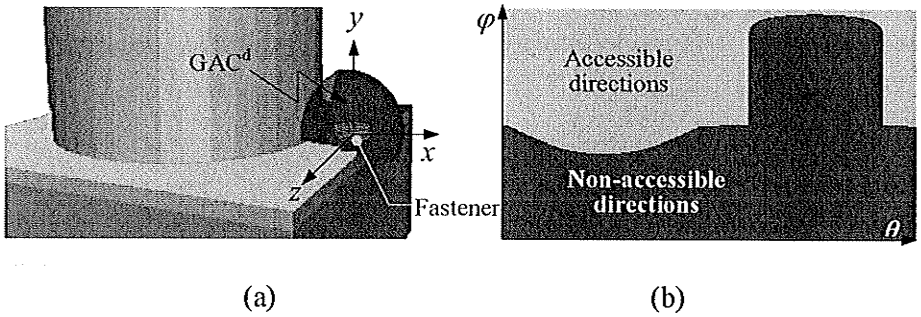


**Figure 4.2**: Example of (a) a GAC$^d$ formed around a fastener, and (b) its feasibility map.

As shown in Figure 4.2(a), the region of the GAC$^d$ represents a set of accessible directions that are defined by angles $\varphi$s and $\theta$s. The intersection region (*i.e.*, hidden region of the sphere) represents a set of non-accessible directions that block access from outside towards these directions. In this case, the shortest depth information is determined at each pixel $(\theta,\varphi)$ in the non-accessible directions.

## 4.3 Tool Representation and Classification

In this research, an assembly tool is defined as a set of four articulated devices: an effective handle, a head, an extension and a base. For instance, the effective handle includes the maximum length of a tool handle, not involving the portion that is grasped and applied by human hands or robot arms. The base is the tool-end portion occupying and contacting a fastener head. Based on these descriptions, a parameterized assembly-tool can be efficiently defined, and the parameters used to define the assembly-tool can be derived. As shown in Figure 4.3, 15 geometric parameters are used to represent an assembly tool, which are $l_e$, $w$, $h$, $r_a$, $r_e$, $r_b$, $e_a$, $e_e$, $e_b$, $e_f$, $e_h$, $e_n$, $\alpha_{\min}$, $\alpha_{\max}$ and $\beta$.

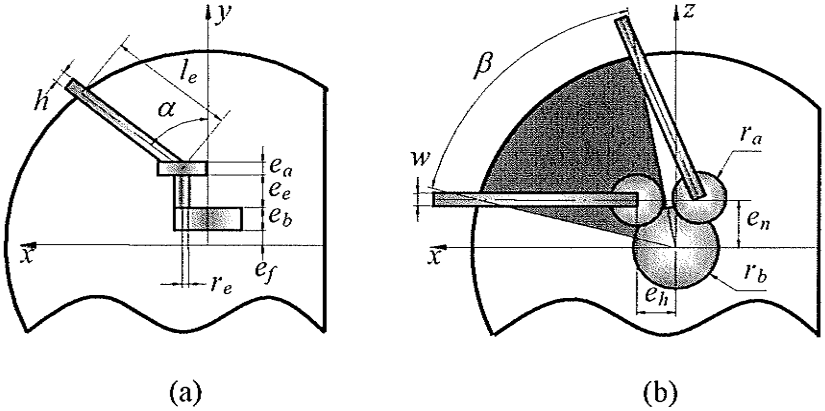


**Figure 4.3**: Tool representation used in this research; (a) *xy*-plane and (b) *xz*-plane views.

According to tool motions, all assembly tools used are classified into two categories: fastener-axis action tools (FATs) and tool-axis action tools (TATs). TATs move or rotate about their axes during assembling or disassembling a fastener, including such tools as screwdrivers, power tools and speeders with sockets. FATs rotating about the fastener axis include various wrenches and ratchets with sockets. This tool classification is advantageous for analyzing a tool application moving or rotating relative to a fastener, and it subsequently provides a searching pattern within a GAC$^d$. Figure 4.4 shows the assembly tool classification defined in this research. In particular, Figure 4.4(c) and (d) show real assembly tools classified by this definition. These tools are collected from a tool catalogue by Snap-on Company (2004).

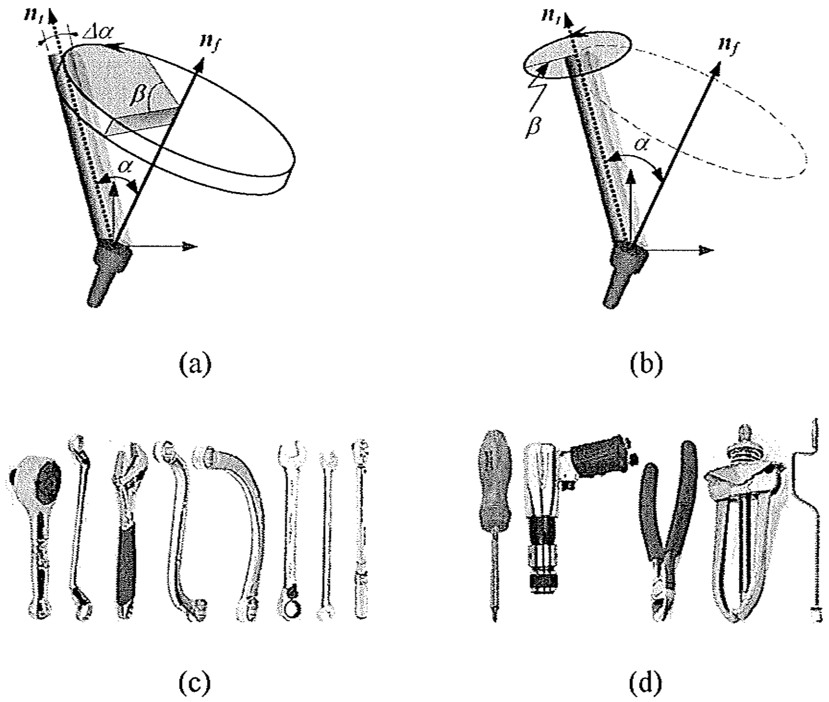


**Figure 4.4**: Tool classification used in this research: (a) FAT action (b) TAT action (c) various tools classified as FATs, and (d) various tools classified as TATs.

## 4.4 Design of Assembly Tool DB

Based on the RDB representation (Watson, 1999), the assembly tool DB is modelled in this research. The assembly tool DB includes hand tool and power tool DBs; the ERD of the hand tool DB is illustrated in Figure 4.5. As shown in Figure 4.5, the structure of the hand tool DB consists of 37 tables and relationships, except fastener-related tables. For each table, basic tool data including geometry information are obtained from a tool catalogue by Snap-on Company (2004). This geometry information of each basic tool is used to define the 15 tool parameters for a complete tool that is stored in the *HandTool* table. Thus, a selected tool retrieved from the *HandTool* table is directly used for a feasibility analysis based on the tool classification and tool parameters.

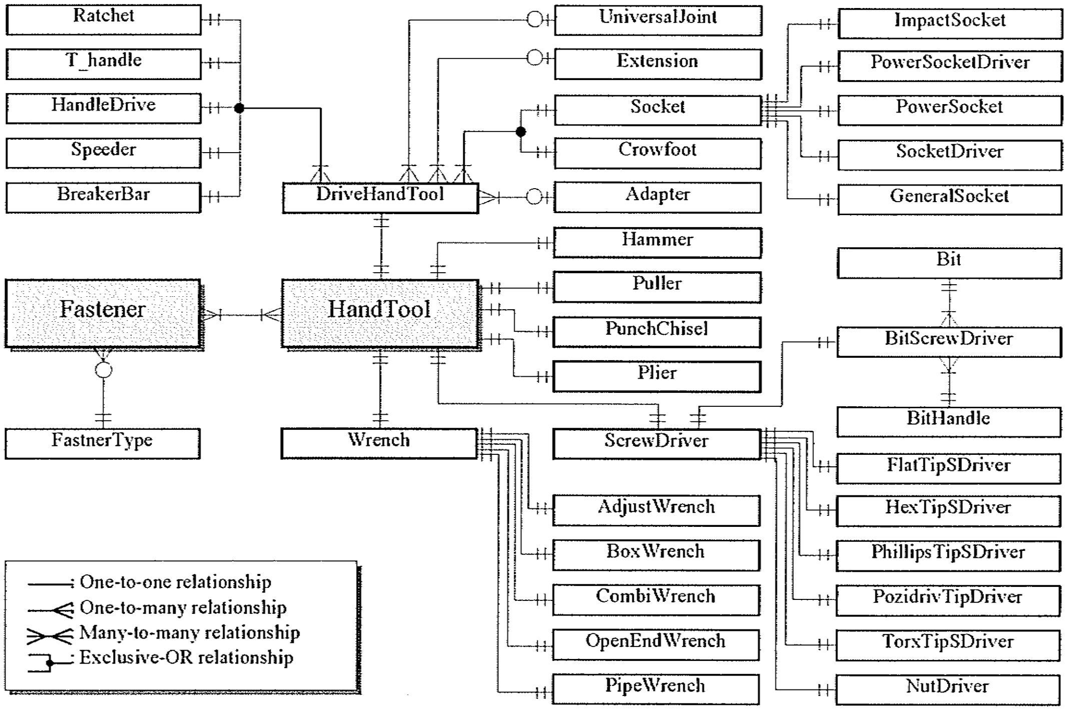


**Figure 4.5**: ERD of hand tool DB.

The *HandTool* table in Figure 4.5 has a many-to-many relationship (Watson, 1999) with the *Fastener* table so that various tool alternatives can be retrieved and analyzed for a given fastener. In particular, drive hand tools create many tool alternatives depending on types of handles used, and the use of extensions and universal joints. These tool alternatives generated by the combination of tool components dramatically increase when an adapter is used, which allows a tool handle (*i.e.*, 3/8″ drive) to use other extensions or sockets with different drive sizes (*i.e.*, 1/4″). With the support of the OOP technology combined with the assembly tool DB, the goal in this chapter is achieved efficiently by quickly examining various assembly-tool alternatives for a given fastener.

## 4.5 Feasibility Analysis of an Assembly Tool

The feasibility analysis of an assembly tool is a searching process with a constructed GAC$^d$. The four articulated devices of an assembly tool are separately analyzed for determining their feasibility against surrounding obstacles. Subsequently, a tool's feasibility for a fastener is guaranteed when all the four articulated devices are feasible via analysis methods being presented below.

### 4.5.1 Feasibility analysis of the effective handle

An effective handle has most movements among the articulated devices of a tool. For instance, it may rotate about $y$-axis with variations in the access angle $\alpha$ and the fastener-removal displacement $e_f$. In order to analyze its feasibility, a searching range based on these variations is defined along the $\varphi$ direction at a longitude angle $\theta$. The defined searching range at the angle $\theta$ is used for the interference check of the effective handle with a GAC$^d$ including depth information. Thus, this check is executed until the required minimum tool-application angle $\beta$ is found within the GAC$^d$. As shown in Figure 4.6, a searching range for an effective handle at a longitude angle $\theta$ is defined by four angles $\varphi_0$, $\varphi_1$, $\sigma_0$ and $\sigma_1$ along the $\varphi$ direction. The four angles are defined via four vertices of the effective handle; these are simply determined as follows.

$$\sigma_0 = \tan^{-1}\left(\frac{e_h' - \mathrm{A}}{e_a + e_e + e_b + e_f + \mathrm{B}}\right) \tag{4.4}$$

$$\sigma_1 = \tan^{-1}\left(\frac{e_h' - \mathrm{B}}{e_a + e_e + e_b + e_f - \mathrm{A}}\right) \tag{4.5}$$

$$\varphi_0 = \cos^{-1}(\mathrm{Y}/\mathrm{L}) - \Delta\varphi/2 \text{ and } \varphi_1 = \varphi_0 + \Delta\varphi \tag{4.6}$$

Where, $e_h' = \sqrt{e_h^2 + e_n^2}$ , $\mathrm{A} = \frac{h}{\sqrt{2}}\cos\left(\alpha - \frac{\pi}{4}\right)$ , $\mathrm{B} = \frac{h}{\sqrt{2}}\sin\left(\alpha - \frac{\pi}{4}\right)$ , $\mathrm{L} = \sqrt{\mathrm{X}^2 + \mathrm{Y}^2 + e_n{}^2}$

$\mathrm{X} = l_e \sin\alpha + e_h$; $\mathrm{Y} = l_e \cos\alpha + e_a + e_e + e_b + e_f$, and $\Delta\varphi = \tan^{-1}\left\{\frac{h}{\mathrm{L}\cos\left(\alpha - \cos^{-1}(\mathrm{Y}/\mathrm{L})\right)}\right\}$.

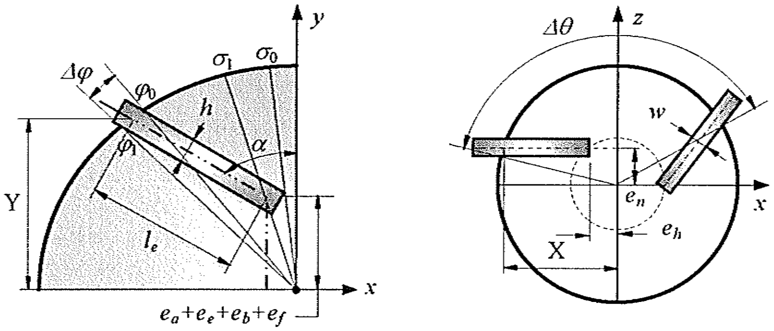


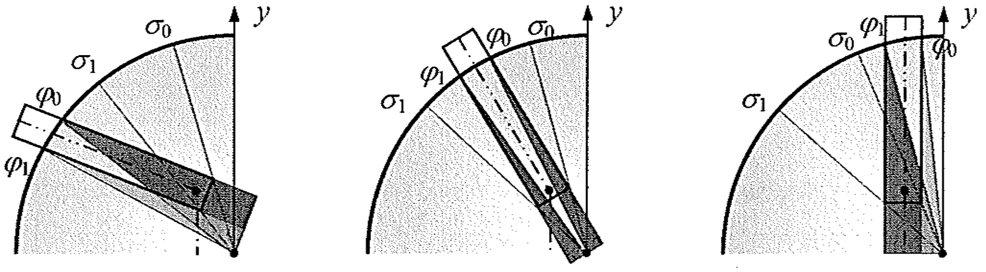
**Figure 4.6**: Determination of a searching range along the $\varphi$ direction for the feasibility analysis of an effective handle.

In addition, the required tool-application angle $\beta$ is converted as $\Delta\theta$ shown in Figure 4.6, which includes the angle generated by the width of an effective handle. Thus, the converted tool-application angle $\Delta\theta$ is calculated as follows.

$$\Delta\theta \cong \beta + \tan^{-1}(w/X) \tag{4.7}$$

In Equation (4.7), particularly, $\Delta\theta$ does not include $\beta$ for a TAT, according to the tool classification defined in this research. With the searching range defined by four colatitude angles $\varphi_0$, $\varphi_1$, $\sigma_0$ and $\sigma_1$, and the converted tool-application angle $\Delta\theta$, a searching process is executed with a GAC$^d$.

A criterion to determine the feasibility of the effective handle can be different depending on angle configurations that are formed by four angles $\varphi_0$, $\varphi_1$, $\sigma_0$ and $\sigma_1$. Figure 4.7 shows three different angle configurations to select a feasibility criterion, which are examined in this research.



(a) (b) (c)

**Figure 4.7**: Angle configurations formed by four angles (a) $\varphi_0<\sigma_0$ and $\varphi_1<\sigma_1$ (b) $\varphi_0<\sigma_0$ and $\varphi_1>\sigma_1$, and (c) $\varphi_0>\sigma_0$ and $\varphi_1>\sigma_1$.

The different feasibility criteria are applied to the three angle configurations shown in Figure 4.7. The following criteria derived in Figure 4.8 are respectively used to examine the feasibility for the cases (a), (b) and (c) in Figure 4.7.

$$\Lambda = \begin{cases} 1 & if \left(r^{\varphi_0}(\varphi) \le R(\theta,\varphi)\right) \vee \left(r^{\varphi_1}(\varphi) \ge R(\theta,\varphi)\right) \\ 0 & otherwise \qquad (\sigma_0 \le \varphi \le \varphi_1) \end{cases} \tag{4.8}$$

$$\Lambda = \begin{cases} 1 & if \left(\forall r^{\varphi_0}(\varphi') \le R(\theta,\varphi')\right) \wedge \left(\forall r^{\varphi_1}(\varphi'') \le R(\theta,\varphi'')\right) \\ 0 & otherwise \qquad (\sigma_0 \le \varphi' \le \varphi_0 \text{ and } \varphi_1 \le \varphi'' \le \sigma_1) \end{cases} \tag{4.9}$$

$$\Lambda = \begin{cases} 1 & if \left(r^{\varphi_0}(\varphi) \ge R(\theta,\varphi)\right) \vee \left(r^{\varphi_1}(\varphi) \le R(\theta,\varphi)\right) \\ 0 & otherwise \qquad (\varphi_0 \le \varphi \le \sigma_1) \end{cases} \tag{4.10}$$

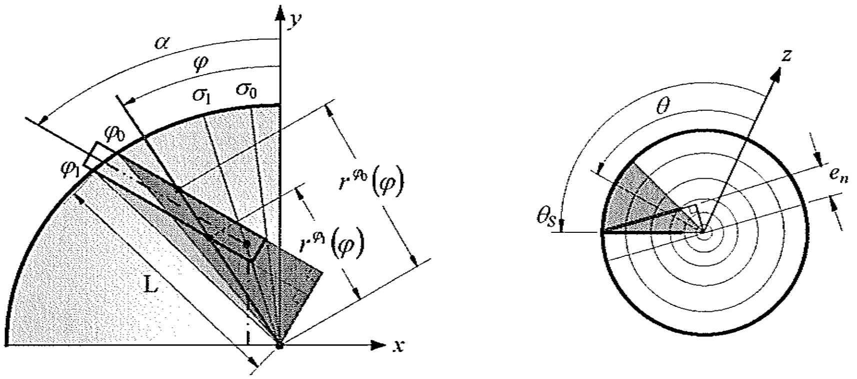


**Figure 4.8**: A detailed angle configuration of the effective handle.

For Equations (4.8), (4.9) and (4.10), a distance $r^{\varphi_0}(\varphi)$ or $r^{\varphi_1}(\varphi)$ can be determined in Figure 4.8, and expressed as follows.

$$r^{\varphi^*}(\varphi) = \frac{\mathrm{L}\sin(\alpha - \varphi^*)}{\sin(\alpha - \varphi)} \tag{4.11}$$

Where, $\varphi^*$ is either $\varphi_0$ or $\varphi_1$. As shown in Figure 4.8, in particular, a longitude angle $\theta$ used in Equations (4.8), (4.9) and (4.10) is not constant at an angle $\varphi$ although the end of an effective handle is positioned at the longitude angle $\theta_S$ that is the starting position to search a required feasible range. It results from the parameter $e_n$ that has the effective handle misaligned with the angle $\theta_S$. Therefore, Equations (4.8), (4.9) and (4.10) need a conversion to the actual $\theta$ according to a colatitude angle $\varphi$, and it is determined as follows.

$$\theta(\varphi) = \theta_S + \tan^{-1}\left(\frac{\mathrm{L}\sin\alpha\sin\left(\alpha - \cos^{-1}(\mathrm{Y}/\mathrm{L})\right)}{e_n \sin(\alpha - \varphi)}\right) - \tan^{-1}\left(\frac{\mathrm{X}}{e_n}\right) \tag{4.12}$$

## 4.5.2 Feasibility analysis of the tool head and extension

For the feasibility analysis of a tool head, we consider the *xz*-plane in which an interference check is executed between the tool head and a GAC$^d$. As shown in Figure 4.9, a searching range along the $\varphi$ direction is first determined as follows.

$$\varphi' = \tan^{-1}\left(\frac{r_a + e_h}{e_a + e_b + e_e + e_f}\right) \text{and}\ \varphi'' = \tan^{-1}\left(\frac{r_a + e_h}{e_b + e_e + e_f}\right) \tag{4.13}$$

A tool head in Figure 4.9 makes an eccentric circle if at least one of the two parameters $e_h$ and $e_n$ is not equal to zero (Refer to Figure 4.3). When an assembly tool moves, the eccentric circle at a longitude angle $\theta_S$ also moves or rotates about the *y*-axis. At every longitude angle $\theta$, therefore, the length $r_h(\lambda, \psi)$ for all the directions around the eccentric circle has to be calculated and checked with a GAC$^d$. At every longitude angle $\theta$, this checking process is repeatedly executed for a searching range defined by Equation (4.13).

Subsequently, this process is executed until the converted minimum tool-application angle $\Delta\theta$ is completely searched within a constructed GAC$^d$.

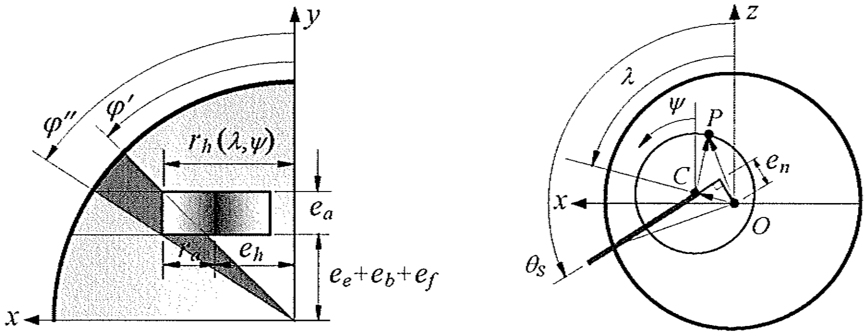


**Figure 4.9**: Determination of a searching range along the $\varphi$ direction for the feasibility analysis of the tool head.

When an assembly tool is at a longitude angle $\theta_S$, the length $r_h(\lambda, \psi)$ shown in Figure 4.9 is calculated as follows.

$$r_h(\lambda,\psi)=\sqrt{(e_h'\sin\lambda+r_a\sin\psi)^2+(e_h'\cos\lambda+r_a\cos\psi)^2} \tag{4.14}$$

Where, $\lambda=\theta_S-\tan^{-1}(X/e_n)+\tan^{-1}(e_h/e_n)$ and $0\leq\psi<2\pi$. Thus, the following Equation (4.15) expresses a criterion to check whether interference exists between a tool head and obstacles via a GAC$^d$. In Equation (4.15), a depth towards the direction defined at a pixel of a GAC$^d$ is projected on the *xz*-plane, and compared with a $r_h(\lambda,\psi)$ defined by Equation (4.14).

$$\Lambda=\begin{cases}1 & if\left\|R(\theta,\varphi)\hat{T}(\theta,\varphi)\cdot\left(\hat{i}+\hat{k}\right)\right\|-r_h(\lambda,\psi)>0\\ 0 & otherwise\quad\left(0\leq\psi<2\pi\text{ and }\varphi'\leq\varphi\leq\varphi''\right)\end{cases} \tag{4.15}$$

Where, $\theta = \cos^{-1}\left[(e_h' \cos\lambda + r_a \cos\psi)/r_h(\lambda,\psi)\right]$.

Similarly, the same method is used for the feasibility analysis of a tool extension in a different searching range. The radius of tool head $r_a$ in Equations (4.14) and (4.15) is only replaced by the radius of tool extension $r_e$. The searching range for the feasibility analysis of a tool extension can be modified from Equation (4.13), and expressed as follows.

$$\varphi' = \tan^{-1}\left(\frac{r_e + e_h}{e_b + e_f + e_e}\right) \text{and } \varphi'' = \tan^{-1}\left(\frac{r_e + e_h}{e_b + e_f}\right) \tag{4.16}$$

### 4.5.3 Feasibility analysis of the tool end

Compared with other articulated devices, tool end motions only depend on the variation of the fastener position $e_f$. In this research, therefore, the feasibility analysis of the tool end is once executed with the total length of a fastener $l_f$. In addition, the searching space along both $\varphi$ and $\theta$ directions includes the half area of a GAC$^d$. As shown in Figure 4.10, a colatitude angle $\varphi'$is first determined as follows.

$$\varphi' = \tan^{-1}\{r_b/(e_b + l_f)\} \tag{4.17}$$

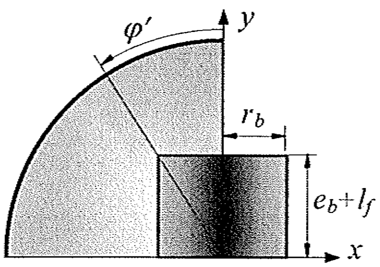


**Figure 4.10**: Determination of a searching range along the $\varphi$ direction for the feasibility analysis of the tool end.

This angle is used to divide the searching space in checking interference of the tool end with obstacles. Based on the determined angle $\varphi'$, the following criterion is used for the interference check with a GAC$^d$.

$$\Lambda = \begin{cases} if\ \varphi \le \varphi' & \begin{cases} 1 & if \left\| R(\theta,\varphi)\hat{T}(\theta,\varphi)\cdot\hat{j} \right\| - \left(e_b + l_f\right) > 0 \\ 0 & otherwise \end{cases} \\ else & \begin{cases} 1 & if \left\| R(\theta,\varphi)\hat{T}(\theta,\varphi)\cdot\left(\hat{i}+\hat{k}\right) \right\| - r_b > 0 \\ 0 & otherwise \end{cases} \end{cases} \tag{4.18}$$
$$\left(0 \le \varphi \le \pi/2\ and\ 0 \le \theta < 2\pi\right)$$

### 4.5.4 Overall procedure of tool feasibility analysis

Figure 4.11 shows a flow chart describing the overall procedure of analyzing the tool feasibility of an assembly tool. With variations in the tool access angle $\alpha$ and the fastener removal displacement $e_f$, a searching process is repeatedly executed on a constructed GAC$^d$. The process includes the four individual feasibility analyses for the articulated devices of an assembly tool, and continues until a fastener is completely disassembled (*i.e.*, $e_f \ge l_f$ in Figure 4.11) without any collision with surroundings. If there is no tool-application space at a fastener position, tool access angle $\alpha$ increases $\left(\alpha_{\min} \le \alpha \le \alpha_{\max}\right)$ to define a different searching space. Then, the searching process is executed based on the newly defined searching space in a GAC$^d$. The fastener position only increases (*i.e.*, $e_f$=$e_f$ +1 in Figure 4.11) for the next searching process if a feasible tool-application space is found by satisfying $\Delta\theta$ at the current fastener position $e_f$.

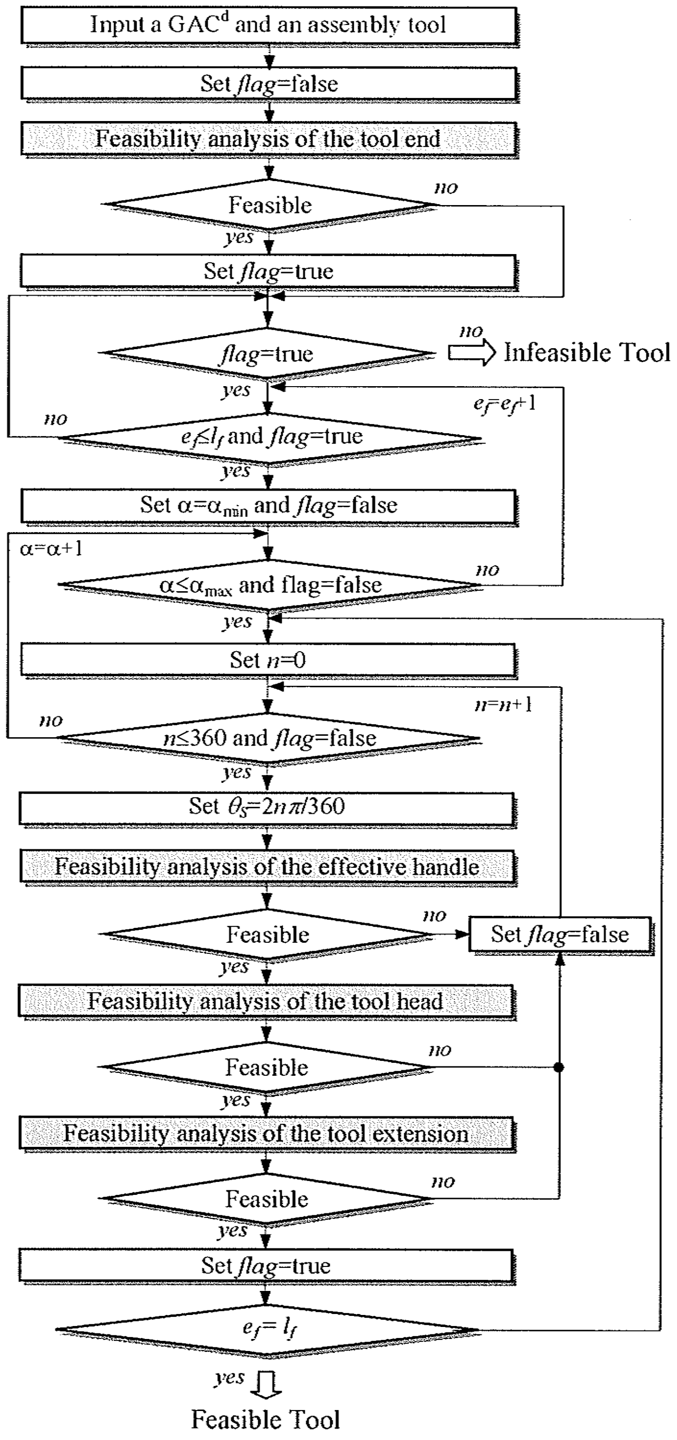

Input a GAC$^d$ and an assembly tool
Set *flag*=false
Feasibility analysis of the tool end
Feasible
no
yes
Set *flag*=true
*flag*=true
no
Infeasible Tool
yes
$e_f$=$e_f$+1
no
$e_f$≤$l_f$ and *flag*=true
yes
Set α=α$_{min}$ and *flag*=false
α=α+1
no
α≤α$_{max}$ and flag=false
yes
Set *n*=0
*n*=*n*+1
no
*n*≤360 and *flag*=false
yes
Set $\theta_S$=2*n*π/360
Feasibility analysis of the effective handle
Feasible
no
Set *flag*=false
yes
Feasibility analysis of the tool head
Feasible
no
yes
Feasibility analysis of the tool extension
Feasible
no
yes
Set *flag*=true
$e_f$= $l_f$
yes
Feasible Tool

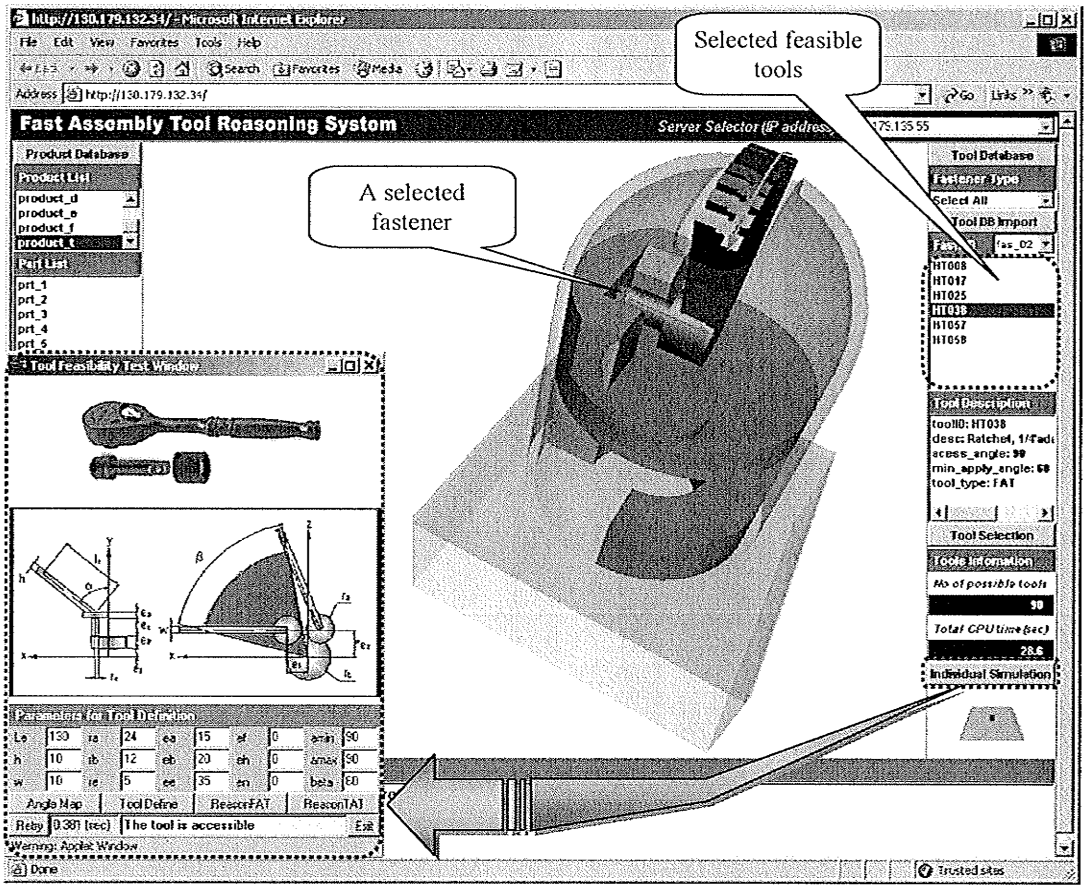
**Figure 4.11**: Overall procedure of the tool feasibility analysis.

## 4.6 Sample Applications

Several examples are tested to evaluate the efficiency of the approach developed in this chapter. Moreover, a system is implemented to test the feasibility of assembly tools in a dynamic manufacturing environment. The system is based on Web and DB technologies, and will be further extended to other systems for dynamic assembly and disassembly planning. The architecture and implementation methods of the system are later discussed in Chapter 8. In particular, central processing unit (CPU) time is used as an efficiency measure for this approach because a geometric tool-reasoning process is computationally expensive. The CPU time is measured on a Pentium 4 1.7 GHz personal computer (PC) with 512M read access memory (RAM). It only includes the time for the tool feasibility test without the time to construct a $GAC^d$, which takes a few seconds.

To analyse tool feasibility, a fastener is first selected for the product shown in Figure 4.12. The developed system allows a user to choose parts to construct the $GAC^d$ for a selected fastener so that various part configurations can be simply created from a product. Once a fastener is chosen by a user, the system retrieves all possible tool alternatives from the assembly tool DB, which are available for the selected fastener. Subsequently, the system tests all the retrieved possible tools to find feasible assembly tools based on the current part configuration for a selected fastener, which is here represented as a $GAC^d$.

As shown in Figure 4.12, six feasible tools are chosen from 90 retrieved possible tools for a selected fastener that is a hex-head bolt with diameter=3/8″ and length=1″. The fastener attached at the side block is shown in Figure 4.12. Moreover, these feasibility analyses for all possible tools took about 28.6 CPU seconds.



Server Selector (IP address): 179.135.55

Product List: product_d, product_e, product_f, product_t

Part List: prt_1, prt_2, prt_3, prt_4, prt_5

Fastener Type: Select All

fas_02

HT008
HT017
HT025
HT038
HT057
HT058

toolID: HT038
desc: Ratchet, 1/4"adt
acess_angle: 90
min_apply_angle: 60
tool_type: FAT

No of possible tools: 90

Total CPU time (sec): 28.6

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Lo | 130 | ra | 24 | ea | 15 | ef | 0 | amin | 90 |
| h | 10 | rb | 12 | eb | 20 | eh | 0 | amax | 90 |
| w | 10 | re | 5 | ee | 35 | en | 0 | beta | 60 |

0.381 (sec) The tool is accessible

Warning: Applet Window

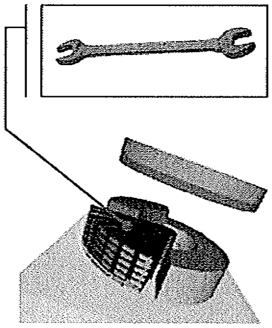
**Figure 4.12**: Feasibility analysis for a selected fastener (a hex-head bolt, dia.=3/8" and length=1") via the developed system.

In particular, this system allows a user to individually simulate a selected feasible tool; a tool consisting of a ratchet with an adapter and a socket is simulated via the tool feasibility analysis window shown in Figure 4.12. The window also shows the measured CPU time and the result-whether the selected tool is feasible or not. To dynamically generate tool loci representing a tool application about the fastener axis, such information as $\theta_S$, $\Delta\theta$, α and $\varphi$ based on a fastener position $e_f$ is stored in the system during the tool feasibility analysis. This information and tool parameters of a tested tool are used to generate dynamic tool loci when all the feasibility analyses of the tool are successfully completed. It helps a user examine tool interference with obstacles during the simulation of a tool application.

The following figures show other results of the feasibility analyses for various assembly tools. Each figure includes parameters used to define an assembly tool and the measured CPU time. Figure 4.13(a), (b) and (c) show the results of the feasibility analyses for several FATs. Since a rigid tool also has a little variation in the access angle $\alpha$, several experiments have a small margin in the access angle, which changes constant tool loci. In addition, the parameter $e_n$ is applied in Figure 4.13(c). With the parameter $e_h$, the parameter $e_n$ is useful to define eccentric tools such as a crowfoot wrench shown in Figure 4.13(c). Based on tests shown in Figure 4.13, the measured CPU times for FATs range from 0.261 to 0.691 sec.

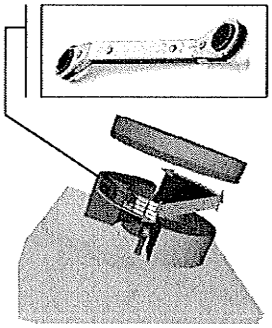
Figure 4.14(a) and (b) show results of the feasibility analyses for TATs. Their CPU times are measured in 0.362 and 0.531 sec, respectively. In particular, a speeder with a universal joint was tested in Figure 4.14(b). The range of a tool access angle is assumed as $\alpha$=30~60° for this test. As a result, this test searched the exact side hole of a part. This side hole allows the tested tool to be applied with variable access angles in disassembling the selected fastener. This test was executed in 0.531 sec, which repeatedly searched a GAC$^d$ because of the large variation in access angles.

| Parameter (mm) | | | |
|---|---|---|---|
| $l_e$ | 120 | $e_e$ | 0 |
| $h$ | 5 | $e_f$ | 0 |
| $w$ | 15 | $e_h$ | 0 |
| $r_a$ | 15 | $e_n$ | 0 |
| $r_b$ | 15 | $\alpha_{\min}$ | 85° |
| $r_e$ | 0 | $\alpha_{\max}$ | 90° |
| $e_a$ | 0 | $\beta$ | 60° |
| $e_b$ | 6 | FAT | |
| Feasible (0.321 sec) | | | |



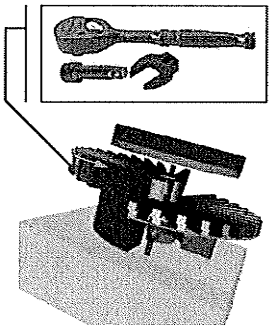
(a)

| Parameter (mm) | | | |
|---|---|---|---|
| $l_e$ | 100 | $e_e$ | 0 |
| $h$ | 5 | $e_f$ | 0 |
| $w$ | 15 | $e_h$ | 0 |
| $r_a$ | 10 | $e_n$ | 0 |
| $r_b$ | 10 | $\alpha_{\min}$ | 65° |
| $r_e$ | 0 | $\alpha_{\max}$ | 65° |
| $e_a$ | 0 | $\beta$ | 60° |
| $e_b$ | 5 | FAT | |
| Feasible (0.261 sec) | | | |



(b)

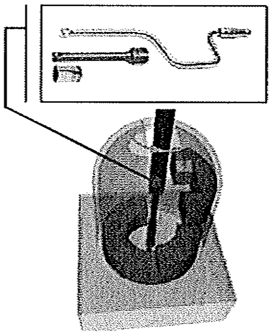
| Parameter (mm) | | | |
|---|---|---|---|
| $l_e$ | 130 | $e_e$ | 35 |
| $h$ | 10 | $e_f$ | 0 |
| $w$ | 10 | $e_h$ | 0 |
| $r_a$ | 24 | $e_n$ | 13 |
| $r_b$ | 15 | $\alpha_{\min}$ | 85° |
| $r_e$ | 5 | $\alpha_{\max}$ | 90° |
| $e_a$ | 15 | $\beta$ | 150° |
| $e_b$ | 8 | FAT | |
| Feasible (0.691 sec) | | | |



(c)

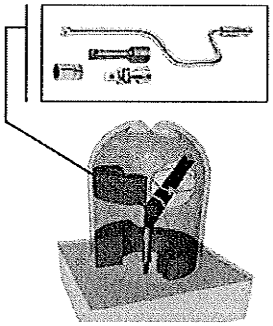
**Figure 4.13**: Experiment results of the feasibility tests for FATs: (a) an open-end wrench (α=90°) (b) a ratchet box wrench ($\alpha$=65°), and (c) a crowfoot wrench with an extension ($\alpha$=90° and $e_n$=13 mm).

| Parameter (mm) | | | |
|---|---|---|---|
| $l_e$ | 177 | $e_e$ | 127 |
| $h$ | 12 | $e_f$ | 0 |
| $w$ | 12 | $e_h$ | 0 |
| $r_a$ | 7 | $e_n$ | 0 |
| $r_b$ | 8 | $\alpha_{\min}$ | 0° |
| $r_e$ | 6 | $\alpha_{\max}$ | 0° |
| $e_a$ | 15 | $\beta$ | 360° |
| $e_b$ | 38 | TAT | |
| Feasible (0. 362 sec) | | | |



(a)

| Parameter (mm) | | | |
|---|---|---|---|
| $l_e$ | 177 | $e_e$ | 40 |
| $h$ | 12 | $e_f$ | 0 |
| $w$ | 12 | $e_h$ | 0 |
| $r_a$ | 7 | $e_n$ | 0 |
| $r_b$ | 8 | $\alpha_{\min}$ | 30° |
| $r_e$ | 6 | $\alpha_{\max}$ | 60° |
| $e_a$ | 15 | $\beta$ | 360° |
| $e_b$ | 38 | TAT | |
| Feasible (0. 531 sec) | | | |



(b)

**Figure 4.14**: Experiment results of the feasibility tests for TATs: (a) a speeder with an extension and a deep socket ($\alpha$=0°), and (b) a speeder with a universal joint, an extension and a deep socket ($\alpha$=30~60°).

## 4.7 Chapter Summary

This chapter presents a novel approach to geometric accessibility analysis for fast assembly tool reasoning. The developed tool reasoning method will be further applied in dynamic assembly and disassembly planning presented in Chapters 6 and 7, respectively.

A GAC$^d$ is defined based on a discrete approximation of surroundings of a fastener, which includes both a set of accessible directions and depth information. This subsequently forms a discrete non-homogeneous sphere. The depth from the centre point is determined for each non-accessible direction on a GAC$^d$ by projecting a unit vector onto a face of a surrounding part or obstacle.

An assembly tool is modelled as four articulated devices. The 15 parameters are extracted to define an assembly tool based on the tool representation proposed in this chapter. Moreover, two types of the tools, FATs and TATs based on the tool motions, are classified to efficiently support a fast feasibility analysis of an assembly tool. The tool parameters and classifications for general assembly tools are stored in the assembly-tool DB designed in this research.

The feasibility of an assembly tool is analyzed using the defined GAC$^d$. The analysis is separately performed for each articulated device of a tool. The developed algorithms use a simple algebra in the defined searching range, which can quickly determine the geometric feasibility of an assembly tool. Moreover, the algorithms can simply deal with complex variations in fastener movement and tool access angle.

With respect to the research on the tool feasibility analysis, published papers in international journals and conferences are as follows.

**Chung, C.** and Peng, Q. (2005). A novel approach to geometric feasibility analysis for fast assembly tool reasoning. International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-005-0173-z (First Online).

**Chung, C.** and Peng, Q. (2005). Fast assembly tool reasoning based on geometric accessibility analysis. ASME 2005 Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Long Beach, California, USA, September 24-28.

# Chapter 5

# Topological Disassemblability of Parts

This chapter presents an approach to topological disassemblability analysis of parts. Two primitive matrices are introduced for supporting the part disassemblability analysis. These matrices represent topological constraints of parts. A systematic method is also proposed to automatically construct these matrices from CAD solid models. The methods are based on various collision detection techniques developed in this chapter.

## 5.1 Proposed Approach

Disassemblability of a product is a function of several parameters such as exertion of manual force for disassembly, degree of precision required for effective tool placement, weight, size, material and shape of components being disassembled, and use of hand tools (Desai and Mital, 2003). Topological disassemblability is based on the geometric constraints that the shape of parts creates in a product. In assembly or non-destructive disassembly planning, the topological disassemblability determines the priority of parts being assembled or disassembled. With the assembly-tool feasibility presented in Chapter

4, this constraint forms a means to generate a practical and realistic plan for assembly or disassembly.

This chapter presents an approach to topological disassemblability analysis of parts. The part disassemblability analysis is executed based on two matrices: the fastener-part connectivity matrix (*FP*) and the part accessibility matrix (*PA*). A systematic method is also presented to construct these matrices from a CAD file. The part disassemblability analysis will be used in dynamic assembly and disassembly planning, which are presented in Chapters 6 and 7, respectively.

## 5.2 Two Primitive Matrices

The matrix ***FP*** that represents connectivity between fasteners and parts is defined in Equation (5.1). The row and column vectors represent the fastener and part sets, ***M*** and *N*, in a product, respectively. In Equation (5.1), $FP_{i,j}$ is 1 if fastener *i* is connected to part *j* otherwise, 0.

$$\boldsymbol{FP} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ i \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & j \end{array} \\ \begin{bmatrix} FP_{1,1} & FP_{1,2} & \cdots & FP_{1,j} \\ FP_{2,1} & FP_{2,2} & \cdots & FP_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ FP_{i,1} & FP_{i,2} & \cdots & FP_{i,j} \end{bmatrix} \end{array} \quad (i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n) \qquad (5.1)$$

Where, $FP_{i,j} = \begin{cases} 1 & \textit{if fastener i is connected to part j} \\ 0 & \textit{otherwise} \end{cases}$

Accessibility of part *i* to its adjacent part *j* is defined as the set of directions with which part *i* can be moved relative to part *j*, and is denoted as a directionality matrix, $\boldsymbol{PA}^{i,j}$ (Srinivasan *et al.*, 1999). Equation (5.2) shows the matrix *PA*.

$$PA = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ i \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & j \end{array} \\ \begin{bmatrix} PA^{1,1} & PA^{1,2} & \cdots & PA^{1,j} \\ PA^{2,1} & PA^{2,2} & \cdots & PA^{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ PA^{i,1} & PA^{i,2} & \cdots & PA^{i,j} \end{bmatrix} \end{array} \quad (i, j = 1, 2, \cdots, n) \qquad (5.2)$$

A directionality matrix $PA^{i,j}$ in Equation (5.2) consists of a set of directions. Corresponding to the matrix size of $PA^{i,j}$, the number of directions can increase by a user definition in a coordinate system. For instance, Figure 5.1(a) shows 6 directions, $\pm x$, $\pm y$ and $\pm z$, defined in the Cartesian coordinate system.

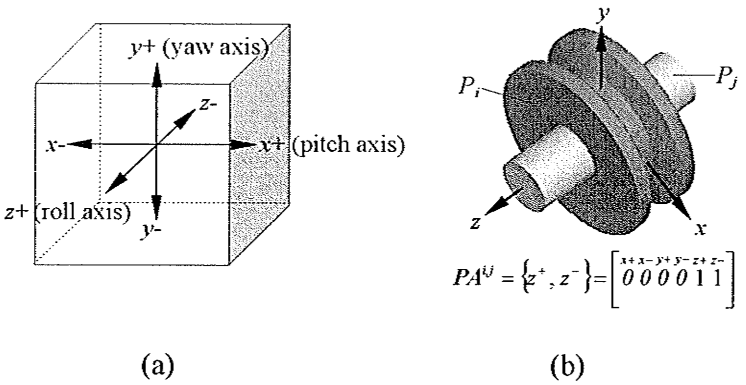

y+ (yaw axis)
z-
x-
x+ (pitch axis)
z+ (roll axis)
y-
y
$P_i$
$P_j$
z
x
$PA^{i,j} = \{z^+, z^-\} = \begin{bmatrix} x+ & x- & y+ & y- & z+ & z- \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$
(a)
(b)

**Figure 5.1**: An example of defining an element $PA^{i,j}$: (a) 6 directions in the Cartesian coordinate system, and (b) representation of the element $PA^{i,j}$.

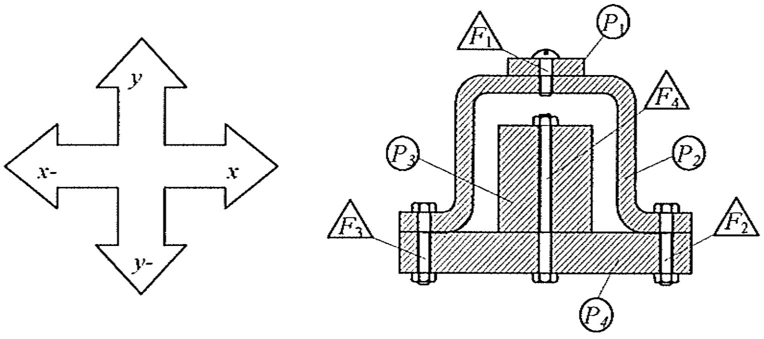
Part $i$ shown in Figure 5.1(b) can only move relative to part $j$ along the two directions, $z+$ and $z-$. Thus, the element $PA^{i,j}$ is represented as [*x+*:*x-*:*y+*:*y-*:*z+*:*z-*⇒*000011*], where *0* and *1* respectively represent an inaccessible direction, and an accessible direction. Similarly, if part $i$ equals to part $j$, the element $PA^{i,i}$ in the matrix $PA$ has the complete set of six-directions, $\pm x$, $\pm y$ and $\pm z$, and is represented as [*111111*].

For a part $P_i$, topological disassemblability ($\Delta i$) within the set $N$ is determined as the following Equation (5.3).

$$\Delta i = \begin{cases} 1 & if \bigcap_{j \in N} PA^{i,j} \neq \Phi, \text{subject to} \sum_{k} FP_{k,i} = 0, k \in M \\ 0 & otherwise \end{cases} \tag{5.3}$$

Equation (5.3) represents that a part satisfying these geometric constraints should not be connected with any fastener when it is disassembled. The fastener constraint in Equation (5.3) is generally satisfied by the two following cases: (1) there is no fastener connecting the part, and (2) all related fasteners are removed before disassembling the part.

Take a simple product as an example to show how it is represented by two primitive matrices, *FP* and *PA* defined in Equations (5.1) and (5.2), respectively. Figure 5.2(b) illustrates a section drawing of an example product with four parts and four fasteners. The four fasteners are three bolts-nuts and one screw. The applied four directions are also shown in Figure 5.2(a), namely $x+$, $x-$, $y+$ and $y-$.



(a) (b)

**Figure 5.2**: An example of defining two matrices, ***FP*** and ***PA***: (a) applied directions in the Cartesian coordinate system, and (b) a section drawing of the example product.

Using Equations (5.1) and (5.2), the primitive matrices of the example part are defined as follows.

$$FP = \begin{array}{c} \\ F_1 \\ F_2 \\ F_3 \\ F_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \text{ and } PA = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \end{array} \\ \begin{bmatrix} \{1111\} & \{1110\} & \{1110\} & \{1110\} \\ \{1101\} & \{1111\} & \{0010\} & \{1110\} \\ \{1101\} & \{0001\} & \{1111\} & \{1110\} \\ \{1101\} & \{1101\} & \{1101\} & \{1111\} \end{bmatrix} \end{array}$$

By Equation (5.3) with the matrix *PA* defined above, $\Delta_1 = \bigcap_{j \in N} PA^{1,j} = \{1110\} = \{x+, x-, y+\}$, $\Delta_2$, $\Delta_3 = \{0000\} = \Phi$, and $\Delta_4 = \{1101\} = \{x+, x-, y-\}$. Subsequently, only two parts $P_1$ and $P_4$ can be first disassembled after fastener sets $\{F_1\}$ and $\{F_2, F_3, F_4\}$ are respectively removed.

## 5.3 Construction of Primitive Matrices

A systematic method is presented to automatically construct the primitive matrices ***FP*** and *PA* from a CAD file represented in VRML. As mentioned in Chapter 4, a component in this research is referred to as a set of triangle patches, which is widely used for solid representations in computer graphics. Figure 5.3 shows the hierarchical structure of a product CAD model used in this research.

As shown in Figure 5.3, a product consists of a number of parts and fasteners. The product is normally generated from an assembly CAD model integrating part and fastener CAD models via a CAD system. Each component like a part or a fastener is represented as a number of triangle patches, each patch is defined by three vertices $V_0^i$, $V_1^i$ and $V_2^i$, and its normal vector $\boldsymbol{n}_i$.

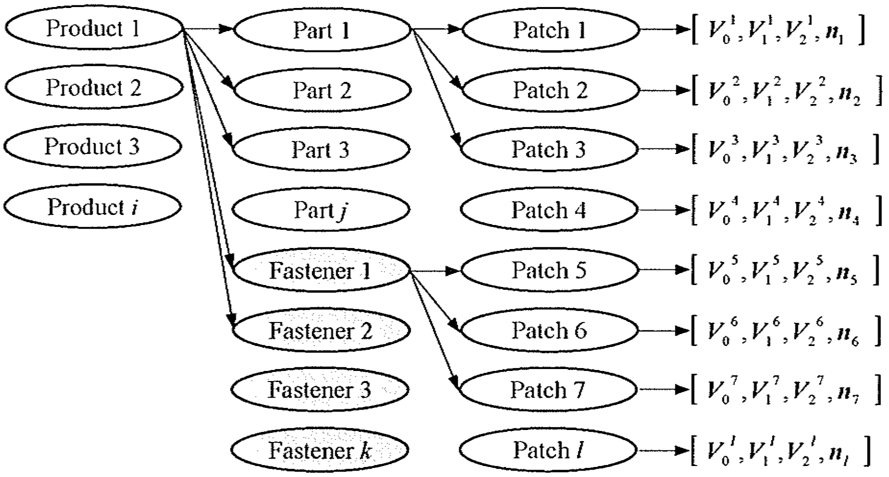

Product 1
Product 2
Product 3
Product *i*
Part 1
Part 2
Part 3
Part *j*
Fastener 1
Fastener 2
Fastener 3
Fastener *k*
Patch 1
Patch 2
Patch 3
Patch 4
Patch 5
Patch 6
Patch 7
Patch *l*
$[V_0^1, V_1^1, V_2^1, n_1]$
$[V_0^2, V_1^2, V_2^2, n_2]$
$[V_0^3, V_1^3, V_2^3, n_3]$
$[V_0^4, V_1^4, V_2^4, n_4]$
$[V_0^5, V_1^5, V_2^5, n_5]$
$[V_0^6, V_1^6, V_2^6, n_6]$
$[V_0^7, V_1^7, V_2^7, n_7]$
$[V_0^l, V_1^l, V_2^l, n_l]$

**Figure 5.3**: Hierarchical structure of a product CAD model used in this research.

For determining primitive matrices ***FP*** and ***PA***, several collision detection methods based on triangle patches are used in this research. A collision between components is quickly detected by the algorithm developed, which combines bounding sphere- (BS), oriented bounding box- (OBB), and triangle patch-based methods. The fundamentals used in the methods (Gottschalk *et al.*, 1996; Möller, 1997; Peng, 2002) are also summarized in Appendix A.13.

Algorithm 5.1 describes the procedure to determine fastener interference in constructing the matrix ***FP***. The main goal of this algorithm is to quickly find a triangle patch of a part, which might collide with a fastener. It starts with the BS-based test with two BSs, $BS_i$ and $BS_j$ approximating fastener $i$ (*i.e.*, $F_i$, $i \in M$) and part $j$ (*i.e.*, $P_j$, $j \in N$), respectively. If a collision is detected between $BS_i$ and $BS_j$, the $F_i$ and $P_j$ are respectively approximated into $OBB_i$ and $OBB^j$ for a more precise collision test. The $OBB^j$ is divided in two OBBs via the top-down approach in the hierarchical OBB-tree method (Gottschalk *et al.*, 1996) if a collision still exists between $OBB_i$ and $OBB^j$. Two OBBs, $OBB^j{}_1$ and $OBB^j{}_2$ are again applied to the OBB-based test to choose a new $OBB^j$ with collision. This division process is continuously executed while a collision with the $OBB_i$ occurs in either $OBB^j{}_1$ or $OBB^j{}_2$, and a chosen OOB with the collision consists of two or more triangle patches.

**Algorithm 5.1**: Determination of the matrix *FP*

```
Input part and fastener sets, $N$ and $M$
For $i = 1, \ldots, m \Rightarrow$
    Generate $BS_i$ based on fastener $F_i$
    For $j = 1, \ldots, n \Rightarrow$
        Let $FP_{i,j} = 0$
        Generate $BS_j$ based on part $P_j$
        Execute BS-based collision test $(BS_i, BS_j)$   /* Appendix A.13(1) */
        If there is a collision then
            Generate $OBB_i$ based on fastener $F_i$
            Generate $OBB^j$ based on part $P_j$
            Execute the OBB-based collision test $(OBB_i, OBB^j)$
                                                        /* Appendix A.13(3) */
            Do while (there is a collision) $\Rightarrow$
                If $OBB^j$ is defined by only a triangle patch $T_k^{\,j}$ then
                    Take a triangle patch $(T_i)$ from $OBB_i$
                    Execute triangle-to-triangle interference test $(T_i, T_k^{\,j})$
                                                        /* Appendix A.13(2) */
                    If there is a collision then $FP_{i,j} = 1$
                    Exit Do
                Else
                    Partitioning $OBB^j$ via the hierarchical OBB-tree method $\Rightarrow$
                    $OBB^j{}_1$ and $OBB^j{}_2$
                    Execute the OBB-based collision test $(OBB_i, OBB^j{}_1)$
                    Execute the OBB-based collision test $(OBB_i, OBB^j{}_2)$
                                                        /* Appendix A.13(3) */
                    Let $OBB^j = OBB^j{}_1$ or $OBB^j{}_2$ if there is a collision
```

As shown in Algorithm 5.1, subsequently, the triangle-triangle intersection test is carried out via one triangle patch of the $OBB_i$ and the interfered patch $k$ (*i.e.*, $T_k^j$ in Algorithm 5.1) of $P_j$. Moreover, a binary value (*i.e.*, *0* or *1*) for an element $FP_{i,j}$ in the matrix $\boldsymbol{FP}$ is determined by the result of this test. Figure 5.4 shows the overall procedure of determining fastener interference via Algorithm 5.1.

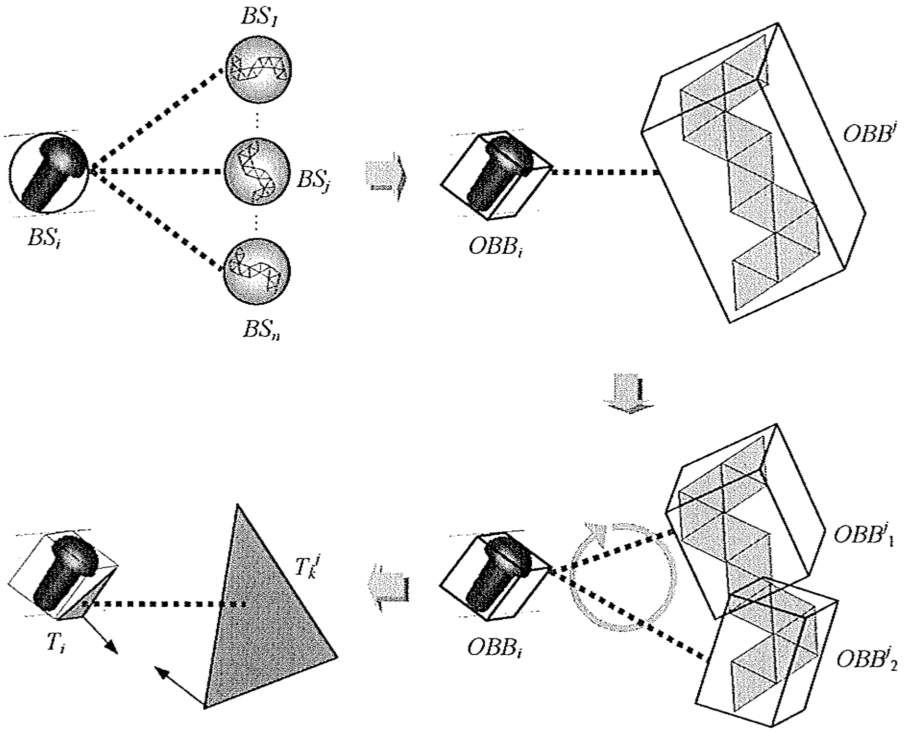
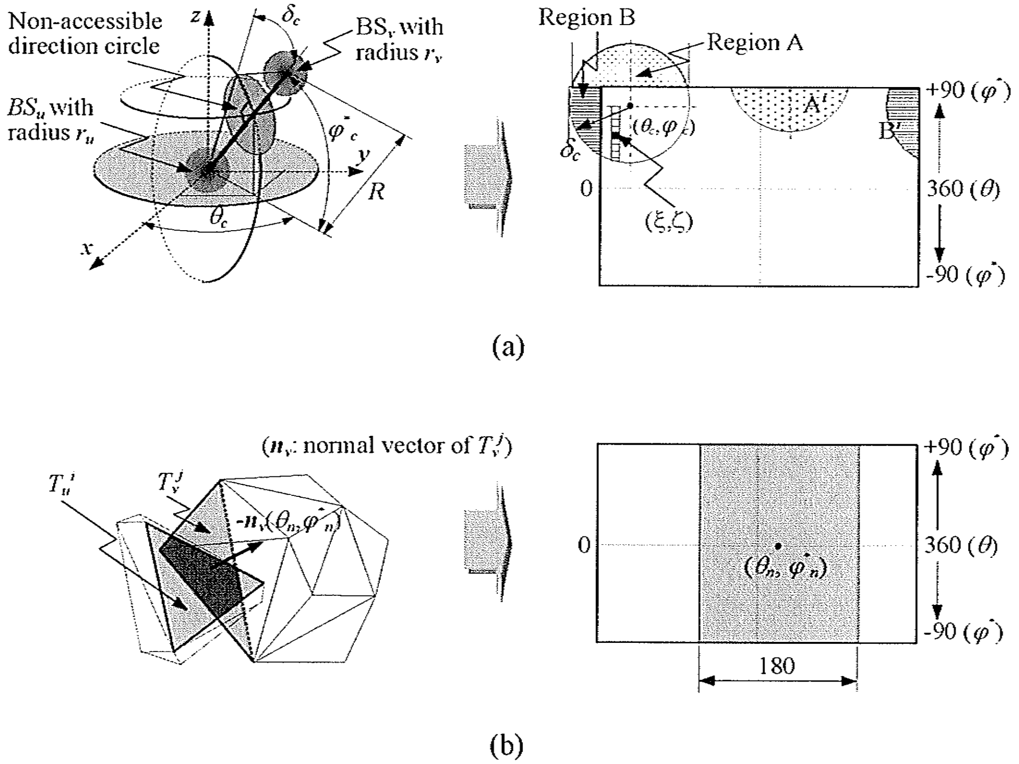


**Figure 5.4**: Overall procedure of determining fastener interference via Algorithm 5.1.

Determining $PA^{i,j}$ ($i \neq j$ and $i,j \in N$) in the matrix $PA$ is a computationally intensive process because all disassembly directions should be considered for all part combinations that have complex shapes with a large number of triangle patches. In this research, a digitized disassembly directionality chart (D$^3$C) is introduced. A D$^3$C here stores only direction information for determining part accessibility. Similar with the GAC$^d$ presented in Chapter 4, the D$^3$C consists of 180×360 pixels. Total 64,800 directions are defined by the pixels in a D$^3$C, each of which represents a different direction vector in the global Spherical coordinate. The pixels of a D$^3$C are simply represented as the Boolean data type, *true* (inaccessible) or *false* (accessible), which is used for searching accessible directions of a part.

To efficiently generate the D$^3$C for a $\boldsymbol{PA}^{i,j}$, two mapping processes are presented, which are only based on two BSs with no collision and two triangle patches with collision. The collision detection process presented in Algorithm 5.1 is here used to obtain these two types of objects. In particular, a BS is approximately obtained from not only a part with no collision, but also an OBB with no collision during the repeating process of the OBB division and collision detection.

During the process, therefore, one or more sets of BSs are generated for two parts, $P_i$ ($i \in N$) and $P_j$ ($j \in N$), and these sets are mapped into the D$^3$C for a $\boldsymbol{PA}^{i,j}$. If $P_i$ and $P_j$ are contacted, two triangle patches with collision are subsequently detected, and mapped into the D$^3$C for the $\boldsymbol{PA}^{i,j}$ via the triangle patch-based mapping process. This is the worst case, and computationally expensive because the collision detection tests are continuously executed from the top level to the bottom level of two parts. Both BS-based mapping processes and a triangle patch-based mapping process are subsequently executed for this case. Figure 5.5 shows two mapping processes: BS- and triangle patch-based mapping processes.



(a)

(b)

**Figure 5.5**: Determination of the D$^3$C for $PA^{i,j}$: (a) mapping to a D$^3$C with non-accessible direction circle between two BSs, and (b) mapping to a D$^3$C with two contacting patches.

In Figure 5.5(a), a non-accessible direction circle is defined if there is no collision between given two BSs, $BS_u$ and $BS_v$. As mentioned before, the $BS_u$ or $BS_v$ is approximately obtained from not only a part with no collision, but also an OBB with no collision during the repeating process of the OBB division and collision detection. The non-accessible direction circle in Figure 5.5(a) is a blocking region consisting of a set of directions between two BSs, and is represented by longitude ($\theta$), latitude ($\varphi^*$) and radius ($\delta$). If a vector between $BS_u$ and $BS_v$ in Figure 5.5(a) is represented by $x\hat{i} + y\hat{j} + z\hat{k}$, the non-accessible direction circle is defined by Equation (5.4).

$$\begin{cases} \varphi_C^* = \sin^{-1}(z / R) \\ \theta_C = \tan^{-1}(y / x) \\ \delta_C = \sin^{-1}\{(r_u + r_v)/ R\} \end{cases} \tag{5.4}$$

Where, $R = \left\lfloor x^2 + y^2 + z^2 \right\rfloor^{1/2}$. As shown in Figure 5.5(a), subsequently, a defined non-accessible direction circle by Equation (5.4) is mapped into the D$^3$C for a *PA*$^{i,j}$.

As shown in Figure 5.5(b), a part $P_i$ is in contact with another part $P_j$, and two touching triangle patches are $T_u^i$ and $T_v^j$. In this case, a non-accessible direction region of 180×180 degree is mapped into the D$^3$C shown in Figure 5.5(b). The centre point of this square region is the negative normal vector $-\boldsymbol{n}_v(\theta_n, \varphi^*_n)$ of $T_v^j$. Similar to the result from Gaussian sphere-based approaches (Mo *et al.*, 2002; Pomares *et al.*, 2004), the square region results in blocking half of the disassembly directionality space for the disassembly of part *i*.

In particular, to fit all the angles $\xi$ and $\zeta$ digitized from non-accessible direction shapes into a D$^3$C, this research uses several transformation equations, one of which is introduced as follows.

$$\mathit{If}\ |\zeta| > 90^\circ \begin{cases} \xi = [360 + (180 + \xi)] \,\mathbf{mod}\, 360 \\ \zeta = \zeta / |\zeta| (180 - |\zeta|) \end{cases} \tag{5.5}$$

This equation is applied for both mapping processes shown in Figures 5.5(a) and (b). By Equation (5.5), for instance, region *A* shown in Figure 5.5(a) is transformed to regions *A'*. Subsequently, all the transformed angles $\xi$ and $\zeta$ are fitted into the D$^3$C for a ***PA***$^{i,j}$. Moreover, this methodology enables to use a simple coordinate transformation of ***PA***$^{i,j}$

when $PA^{j,i}$ is already constructed so that it can make the system avoid the same calculation with parts $i$ and $j$.

A case study example of determining a $PA^{i,j}$ is illustrated in Figure 5.6. The product consists of 17 parts tightly connected with 20 fasteners including bolts, screws and nuts. As shown in Figure 5.6, part $P_1$ has topological disassemblability relative to part $P_6$ examined by the method developed in this chapter.

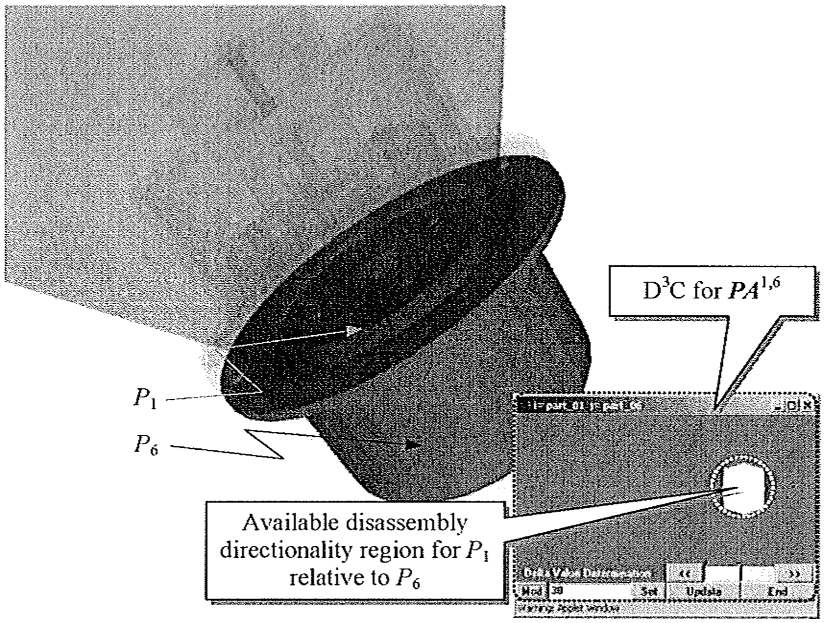
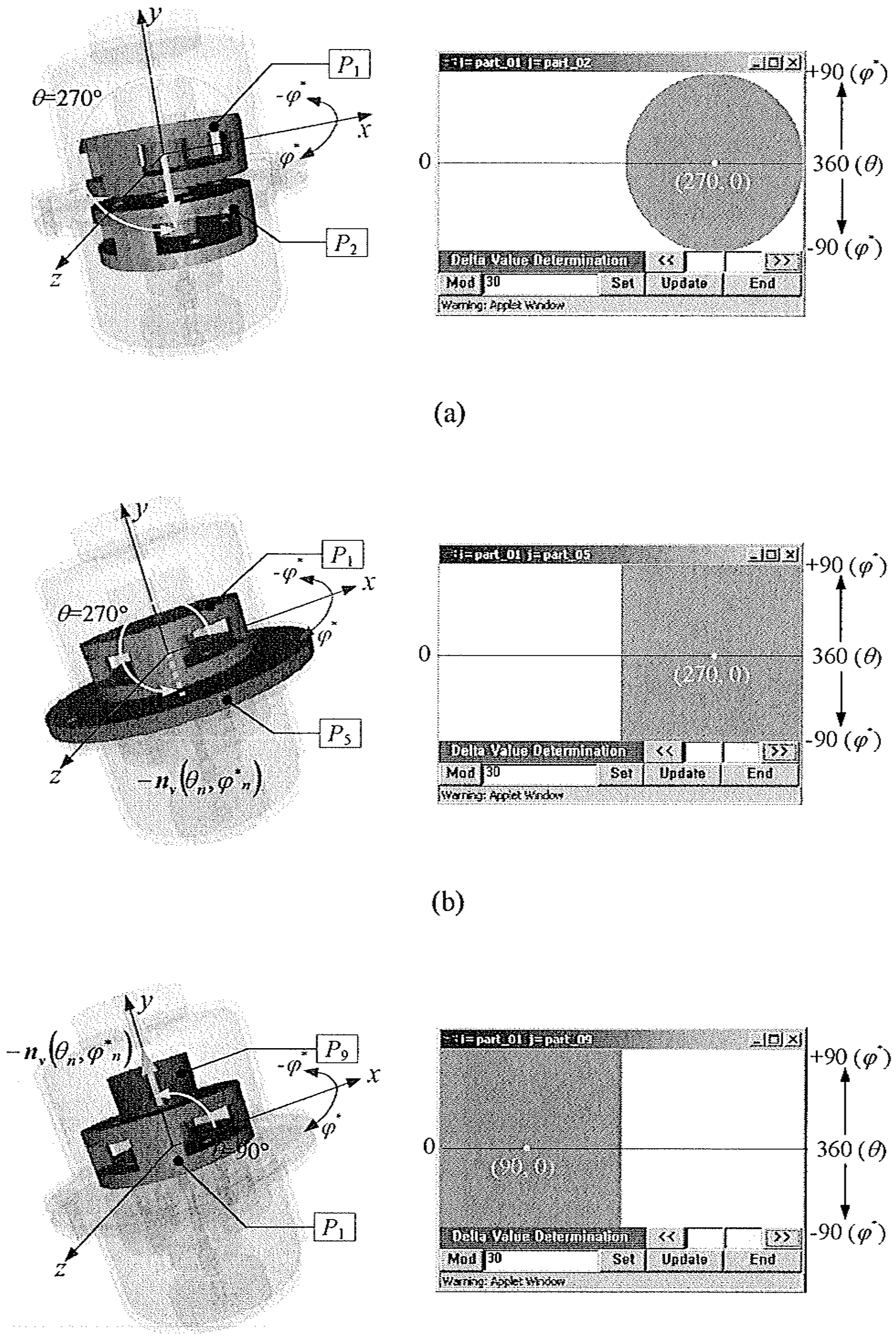

D$^3$C for $PA^{1,6}$
$P_1$
$P_6$
Available disassembly directionality region for $P_1$ relative to $P_6$
i= part_01 j= part_06
«
»
Mod
30
Set
Update
End

**Figure 5.6**: A test result of determining $PA^{i,j}$.

The white region of the D$^3$C in Figure 5.6 represents an available disassembly directionality region for $P_1$ while the rest of the regions represent non-accessible directions. In particular, the available disassembly directionality region is a set of accessible directions; each direction allows the assembling or disassembling of $P_1$

relative to $P_6$. This set of accessible directions is converted to the direction cone shown in Figure 5.6.

Similarly, various topological disassemblability tests are executed for several parts, and the results are illustrated in Figure 5.7. Grey coloured regions in Figure 5.7 are formed by non-accessible direction circles or regions, both of which are defined by the mapping processes shown in Figure 5.5. In particular, Figures 5.7(b) and (c) illustrate the topological disassemblability tests for two contacted parts. In Figure 5.7(b), for instance, a non-accessible direction region of 180×180 degree is mapped into the D$^3$C of the $\boldsymbol{PA}^{1,5}$ after automatically detecting two-contacted triangle patches of $P_1$ and $P_5$. The centre point of this square is defined by $-\boldsymbol{n}_v(\theta_n, \varphi^*{}_n)$, the (-) normal vector of the contacted patch of $P_5$. This is the same result of the Gaussian sphere-based approaches (Mo *et al.*, 2002; Pomares *et al.*, 2004), resulting in blocking half of the disassembly directionality space for the disassembly of $P_1$.



(a)

(b)

(c)

**Figure 5.7**: Test results of determining a $PA^{i,j}$ (a) $PA^{1,2}$ (b) $PA^{1,5}$ and (c) $PA^{1,9}$.

## 5.4 Chapter Summary

This chapter presented an approach to topological disassemblability analysis of parts in a product. Two primitive matrices were introduced in this chapter, which are $FP$ and $PA$. A constraint for ensuring part disassemblability was formulated based on these two primitive matrices. These matrices and the constraint will be further used for dynamic assembly and disassembly planning presented in Chapters 6 and 7, respectively.

A systematic method was developed to construct two primitive matrices from a CAD file. Several collision detection methods were used in this chapter, namely BS-, OBB-, and triangle patch-based collision detection methods. A digitized disassembly directionality chart, a $D^3C$, was introduced to represent a part accessibility $PA^{i,j}$. In determining complicated topological disassemblability, the $D^3C$ allows for mapping 3D geometric constraints between parts into a 2D directionality map. This is advantageous not only for executing operations including merging all $PA^{i,j}$ in the matrix $PA$, but also for searching available directions for a part assembly or disassembly.

With respect to the research on the part disassemblability analysis, published papers in international journals and conferences are as follows.

**Chung, C.** and Peng, Q. (2005). A hybrid approach to selective-disassembly sequence planning for de-manufacturing and its implementation on the Internet. *Special issue on intelligent disassembly in the de-manufacturing process*, International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-005-0038-5 (First Online).

**Chung, C.** and Peng, Q. (2005). An integrated approach to selective-disassembly sequence planning. International Journal of Robotics and Computer Integrated Manufacturing (RCIM) 21(4-5): 475-485.

# Chapter 6

# Dynamic Assembly Planning

This chapter presents a GA-based approach to dynamic assembly planning at the production level. The approach is based on the assembly-tool feasibility analysis and the part disassemblability analysis, presented in Chapters 4 and 5, respectively. Outcomes of this process are the best set of assembly tools and an optimized high-fidelity assembly plan satisfying both topological constraints of components and tool feasibility.

## 6.1 Proposed Approach

The proposed approach for dynamic assembly planning aims to generate a high fidelity assembly sequence and optimal assembly-tool alternatives in a dynamic manufacturing environment. The main idea of this approach is to embed assembly tools into the process for searching optimal assembly sequences. To achieve the objective of this chapter, this approach uses a GA-based assembly planning process with the assembly-tool feasibility analysis and the part disassemblability analysis presented in Chapters 4 and 5, respectively. The two analyses help to generate a high fidelity assembly sequence that ensures both assembly-tool feasibility and topological part disassemblability. In particular, the assembly-tool DB designed in Chapter 4 is used in order to store the tool information available in a dynamic manufacturing environment. This DB efficiently supports dynamic assembly planning with the proper tool selection, which is also called the tool selection-embedded optimal assembly planning in this chapter. What follows are the major contents included in this proposed approach.

***Construction of fastener-based assembly structure.*** An FBAS is constructed using the assembly-tool feasibility analysis and the part disassemblability analysis presented in Chapters 4 and 5, respectively. Fastener visibility is also considered in constructing an FBAS. The fastener visibility, which can be an option for reliable assembly, allows an assembler to observe the fastener being assembled by an assembly tool. A constructed $\mathrm{GAC}^d$ for conducting the assembly-tool feasibility analysis is used to analyze fastener visibility. In particular, an object-oriented fastener class with assembly-tool OODBs is used here efficiently to handle many retrieved tool alternatives from the assembly-tool DB during the process of constructing an FBAS.

***GA-based assembly planning.*** A two-tier GA is proposed for the tool selection-embedded optimal assembly planning. The two-tier GA is executed with a constructed FBAS. It consists of two GA-based optimizations: (1) an assembly level based GA for creating the initial population and (2) a product based GA for generating an optimal assembly sequence. It helps to optimize fastener-based assembly sequences with a reduced number of generations or iterations.

At the end of this chapter, an example is applied to perform the dynamic assembly planning and to simulate tool applications based on the created assembly-tool set. In a computer generated 3D environment, this simulation shows clearly that the assembly-tool set is appropriate for an optimized assembly sequence.

## 6.2 Construction of FBAS

The tool selection-embedded assembly planning starts with constructing an FBAS. Although the notion has been applied by many researchers (Tseng and Li, 1999; Yin *et al.*, 2003), this research proposes an FBAS based on the assembly-tool feasibility analysis and the part disassemblability analysis presented in Chapters 4 and 5, respectively. The construction of an FBAS is based on the reverse assembling sequences (*i.e.*, disassembly) of a complete product. Following this notion, the top level of an FBAS is first defined from a complete product. Figure 6.1 illustrates the concept of the FBAS used.

As shown in Figure 6.1, the FBAS of a product is constructed in assembly indenture levels. Each level consists of parts and fasteners, which ensure topological part disassemblability and assembly-tool feasibility, respectively. In particular, it is assumed that there is no fastener precedence within the same assembly indenture level. At each level, parts are first positioned, and then the fasteners connect them by means of assembly tools. To complete the assembly of a product, all the parts and fasteners are assembled in the order of indenture levels starting from the level 1.

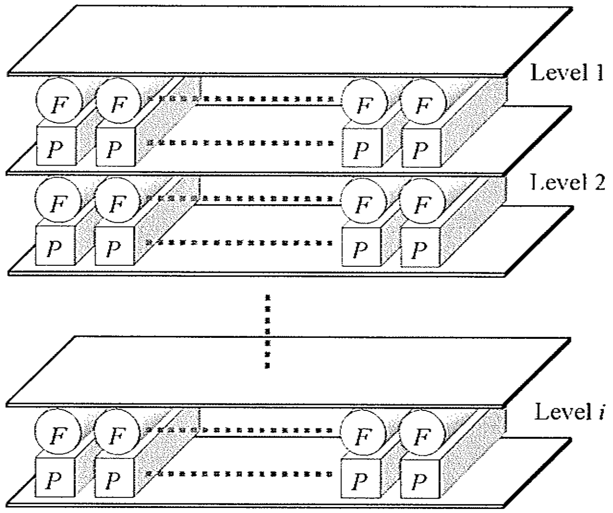


**Figure 6.1**: Notion of the FBAS used in this chapter.

Based on the notion above, Figure 6.2 shows the overall procedure to construct an FBAS. The system developed in this research retrieves a product from the product DB. The product consists of fasteners and parts, represented by a set of triangle patches that were discussed in Chapter 5. These patches are used not only to construct the GAC$^d$ presented in Chapter 4, but also to determine two primitive matrices presented in Chapter 5. In particular, all the specifications of the stored fasteners are pre-defined in the fastener DB so that the system easily retrieves available assembly tools associated with the fasteners.

To execute the assembly-tool feasibility analysis, as shown in Figure 6.2, the developed system retrieves available tools from the assembly tool DB. All of the available assembly tools are kept for each fastener in a product without considering any geometric constraint that is determined by a certain part configuration. At every assembly level, the developed system computes the assembly-tool feasibility for all tools retrieved for each fastener. After computing for all the fasteners in a product, the system determines accessible fasteners with at least one feasible tool. The accessible fasteners are used to determine potentially removable parts via the defined matrix *FP* in Figure 6.2.

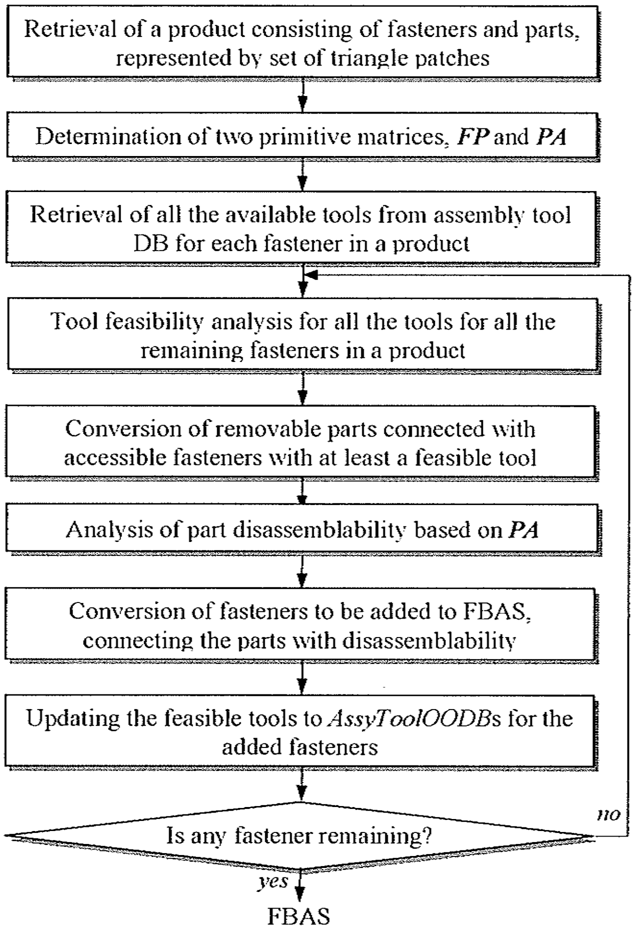

Retrieval of a product consisting of fasteners and parts, represented by set of triangle patches
Determination of two primitive matrices, FP and PA
Retrieval of all the available tools from assembly tool DB for each fastener in a product
Tool feasibility analysis for all the tools for all the remaining fasteners in a product
Conversion of removable parts connected with accessible fasteners with at least a feasible tool
Analysis of part disassemblability based on PA
Conversion of fasteners to be added to FBAS, connecting the parts with disassemblability
Updating the feasible tools to AssyToolOODBs for the added fasteners
Is any fastener remaining?
no
yes
FBAS

**Figure 6.2**: Overall procedure to construct an FBAS in this chapter.

The removable parts ($P^r$) are simply determined by following Equation (6.1) when a set of accessible fasteners is denoted as $F^a$.

$$P^r = \bigcup \left\{ j \in N \mid \sum_{i \in M - F^a} FP_{i,j} = 0 \right\} \qquad (6.1)$$

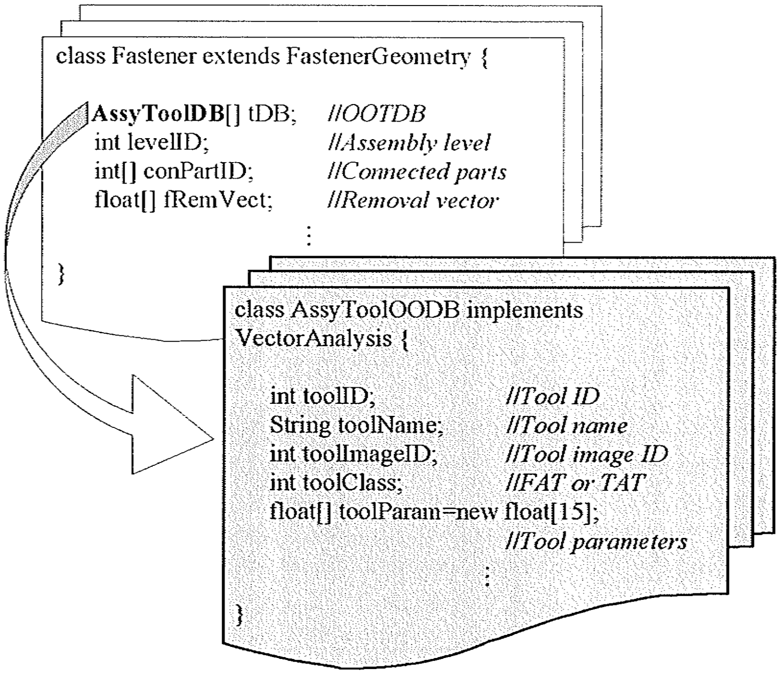
In Equation (6.1), the set $P^r$ represents the parts that may be connected with the accessible fasteners, or may not be connected with any fastener. To guarantee that these parts can be disassembled, the topological part disassemblability analysis should also be performed as shown in Figure 6.2. The set $P^r$ is updated by following Equation (6.2) based on the matrix $PA$.

$$P^r = \bigcup \left\{ i \in P^r \mid \bigcap_{j \in N} PA^{i,j} \neq \Phi \right\} \tag{6.2}$$

This process is to topologically ensure the parts' assemblability at an assembling stage. In Figure 6.2, the updated part set $P^r$ is used to recursively determine accessible fasteners $F^a$ by following Equation (6.3).

$$F^a = \bigcup_{j \in P^r} \left\{ i \in F^a \mid FP_{i,j} = 1 \right\} \tag{6.3}$$

The updated fastener set $F^a$ is subsequently added to an FBAS with the removal part information. The fasteners added to an FBAS selectively keep their feasible assembly tools, the ones that passed the assembly-tool feasibility analysis. This process shown in Figure 6.2 is repeatedly carried out until all fasteners in a product are added to an FBAS. In every loop, added fasteners and parts to an FBAS are subtracted from the sets $M$ and $N$, respectively. In particular, to efficiently store all information with respect to fasteners, a fastener class is defined via Java. Figure 6.3 shows the data structure of *Fastener* class with tool OODBs named *AssyToolOODB*s (Refer to Appendix A.14).



```
class Fastener extends FastenerGeometry {

    AssyToolDB[] tDB;     //OOTDB
    int levelID;          //Assembly level
    int[] conPartID;      //Connected parts
    float[] fRemVect;     //Removal vector
                  :
}
```

```
class AssyToolOODB implements
VectorAnalysis {

    int toolID;                 //Tool ID
    String toolName;            //Tool name
    int toolImageID;            //Tool image ID
    int toolClass;              //FAT or TAT
    float[] toolParam=new float[15];
                                //Tool parameters
                              :
}
```

**Figure 6.3**: Data structure of *Fastener* and *AssyToolOODB* classes used in this chapter.

As shown in Figure 6.3, *Fastener* class defined in this chapter is applied for each fastener, including such information as the fastener removal vector, assembly indenture level, and parts connected by it. *AssyToolOODB* class shown in Figure 6.3 stores tool information including a tool ID, tool name, tool image, and tool type based on the tool classification defined in Chapter 4. In particular, the *AssyToolOODB* class does not store tool geometry while the *Fastener* class stores fastener geometry. Instead, the *AssyToolOODB* class has methods of dynamically generating a 3D tool model via the 15-tool parameters of a tool retrieved from the assembly tool DB. To help to generate the 3D tool model, moreover, an interfaced class named *VectorAnalysis* class in Figure 6.3 provides various functions for vector operations and transformations.

## 6.3 Genetic Algorithm-Based Assembly Planning

In this research, a GA-based approach is used to search an optimal assembly sequence with tool selection. Many GA-based approaches have been used successfully in the field of assembly planning (Chen and Liao, 1999; Chen and Liu, 2001; Guan *et al.*, 2002; Smith and Smith, 2002; Tseng *et al.*, 2004). GA has been proved superior to traditional graph or matrix-based searching methods.

A GA normally consists of three important components: the chromosome representation, the fitness function, and the GA operators. The chromosome representation and the fitness function transform a real physical problem to the *language* of a GA. The GA operators mainly influence the efficiency and convergence of the algorithm. The following sections discuss these three components of the proposed GA-based assembly planning.

### 6.3.1 Chromosome representation and overall procedure

A two-tier GA shown in Figure 6.4 is introduced to search a global optimal solution effectively. As shown in Figure 6.4, a chromosome is defined as a series of fasteners in an assembly sequence. Each fastener becomes *gene* of a chromosome, which will be manipulated by GA operators. The initial population in Figure 6.4 is randomly generated with all the fasteners in a product. The fasteners in the initial population are sorted and divided by the assembly levels of a constructed FBAS. This process leads to several sub-chromosomes based on the assembly indenture levels. Then, the assembly level-based GA (the first tier GA) runs for each sub-chromosome consisting of fasteners with the same assembly indenture level. When this process is done for the entire sub chromosomes divided, they are combined to form a complete chromosome. Subsequently, the process generates a locally optimized initial population for the next searching process. Based on this population, the product-based GA (the second tier GA) runs to find the optimal assembly sequence in a reduced number of generations.

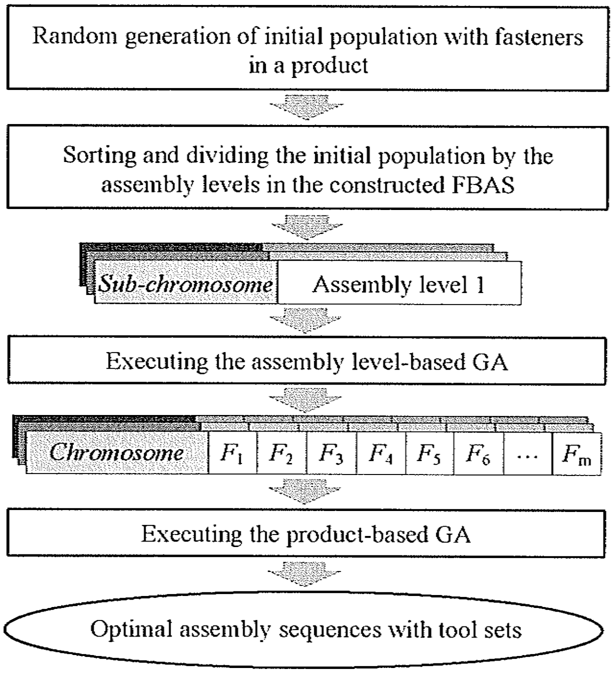

Random generation of initial population with fasteners in a product
Sorting and dividing the initial population by the assembly levels in the constructed FBAS
Sub-chromosome
Assembly level 1
Executing the assembly level-based GA
Chromosome
$F_1$
$F_2$
$F_3$
$F_4$
$F_5$
$F_6$
...
$F_m$
Executing the product-based GA
Optimal assembly sequences with tool sets

**Figure 6.4**: Two-tier GA used in this research.

## 6.3.2 Fitness function

A *fitness* value is assigned to each chromosome to measure quality of an assembly sequence. The fitness value is determined from the value of the objective function. The higher fitness value represents the better assembly sequence. The fitness function (*f*) in this research is determined as follows.

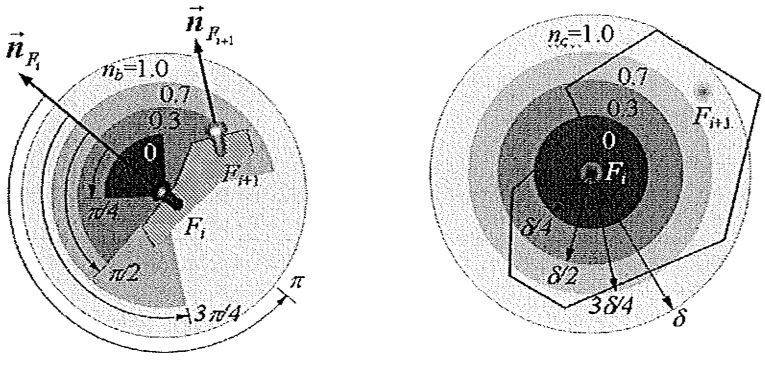
$$f = 2m - \left(w_a n_a + w_b n_b + w_c n_c\right) \tag{6.4}$$

Where, $w_a$ is the weight of tool-change property, $w_b$ is the weight of direction-change property, and $w_c$ is the weight of distance property. These weights are manually set in the developed system. In addition, $n_a$, $n_b$ and $n_c$ have a discrete value between 0 and 1. $n_a$ is determined via tool information while $n_b$ and $n_c$ depend on fastener positions. The details are listed as follows:

$n_a$: 1 if a tool changes between the two following fasteners, $F_i$ and $F_{i+1}$ in an assembly sequence; otherwise, 0.

$n_b$: A pre-defined discrete value between 0 and 1, which is based on the criterion of a direction-change by two removal vectors of the following fasteners, $F_i$ and $F_{i+1}$ in an assembly sequence (Refer to Figure 6.5(a)).

$n_c$: A pre-defined discrete value between 0 and 1, which is based on the criterion of a distance from a fastener $F_i$ to the next fastener $F_{i+1}$ in an assembly sequence (Refer to Figure 6.5(b)).



(a) (b)

**Figure 6.5**: Example of criteria and pre-defined discrete values for (a) $n_b$ and (b) $n_c$.

As shown in Figure 6.5(a), the four discrete values for $n_b$ are predefined by the angular criterion between two removal vectors of the following fasteners, $F_i$ and $F_{i+1}$ in an assembly sequence. Similarity, $n_c$ shown in Figure 6.5(b) also has the four discrete values based on a given distance criterion ($\delta$) in this research.

## 6.3.3 GA operators

There are three major operators used by most GA-based approaches, which are selection, crossover and mutation (Onwubiko, 2000). These three operators are used to manipulate the genetic material or *gene* in a chromosome for reproduction of initial population, or a new population. Similarly, the GA operators used in this research include a population selection, a crossover and two mutations-swap and inversion mutations. These GA operators have been successfully used in searching optimal sequences for assembly planning, a non-deterministic optimization problem. The features of these operators are described in the following sections.

### *6.3.3.1. Population selection*

Population selection uses the roulette wheel method to generate mating pool from the previous generation (Gen and Cheng, 1997). The roulette wheel is divided into slots weighted in proportion to the fitness values of the chromosomes. The wheel is spun and the selected member is the slot corresponding to the final position of the spinner. Clearly, the chromosomes with higher fitness values have higher chances of being selected to the mating tool.

### *6.3.3.2. Crossover*

Based on a given probability, the crossover operator selects two parent chromosomes. A crossover point is randomly chosen, which divides upper and lower nodes in a parent chromosome. After a *child* is identically copied from the parent chromosome, its upper nodes remain the same while its lower nodes are rearranged to follow the sequence in the other parent chromosome. By doing so, the genetic traits of the parent chromosomes are preserved in the *child*. As shown in Figure 6.6, for instance, *child* 1 consisting of five fasteners inherits the upper fasteners, $F_1$ and $F_2$, from *parent* 1 and the rest of the fasteners, $F_3$, $F_4$ and $F_5$, are rearranged by the fastener sequence in *parent* 2.

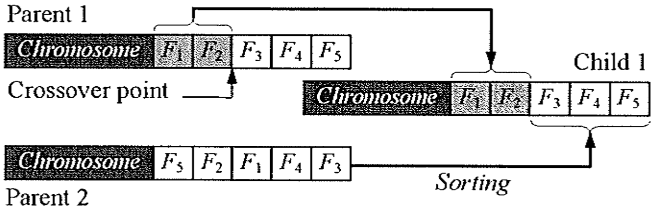

Parent 1
Chromosome
$F_1$ $F_2$ $F_3$ $F_4$ $F_5$
Crossover point
Child 1
Chromosome
$F_1$ $F_2$ $F_3$ $F_4$ $F_5$
Chromosome
$F_5$ $F_2$ $F_1$ $F_4$ $F_3$
Parent 2
Sorting

**Figure 6.6**: Crossover operator.

### *6.3.3.3. Swap mutation*

This operator swaps two nodes in a chromosome. As shown in Figure 6.7, this operator randomly chooses an assembly level in a chromosome. This chromosome is selected based on a given probability. The two fasteners, $F_2$ and $F_4$ shown in Figure 6.7 are randomly chosen, and swapped with each other within the assembly level of the selected chromosome.

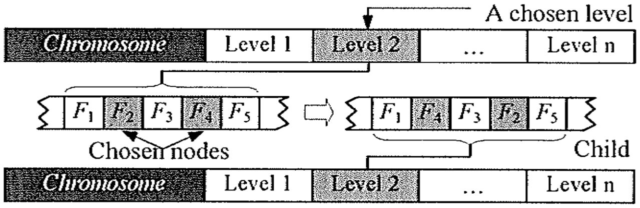


**Figure 6.7**: Swap mutation operator.

### *6.3.3.4. Inversion mutation*

Inversion mutation is a reordering operator applied to the nodes of a chromosome. As shown in Figure 6.8, this operator randomly chooses an assembly level in a chromosome. This chromosome is selected based on a given probability. Then, this GA operator reverses the order of nodes between two randomly chosen positions within the selected assembly level of the chromosome. For instance, the fasteners $F_2$, $F_3$ and $F_4$ in Figure 6.8 are changed into $F_4$, $F_3$ and $F_2$ by inverting the nodes between positions 2 and 4.

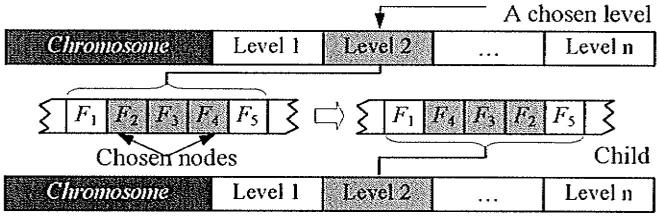
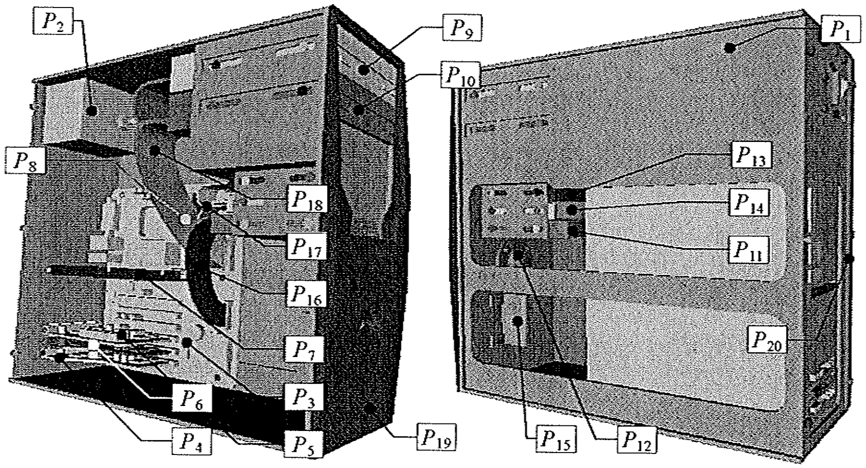


**Figure 6.8**: Inversion mutation operator.

## 6.4 Sample Application

An example is applied in this chapter to execute dynamic assembly planning with tool selection. As shown in Figure 6.9, the PC example consists of 21 parts and 50 fasteners. This CAD model consisting of the parts and fasteners is retrieved from the product DB via a system developed in this research. Each component also consists of a set of triangle patches, which were discussed in Chapter 5.



(a)

| Part ID | Part Name | Part ID | Part Name |
|---|---|---|---|
| $P_1$ | Base | $P_{12}$ | Speaker |
| $P_2$ | Power supplier | $P_{13}$ | Hard drive |
| $P_3$ | Main board | $P_{14}$ | Hard drive |
| $P_4$ | LAN card | $P_{15}$ | Switch |
| $P_5$ | TV card | $P_{16}$ | IDE cable |
| $P_6$ | Sound card | $P_{17}$ | IDE cable |
| $P_7$ | Video card | $P_{18}$ | IDE cable |
| $P_8$ | RAM | $P_{19}$ | Front cover |
| $P_9$ | DVD-ROM drive | $P_{20}$ | Side cover |
| $P_{10}$ | CD-ROM drive | $P_{21}$ | Side cover |
| $P_{11}$ | Floppy drive | | |

(b)

**Figure 6.9**: A PC model used for dynamic assembly planning with tool selection (a) parts in the model, and (b) detailed part list.

The detailed fastener configuration is shown in Figure 6.10. The fasteners in Figure 6.10 consist of 37 screws (#6 cross-head) and 13 bolts (#8 hex-head). The same types of screws and bolts are respectively used for the PC model in this chapter. As shown in Figure 6.10, the bolts in the PC model are $F_1$~$F_7$ and $F_{45}$~$F_{50}$ while the screws are $F_8$~$F_{44}$.

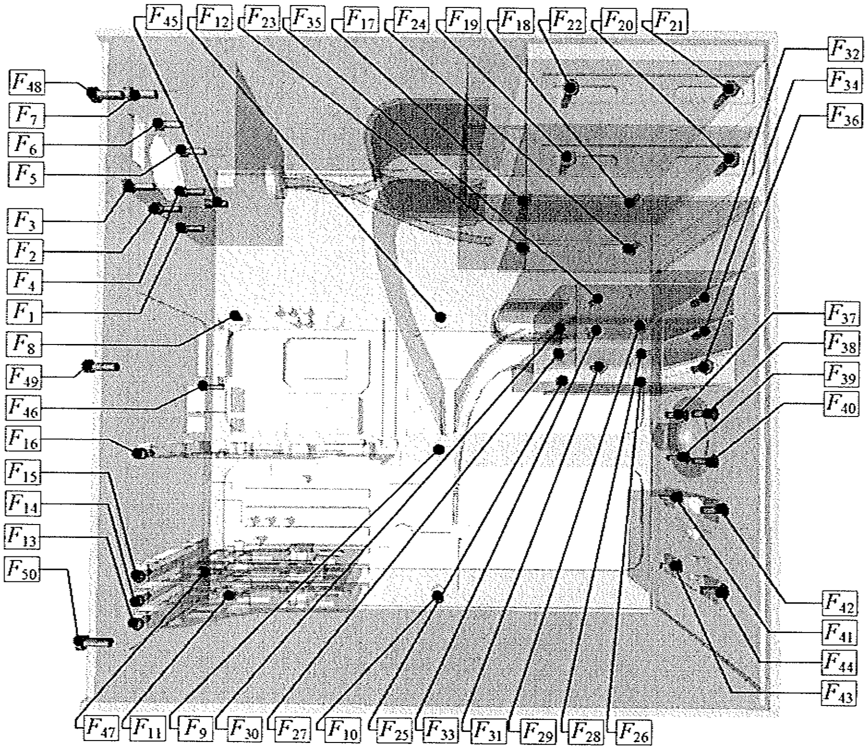


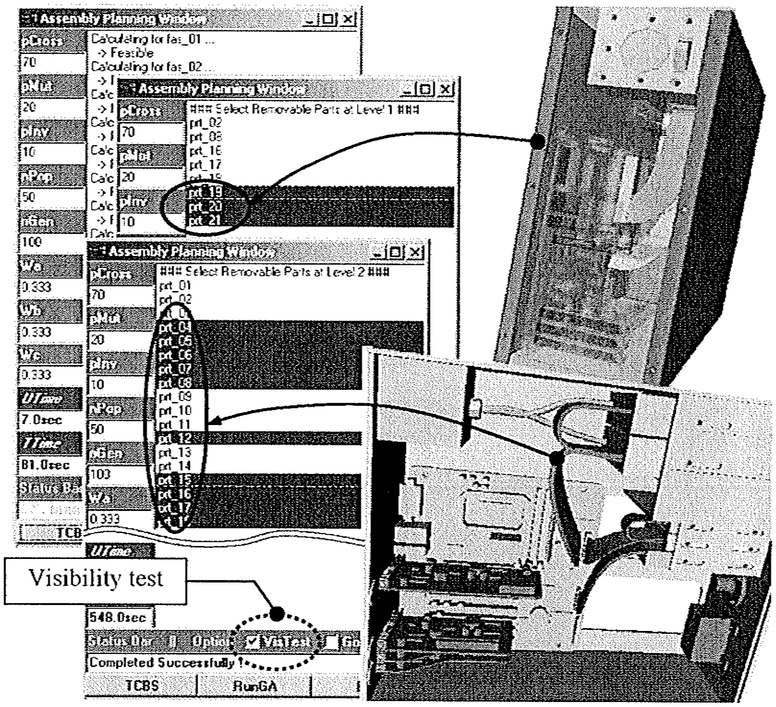
**Figure 6.10**: Fastener configuration in the tested PC model.

The first step of the tool selection-embedded assembly planning is to retrieve all available tools from the assembly tool DB to construct an FBAS for the tested model. For this example, 7 and 14 different assembly tools are respectively retrieved for assembling screws and bolts in the PC model. 7 tools for the screw type include screw drivers with various lengths, ratchet screw drivers, and socket drivers combined with speeders and ratchets, whereas 14 tools for the bolt type include nut drivers, various wrenches, and sockets combined with speeders and ratchets. For each fastener in the PC model, the retrieved tools are initially stored at its *AssyToolOODB*s (Refer to Appendix A.14) defined in this research.

By constructing GAC$^d$s for all remaining fasteners in the tested model, at each assembly level, the assembly-tool feasibility analyses for their *AssyToolOODB*s are performed to determine a set of fasteners ($F^a$) with tool feasibility. The set of fasteners are converted to potentially removable parts ($P^r$) by Equation (6.1). By Equation (6.2), the parts are then tested to check topological disassemblability, and only assemblable parts are chosen from them. By Equation (6.3), subsequently, these assemblable parts are converted into the fasteners that are added to an FBAS. In this example, three assembly levels are defined for assembling the PC model. As shown in Figure 6.11, for instance, the second assembly level includes three integrated drive electronics (IDE) cables, RAM, cards, speaker, and switch. The first assembly level includes three covers, a front and two side covers. This level will be the final stage of assembly because of the used top-down approach (*i.e.*, reverse assembling direction) of constructing an FBAS.

During the process, the *AssyToolOODB*s of a fastener added to an FBAS are also updated to store the newly defined feasible tools of the fastener, which are passed by the assembly-tool feasibility analyses. The process to construct an FBAS for the tested model is a repeated and time-consuming process because assembly-tool feasibility can be different in accordance with a generated part configuration-a GAC$^d$. In this example, the total of 680 CPU seconds has elapsed for the complete computation in constructing the

FBAS of the tested model. This CPU time is not including the time to form two primitive matrices, *PA* and *FP*, which are pre-determined before constructing an FBAS. For each fastener with *AssyToolOODBs*, the mean time of the tool feasibility analyses is 6 CPU seconds that includes the construction time of a GAC[d].



Calculating for fas_01 ...
-> Feasible
Calculating for fas_02 ...

pCross 70
pMut 20
pInv 10
nPop 50
nGen 100
Wa 0.333
Wb 0.333
Wc 0.333
UTime 7.0sec
TTime 81.0sec

### Select Removable Parts at Level 1 ###
prt_02
prt_08
prt_16
prt_17
prt_18
prt_19
prt_20
prt_21

### Select Removable Parts at Level 2 ###
prt_01
prt_02
prt_03
prt_04
prt_05
prt_06
prt_07
prt_08
prt_09
prt_10
prt_11
prt_12
prt_13
prt_14
prt_15
prt_16
prt_17
prt_18

pCross 70
pMut 20
pInv 10
nPop 50
nGen 103
Wa 0.333
UTime 548.0sec

Completed Successfully !

**Figure 6.11**: Construction of the FBAS based on assembly-tool feasibility, tool visibility and part disassemblability analyses.

In addition to the assembly-tool feasibility and topological part disassemblability analyses, the fastener visibility analysis as an option can be chosen in constructing an FBAS. As shown in Figure 6.11, a GUI developed in this research provides a user with a check box to select the visibility analysis option. The analysis virtually checks whether a fastener will be visible from the outside by an assembler during an assembly operation. A constructed $GAC^d$ is also used to simply find its empty pixel that does not have depth information. It is not a necessary condition but a sufficient condition for reliable assembly that may require tasks such as correct tool positioning or tool application monitoring. Figure 6.12 shows the notion of the fastener visibility and assembly-tool feasibility used in this research. As shown in Figure 6.12, a fastener in a part configuration can be removed because an assembly tool is feasible although it is not visible to an assembler. However, this fastener visibility may be critical in robot-assist assembly or automated assembly because a robot arm has a reduced number of DOF compared with a human' arm.

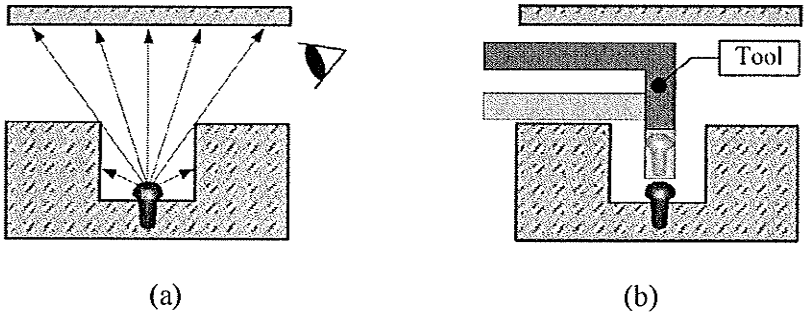
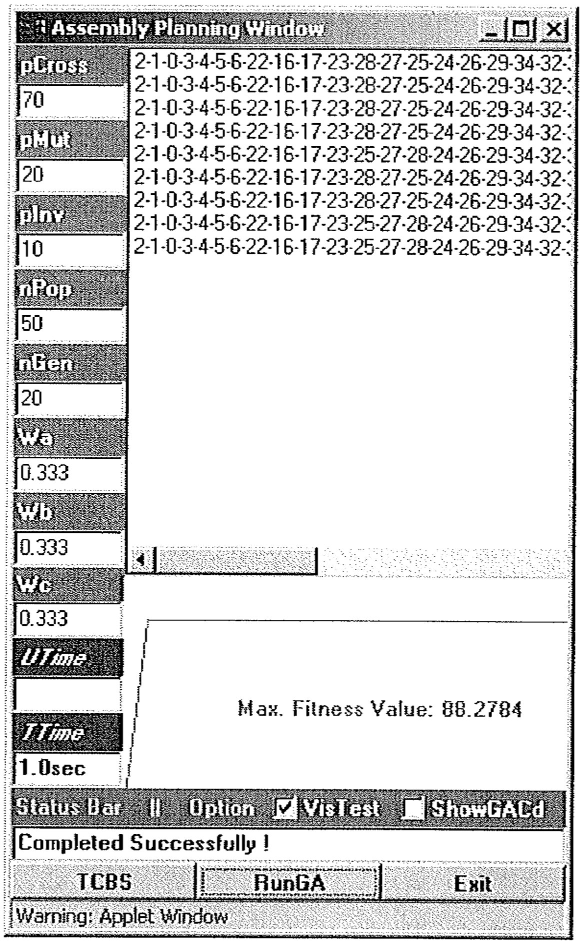


**Figure 6.12**: Notion of the fastener visibility and assembly-tool feasibility in this research (a) no fastener visibility, and (b) assembly-tool feasibility.

Based on the defined FBAS for the PC model, the two-tier GA-based approach proposed generates a group of 9 optimal assembly sequences with the highest fitness value. Figure 6.13 shows the resulting assembly sequences for this example via the system developed. The detailed assembly sequences are also summarized in Table 6.1. These resulting sequences form fastener-based assembly sequences that define the optimal sequences of assembling fasteners using the tools. Parts required at each assembly level are positioned before assembling fasteners. After only 10 generations, the highest fitness value is achieved because of the initial population generated by the assembly-level based GA (the first tier GA) that runs in 50 generations. The used parameters in this assembly planning process are as follows: three probability values, namely *pCross*, *pMut*, *pInv* (70%, 20%, 10%), and three weight values, $w_a$, $w_b$, $w_c$ (0.33, 0.33, 0.33). These parameters are chosen by a user. As shown in Figure 6.13, the assembly planning for the PC model is formed via the GA-based assembly-planning window developed in this research.



pCross: 70
pMut: 20
pInv: 10
nPop: 50
nGen: 20
Wa: 0.333
Wb: 0.333
Wc: 0.333
UTime:
TTime: 1.0sec

```
2-1-0-3-4-5-6-22-16-17-23-28-27-25-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-28-27-25-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-28-27-25-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-28-27-25-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-25-27-28-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-28-27-25-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-28-27-25-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-25-27-28-24-26-29-34-32-
2-1-0-3-4-5-6-22-16-17-23-25-27-28-24-26-29-34-32-
```

Max. Fitness Value: 88.2784

Status Bar || Option ☑ VisTest ■ ShowGACd

Completed Successfully !

Warning: Applet Window

**Figure 6.13**: Assembly planning process for the PC model via the system developed.

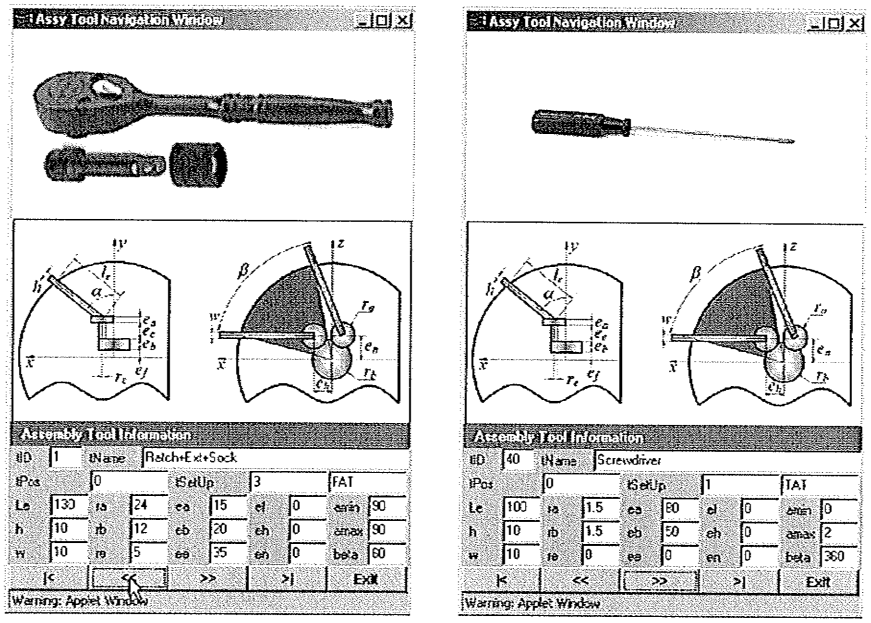
**Table 6.1**: Fastener-based assembly sequences generated for the PC model.

| No. | Generated fastener-based assembly sequence | Fitness value |
|---|---|---|
| 1 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{29}, F_{28}, F_{26}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_8, F_{11}, F_{10}, F_9, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{49}, F_{48}, F_{45}, F_{46}, F_{47}, F_{50}$ | 88.2784 |
| 2 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{29}, F_{28}, F_{26}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_8, F_{11}, F_{10}, F_9, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{45}, F_{48}, F_{49}, F_{50}, F_{47}, F_{46}$ | " |
| 3 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{29}, F_{28}, F_{26}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_8, F_{11}, F_{10}, F_9, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{50}, F_{47}, F_{46}, F_{45}, F_{48}, F_{49}$ | " |
| 4 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{29}, F_{28}, F_{26}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_{11}, F_{10}, F_9, F_8, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{46}, F_{47}, F_{50}, F_{49}, F_{48}, F_{45}$ | " |
| 5 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{26}, F_{28}, F_{29}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_{11}, F_{10}, F_9, F_8, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{49}, F_{48}, F_{45}, F_{46}, F_{47}, F_{50}$ | " |
| 6 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{29}, F_{28}, F_{26}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_{11}, F_{10}, F_9, F_8, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{49}, F_{48}, F_{45}, F_{46}, F_{47}, F_{50}$ | " |
| 7 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{29}, F_{28}, F_{26}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_8, F_{11}, F_{10}, F_9, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{46}, F_{47}, F_{50}, F_{49}, F_{48}, F_{45}$ | " |
| 8 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{26}, F_{28}, F_{29}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_{11}, F_{10}, F_9, F_8, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{45}, F_{48}, F_{49}, F_{50}, F_{47}, F_{46}$ | " |
| 9 | $F_3, F_2, F_1, F_4, F_5, F_6, F_7, F_{23}, F_{17}, F_{18}, F_{24}, F_{26}, F_{28}, F_{29}, F_{25}, F_{27}, F_{30}, F_{35}, F_{33}, F_{31}, F_{36}, F_{34}, F_{32}, F_{20}, F_{21}, F_{22}, F_{19}, F_{12}, F_8, F_{11}, F_{10}, F_9, F_{16}, F_{15}, F_{14}, F_{13}, F_{42}, F_{40}, F_{38}, F_{37}, F_{39}, F_{41}, F_{43}, F_{44}, F_{45}, F_{48}, F_{49}, F_{50}, F_{47}, F_{46}$ | " |

In Table 6.1, each assembly sequence generated for the PC model has several alternatives of complete assembly-tool sets. The alternatives are generated by the combination of the tool sets ($\boldsymbol{Ts}_j$, $\forall j \in M$) for all fasteners ($F_j$, $\forall j \in M$) in an assembly sequence. The set of tools $\boldsymbol{Ts}_j$ for each fastener $F_j$ initially includes all the feasible tools that are stored in its *AssyToolOODBs*. However, $\boldsymbol{Ts}_j$ is redefined during calculating the number of tool changes for the fitness function in Equation (6.4). For instance, if $\boldsymbol{Ts}_j \cap \boldsymbol{Ts}_{j+1}=\varnothing$ between two following fasteners $F_j$ and $F_{j+1}$ in an assembly sequence, the original $\boldsymbol{Ts}_{j+1}$ is applied in the calculation of the number of tool changes for the next two following fasteners $F_{j+1}$ and $F_{j+2}$. This makes an increase in the number of tool changes, which subsequently reduces the fitness value of the assembly sequence.

If $\boldsymbol{Ts}_j \cap \boldsymbol{Ts}_{j+1}\neq\varnothing$, on the other hand, the $\boldsymbol{Ts}_{j+1}$ becomes $\boldsymbol{Ts}_j \cap \boldsymbol{Ts}_{j+1}$. The redefined $\boldsymbol{Ts}_{j+1}$ is applied for the next two following fasteners $F_{j+1}$ and $F_{j+2}$ without any increase in the number of tool changes. In this case, previous tool sets $\boldsymbol{Ts}_i$ ($i=k$, ..., $j$, $k$: the previous point occurring at an increase in the number of tool changes) are also updated to be the same as the newly defined $\boldsymbol{Ts}_{j+1}$. For a generated optimal assembly sequence, subsequently, its alternatives of complete assembly-tool sets are formed by the combination of the redefined tool sets $\boldsymbol{Ts}_j$ ($\forall j\in \boldsymbol{M}$) during the time the GA-based assembly planning is repeatedly calculating the fitness function.

Every complete assembly-tool set generated in this example consists of two feasible tools because of the fitness function used in this chapter, which reduces the number of tool changes. Only two types of fasteners used in the tested model may be also another reason for the reduced number of tools constituting an assembly-tool set. Among the recommended toolset alternatives consisting of different types of two tools, a complete assembly-tool set for all the generated assembly sequences is simply chosen by the system developed in this research. The chosen assembly-tool set is displayed in the tool navigation window that is shown in Figure 6.14. This is based on a simple priority rule in selecting a tool among the tools with the same function. This rule may be formed by considering factors including tool cost, fastening time, fastener-damage effects of tools, ergonomics or operators' preferences, but those factors are not considered in this research. Instead, tools available for a fastener are arbitrarily pre-assigned different priorities in the assembly tool DB. As shown in Figure 6.14, two tools to be used for assembling the PC model are illustrated. The images of the tools are additionally retrieved from the assembly tool DB via the system developed in this research.



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| tID | 1 | tName | Ratch+Ext+Sock | | | | | | |
| tPos | 0 | | | tSetUp | 3 | | FAT | | |
| Le | 130 | ra | 24 | ea | 15 | el | 0 | amin | 90 |
| h | 10 | rb | 12 | eb | 20 | eh | 0 | amax | 90 |
| w | 10 | re | 5 | ee | 35 | en | 0 | beta | 60 |

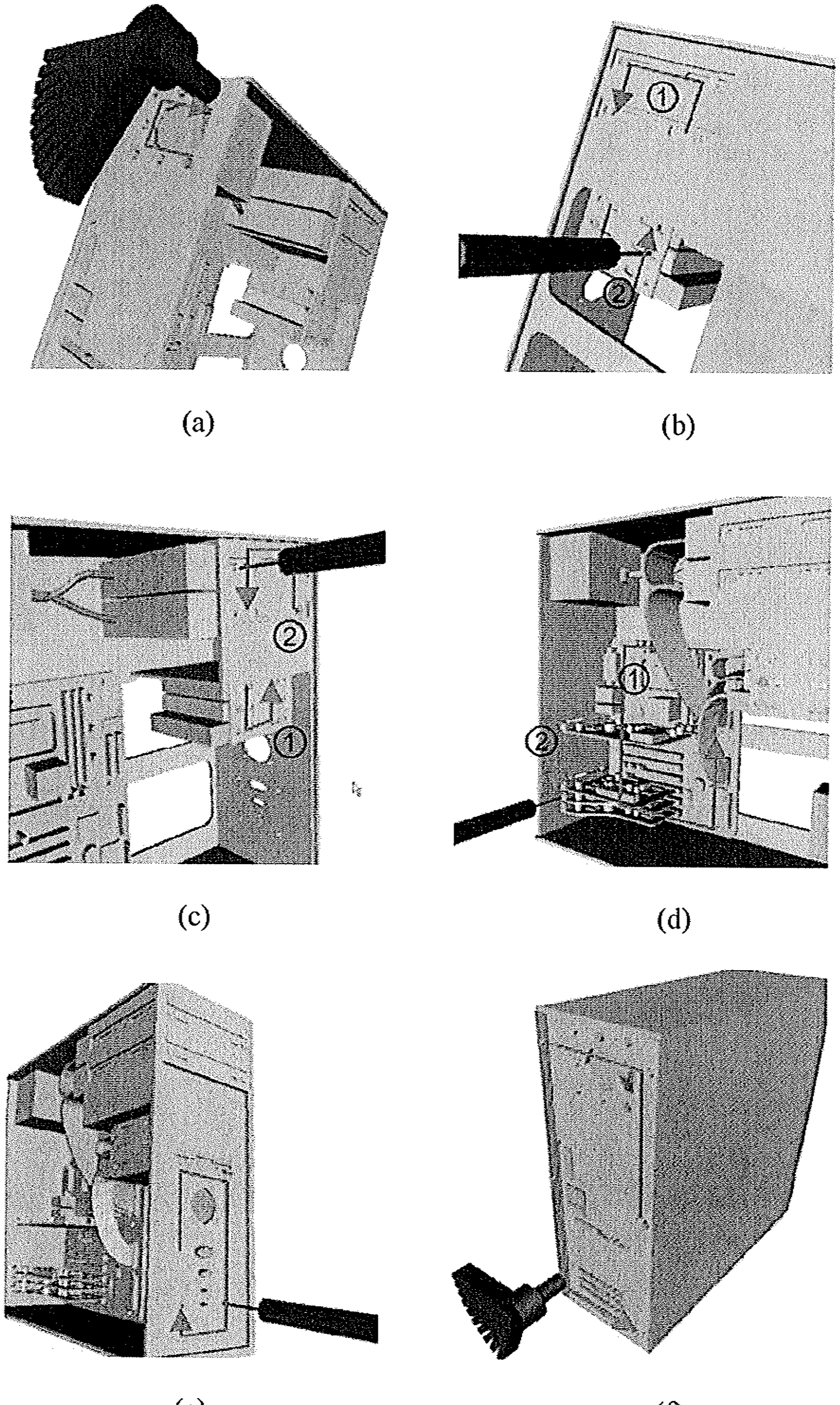
(a)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| tID | 40 | tName | Screwdriver | | | | | | |
| tPos | 0 | | | tSetUp | 1 | | TAT | | |
| Le | 100 | ra | 1.5 | ea | 80 | el | 0 | amin | 0 |
| h | 10 | rb | 1.5 | eb | 50 | eh | 0 | amax | 2 |
| w | 10 | re | 0 | ee | 0 | en | 0 | beta | 360 |

(b)

**Figure 6.14**: Chosen assembly-tool set via the system developed in this research (a) a ratchet with an extension and socket, and (b) a screwdriver.

Figure 6.15 shows a simulation process via the selected assembly-tool set for a generated assembly sequence. The red arrows shown in Figure 6.15 represent the assembly sequences of fasteners in assembling the PC model. In particular, the fastener assembly sequences in Figure 6.15 have specific patterns because of the fitness function used, which reduces direction-changes and distance between two following fasteners $F_j$ and $F_{j+1}$ in an assembly sequence.



(a)

(b)

(c)

(d)

(e)

(f)

**Figure 6.15**: Simulation process via a selected assembly-tool set for a generated assembly sequence (a) step 1 (b) step 2 (c) step 3 (d) step 4 (e) step 5, and (f) step 6.

## 6.5 Chapter Summary

This chapter presented a novel approach to dynamic assembly planning, which is here called tool selection-embedded optimal assembly planning. It generated a high fidelity assembly sequence and proper tools for the assembly based on a dynamic manufacturing environment.

An FBAS was constructed to model the assembly structure of a product. It is based on the assembly-tool feasibility and topological part disassemblability analyses, which were previously presented in Chapters 4 and 5, respectively. In addition, the fastener visibility analysis as an option could be chosen in constructing an FBAS. To analyze fastener visibility, a constructed $GAC^d$ was also used to simply search its empty pixel that does not have depth information. It is not the necessary condition but the sufficient condition for a reliable assembly that may require tasks such as correct tool positioning or tool application monitoring. In particular, an object-oriented fastener class with *AssyToolOODB*s was efficiently used to handle many retrieved tools from the assembly-tool DB during the process to construct an FBAS.

A two-tier GA was introduced for the tool selection-embedded optimal assembly planning in this chapter. Based on the constructed FBAS, the two-tier GA consists of two GA-based methods: an assembly level-based GA for creating the initial population and a product-based GA for generating an optimal assembly sequence. It optimized fastener-based assembly sequences in a reduced number of generations. Based on the experiment performed in this chapter, the results have demonstrated that the developed approach is efficient and practical to generate a high-fidelity optimal assembly sequence and to select assembly-tool sets based on a dynamic shop floor. A complete assembly planning can be performed with a large number of real assembly tools at a dynamic shop floor.

Based on the approach developed in this chapter, it is likely to achieve dynamic assembly planning in a Web-based PD&M environment. A high-fidelity assembly plan can be generated with its feasible tool alternatives in a networked collaborative environment. The generated assembly plan can be used for quickly exploiting manufacturing opportunities for a specific product at the inter-enterprise level, and for realizing assembly at the production level. In addition, the assembly plan can provide VP or VM systems with a reduced number of feasible *what-if* scenarios regarding assembly, and serve at supporting DFA to execute the practical assessment of assemblability, and a reliable estimation of assembly time and cost.

With respect to the research on the dynamic assembly planning, published papers in international journals and conferences are as follows.

**Chung, C.** and Peng, Q. (2006). Tool selection-embedded optimal assembly planning in a dynamic manufacturing environment. Computer-Aided Design (In Review).

**Chung, C.** and Peng, Q. (2005). Tool selection-embedded optimal assembly planning. The 2nd CDEN International Conference on Design Education, Innovation, and Practice. Kananaskis, Alberta, Canada, July 18-20.

**Chung, C.** and Peng, Q. (2006). Evolutionary sequence planning for selective-disassembly in de-manufacturing. International Journal of Computer Integrated Manufacturing (IJCIM). DOI: 10.1080/09511920500324647 (Online Available).

# Chapter 7

# Dynamic Disassembly Planning

This chapter presents an approach to non-destructive SD planning at the product end life level. The approach is based on the assembly-tool feasibility analysis and the part disassemblability analysis, which are presented in Chapters 4 and 5, respectively. To support design at the early stage of product development, this chapter also discusses a de-manufacturability analysis based on the proposed approach.

## 7.1 Proposed Approach

In this chapter, an approach is presented for non-destructive SD planning. The objective is to reduce the number of removals in disassembling a set of selected parts (***Ps***). The approach is based on the assembly-tool feasibility analysis and the part disassemblability, which are presented in Chapters 4 and 5, respectively. Table 7.1 illustrates comparisons between the SD planning and other procedural planning for product assembly or complete disassembly.

**Table 7.1**: Comparisons between SD planning and other procedural planning.

| Category | Procedural planning | SD planning |
|---|---|---|
| Ease to structure | Easy | Difficult |
| Variables used for planning | Each part | Each part and all part combinations |
| Modularity | Important | Extremely important |
| Assembly-tool feasibility | Important | Extremely important |
| Part disassemblability | Important | Extremely important |

In Table 7.1, five categories are used to compare two types of planning approaches, which are namely ease to structure, variables used for planning, modularity, assembly-tool feasibility, and part accessibility (or part disassemblability). For procedure planning like assembly or complete disassembly planning, the structure of a product is modelled by various techniques (Tang *et al.*, 2000) including AND/OR graphs, direct graphs and disassembly Petri-Nets. It forms a static product structure that is not changed. However, this structure cannot be applied in SD planning because of the dynamic nature of the SD planning. SD only considers the disassembly of arbitrarily selected parts in a product. A product structure in SD planning can be varied, depending on the selected parts and the part configuration formed by the rest of the parts of the product. Moreover, a removal disassembled at a time may form a module of parts. It reduces efforts to separate ***Ps*** in a partial and non-procedural disassembly.

For procedural planning, each part in a product is regarded as a variable. The variable is used for an optimization process to generate an economical sequence. However, SD planning requires parts in a product and their combinations for optimization variables.

The variables also rely on the selected parts in a product, unnecessary for including the whole parts in SD, a partial disassembly.

Modularity is often analyzed in procedural planning. The modularity analysis defines modules for the sets of components strongly related to detachability, functionality, stability or other related criteria (Lambert, 2003). In SD planning, this modularity analysis can reduce the number of removals in a product. This depends on the selected parts and the part configuration formed by the rest of the parts of the product. A constructed module allows for rapid access to parts selected for separation.

Assembly-tool feasibility and part disassemblability in SD are more important than those in assembly or complete disassembly. In a partial and non-procedural disassembly, these two factors form a critical constraint to generate a feasible SD sequence ($\mathscr{DS}$). Moreover, part disassemblability should be satisfied for a part and the part clusters, since both of them are being disassembled from a product.

The proposed approach here aims to generate a near-optimal SD sequence that is based on a dynamic DM environment. It is reasonable to find a *good* feasible solution, which can be efficiently computed. SD sequence planning is mathematically a NP-hard problem that is hard to find a known efficient algorithm for solving the problem, and quite unlikely that one exists. To demonstrate the efficiency of the proposed approach, several application examples are provided at the end of this chapter. For aiding design at the early stage of product development, this chapter also discusses a de-manufacturability analysis, *the maintainability analysis*, via the proposed approach. The following are major contents included in this chapter.

***Part disassembly-route matrix***. A matrix named part disassembly-route matrix (***PD***) is formed with the matrix *PA*. The matrix ***PD*** is a predefined precedence graph describing the shortest paths to all the exterior nodes. It is here used to define a module based on part disassemblability. The construction of this matrix involves two steps: rating the disassemblability level for each part, and determining its optimal immediate predecessors in the next higher disassemblability level.

***Selective-disassembly sequence planning***. An approach is proposed for SD planning. This is based on the matrix ***PD*** and the assembly-tool feasibility analysis presented in Chapter 4. Based on this approach, a near optimal SD sequence is generated for the set ***Ps***.

***Maintainability analysis based on the proposed approach***. As a de-manufacturability analysis, the maintainability analysis is discussed in this chapter. This analysis based on US military specifications is efficiently combined with the matrix-based heuristic approach to SD sequence planning developed in this research.

## 7.2 Part Disassembly-Route Matrix

The matrix ***PD*** forms a predefined precedence graph describing the shortest paths to all exterior nodes. This will serve as a guidance to search the disassembly chain ($\mathscr{DC}$) of a part being disassembled. Based on the matrix *PA*, the construction of the matrix ***PD*** involves two steps: rating the disassemblability level for each part, and determining its optimal immediate predecessors to the exterior nodes.

A disassemblability level in this chapter is similar to a wave that propagates inwards from exterior nodes. By checking disassemblability (Refer to Equation (5.3)) of a part, the rating process is carried out until all parts are given disassemblability levels that are assigned to the diagonal elements in the matrix *PD*. Algorithm 7.1 describes the procedure to rate part disassemblability levels in this research.

**Algorithm 7.1**: Determination of part disassemblability levels

```
Input $PA$ and $N$
Initialize the matrix $PD\ (n \times n)$
Let $l = 1$                                   /* disassemblability level */
Do while $(N \neq \Phi) \Rightarrow$
    Let $S = \Phi$                            /* a temporary set of parts */
    For $i = 1,\ldots,n \Rightarrow$
        Let $j = N_i,\ N_i \in N$
        Calculate $\Delta_j$ for $P_j$ within $N$
        If $\Delta_j = 1$ then
            $PD_{jj} = l$ and $S = S \cup \{j\}$
    Let $N = N - S$ and $l = l + 1$
```

Algorithm 7.1 is conceptually based on the *onion peeling approach* starting from exterior nodes of an assembly product and preceding inwards. Based on this notion, a part with the lowest disassemblability level (*i.e.*, $l$=1 in Algorithm 7.1) has the highest priority to be disassembled from a product. In the next step, each part searches its optimal immediate predecessors (*i.e.*, a set of parts) with the least number of removals. The removals consist of parts and their fasteners, both of which should be removed in advance. All possible part combinations are examined for a part to find an optimal disassembly path to the exterior nodes. To reduce the searching time, in particular, this process is repeatedly executed with only parts in two disassemblability levels, $l$ and $l$+1.

This subsequently forms a set of sub-processes, each of which determines the shortest disassembly path of a part to the next lower disassemblability level.

Algorithm 7.2 describes the procedure to determine optimal immediate predecessors for a part in this research.

**Algorithm 7.2**: Determination of immediate predecessors

```
Input $PA$, $FP$, $N$ and $M$
Let $N' = N$                                                      $/ *$ a temporary set of parts $* /$
Let $l = 1$                                                       $/ *$ disassemblability level $* /$
Find $\boldsymbol{S}^l = \{S^l{}_1, S^l{}_2, \cdots, S^l{}_q\}$, $S^l{}_q \in N$            $/ *$ a set of parts in the level $l$ $* /$
Do while $(N' \neq \Phi) \Rightarrow$
    Find $\boldsymbol{S}^{l+1} = \{S^{l+1}{}_1, S^{l+1}{}_2, \cdots, S^{l+1}{}_r\}$, $S^{l+1}{}_r \in N'$   $/ *$ a set of parts in the level $l+1$ $* /$
    For $i = 1, \ldots, r \Rightarrow$
        Let $u$=$S^{l+1}{}_i$, $S^{l+1}{}_i \in \boldsymbol{S}^{l+1}$
        Let $n^{IP} = n + m$            $/ *$ maximized number of immediate predecessors $* /$
        Let $\boldsymbol{O}^{IP} = \Phi$
                $/ *$ a set of optimized immediate predecessors to the next lower level $* /$
        For $k = 1, \ldots, q \Rightarrow$                                  $/ *$ draw number $* /$
            Generate a family set $\breve{\boldsymbol{S}}$ via combinations of $k$-parts in the set $\boldsymbol{S}^l$
            For $j = 1, \ldots, \{q!/k!(q\text{-}k)!\} \Rightarrow$
                Let $n' = n^{IP}$
                Let $N'' = N - \breve{\boldsymbol{S}}^j$, $\breve{\boldsymbol{S}}^j \subset \breve{\boldsymbol{S}}$
                Calculate $\Delta_u$ for $P_u$ within $N''$
                If $\Delta_u = 1$ then
                    Calculate $n'$
                                $/ *$ total number of removals in disassembling $P_u$ $* /$
                    If $n' < n^{IP}$ then
                        $n^{IP} = n'$ and $\boldsymbol{O}^{IP} = \breve{\boldsymbol{S}}^j$
    Let $N = N - \boldsymbol{S}^l$ and $N' = N' - \boldsymbol{S}^{l+1}$
    Let $\boldsymbol{S}^l = \boldsymbol{S}^{l+1}$ and $l = l + 1$
```

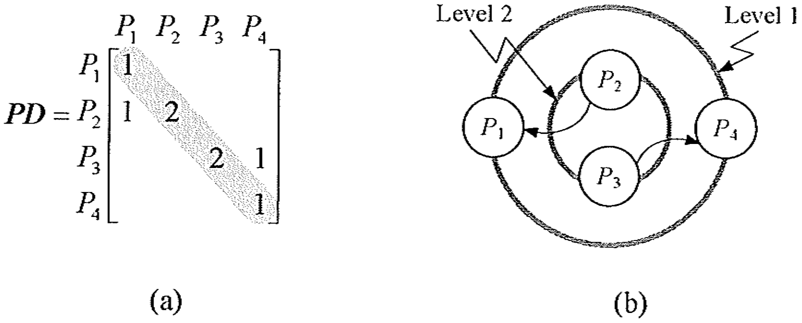
Two sets (*i.e.*, $\boldsymbol{S}^{l}$ and $\boldsymbol{S}^{l+1}$) of parts in the disassemblability levels $l$ and $l$+1 are considered in Algorithm 7.2. The two sets are initially formed with the parts in the first and second disassemblability levels. To determine the shortest disassembly path of a part $\boldsymbol{S}^{l+1}{}_{i}$ to the next lower disassemblability level, the disassemblability by Equation (5.3) is checked for all possible sets (*i.e.*, a family set $\breve{\boldsymbol{S}}$ for $k = 1, 2, \cdots, q$ in Algorithm 7.2) of parts in the level $l$. The family set $\breve{\boldsymbol{S}}$ is generated via a function developed in this research, which is named *part_set_generation* ($q$, $k$, $\boldsymbol{S}^{l}$).

The number ($n^{r}$) of removals with respect to $P_i$ is calculated as follows.

$$n^{r} = \sum_{j} FP_{j,i} + 1, \forall j \in \boldsymbol{M} \tag{7.1}$$

In Equation (7.1), 1 is represented as the removal of $P_i$ itself, and a part and a fastener are equivalently regarded in calculating a number of removals. Once a part obtains its optimized immediate predecessors (*i.e.*, $\boldsymbol{O}^{IP}$ in Algorithm 7.2) to the next lower level, the parts in the set $\boldsymbol{O}^{IP}$ are assigned as binary values to the matrix ***PD***.

Figure 7.1 shows an example of constructing the matrix ***PD***. Based on Algorithms 7.1 and 7.2, the ***PD*** shown in Figure 7.1(a) is constructed for the product illustrated in Figure 5.2(b). The diagonal elements (*i.e.*, $PD_{i,i}$) of ***PD*** in Figure 7.1(a) represent the defined part disassemblability levels. The matrix ***PD*** states the disassembly routes for parts in a product. As shown in Figure 7.1(b), for instance, the parts $P_2$ and $P_3$ on disassemblability level 2 can be disassembled after disassembling the parts $P_1$ and $P_4$, respectively. In this research, these are also represented as disassembly chains $\mathscr{DC}_2$ and $\mathscr{DC}_3$ for parts $P_2$ and $P_3$, respectively. The $\mathscr{DC}_i$ for disassembling $P_i$ includes all the parts to be sequentially removed. Therefore, the determined disassembly chains for parts $P_2$ and $P_3$ become $\mathscr{DC}_2=\{P_1, P_2\}$ and $\mathscr{DC}_3=\{P_4, P_3\}$, respectively.



$$PD = \begin{array}{c|cccc} & P_1 & P_2 & P_3 & P_4 \\ \hline P_1 & 1 & & & \\ P_2 & 1 & 2 & & \\ P_3 & & & 2 & 1 \\ P_4 & & & & 1 \end{array}$$

(a)

(b)

**Figure 7.1**: Example of constructing the matrix *PD* (a) *PD* constructed for the product illustrated in Figure 5.2(b), and (b) diagram of the disassembly routes for the parts.

Algorithm 7.3 describes the procedure to determine the disassembly chain of a part selected for separation.

**Algorithm 7.3**: Determination of disassembly chain based on the matrix *PD*

```
Input $PD$, $N$ and $P_i$                  /* $P_i$ : a selected part for separation */
Let $\mathscr{DC} = \Phi$                            /* disassembly chain */
Let $S = \{P_i\}$                             /* a temporary set of parts */
Do while $(S \neq \Phi) \Rightarrow$
    $\mathscr{DC} = \mathscr{DC} \cup S$
    Let $S = \bigcup_{p \in S} \{q \in N \mid PD_{p,q} = 1, p \neq q\}$
Sort all the parts of $\mathscr{DC}$ in ascending order of the disassemblability level
```

## 7.3 Selective-Disassembly Sequence Planning

The matrix-based heuristic approach to SD sequence planning is performed based on the matrices *FP* and *PD*, and the assembly-tool feasibility analysis presented in Chapter 4.

For a given set ***Ps***, planning starts with determining a new part set (***Np***) consisting of all parts that are associated with the set ***Ps*** in terms of topological part disassemblability and fastener connectivity.

Until the set ***Np*** $(Np \subseteq N)$ has no more increase in its cardinality, a searching process is repeated in the matrices ***PD*** and ***FP*** via the following sequences: determining the disassembly chain $\mathscr{DC}$s for the set ***Np*** (initially ***Np=Ps***) $\Rightarrow$ finding fasteners connecting the parts in the $\mathscr{DC}$s $\Rightarrow$ finding new parts connected with the fasteners $\Rightarrow$ adding the parts to the set ***Np***. Subsequently, the parts in the determined set ***Np*** only participate in the SD sequence planning for the given set ***Ps***.

Figure 7.2 shows the procedure of SD sequence planning developed in this research. As shown in Figure 7.2, the first step is to find fasteners via the matrix ***FP***, the fasteners are related to the parts in the set ***Np***. These fasteners are used to retrieve available tools from assembly tool DB in a dynamic DM environment. Similar to the process of dynamic assembly planning presented in Chapter 6, these retrieved tools are initially stored in *AssyToolOODB*s (Refer to Appendix A.14). Then, accessible fasteners $F^a$ are determined via the assembly-tool feasibility analysis presented in Chapter 4. These are used to collect parts from ***Np*** via the matrix ***FP***, and the collected parts constitute a new set of parts (***Ps'***) in Figure 7.2. Subsequently, each part in the set ***Ps'*** is used to construct a $\mathscr{DC}$to determine whether it can form a subassembly or module. In particular, the parts in the set ***Ps'*** are sorted in descending order of the disassemblability level based on the matrix ***PD*** so that it is likely to construct the largest subassembly (or module of parts) for the first time. This helps minimize the number of removals in SD.

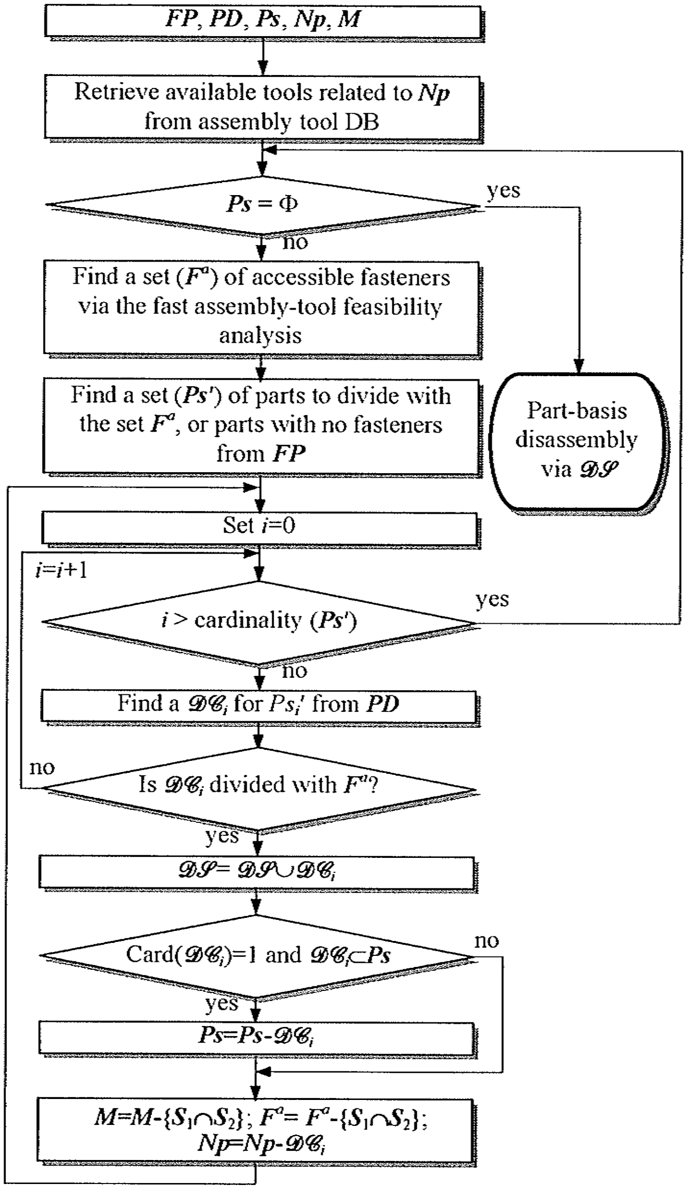

$FP, PD, Ps, Np, M$
Retrieve available tools related to $Np$ from assembly tool DB
$Ps = \Phi$
yes
no
Find a set ($F^a$) of accessible fasteners via the fast assembly-tool feasibility analysis
Find a set ($Ps'$) of parts to divide with the set $F^a$, or parts with no fasteners from $FP$
Part-basis disassembly via $\mathcal{DS}$
Set $i$=0
$i$=$i$+1
$i$ > cardinality ($Ps'$)
yes
no
Find a $\mathcal{DC}_i$ for $Ps_i'$ from $PD$
no
Is $\mathcal{DC}_i$ divided with $F^a$?
yes
$\mathcal{DS} = \mathcal{DS} \cup \mathcal{DC}_i$
Card($\mathcal{DC}_i$)=1 and $\mathcal{DC}_i \subset Ps$
no
yes
$Ps = Ps - \mathcal{DC}_i$
$M = M - \{S_1 \cap S_2\}$; $F^a = F^a - \{S_1 \cap S_2\}$;
$Np = Np - \mathcal{DC}_i$

**Figure 7.2**: Procedure of the SD sequence planning proposed in this research.

As shown in Figure 7.2, once a subassembly (or module) $\mathcal{DC}_i$ for a part $Ps_i'$ ($Ps_i' \in \boldsymbol{Ps}'$) is recommended via the matrix $\boldsymbol{PD}$, a test is performed to determine whether the subassembly can be disassembled by the currently available fasteners $F^a$. Equation (7.2) shows the fastener availability check ($\Omega_i$) for the $\mathscr{DC}_i$ with sets $\boldsymbol{Np}$, $\boldsymbol{M}$ and $F^a$, and the matrix $\boldsymbol{FP}$.

$$\Omega i = \begin{cases} 1 & \text{if } \boldsymbol{S}_1 \cap \boldsymbol{S}_2 \subseteq \boldsymbol{F}^a \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

Where, $\boldsymbol{S}_1 = \bigcup_{q \in \mathscr{DC}_i} \{p \in M \mid FP_{p,q} = 1\}$, and $\boldsymbol{S}_2 = \bigcup_{q \in Np - \mathscr{DC}_i} \{p \in M \mid FP_{p,q} = 1\}$.

Subsequently, a subassembly satisfied to Equation (7.2) is added to the family set $\mathscr{DS}$, and this process repeatedly continues until $\boldsymbol{Ps}=\Phi$. Once the processes shown in Figure 7.2 are completely finished, part-basis disassemblies for parts in the set $\boldsymbol{Ps}$ are executed within the family set $\mathscr{DS}$, which are not individually isolated during the processes. The part-basis disassembly for a part $P_k$ in the set $\boldsymbol{Ps}$ is also determined by the process shown in Figure 7.2, but the set $\boldsymbol{Np}$ is replaced by $\mathscr{DS}_i$ ($\mathscr{DS}_i \subset \mathscr{DS}$), to which the part $P_k$ belongs.

# 7.4 Maintainability Analysis via the Proposed Approach

A quantitative maintainability analysis is incorporated for a de-manufacturability analysis in this chapter. The analysis is based on task 203 in the maintainability prediction standard (MIL-HDBK-472, 1966). MTTR (MIL-HDBK-472, 1966; MIL-STD-470B, 1989; NASA-STD-8729.1, 1998) is used as a system effectiveness characteristic of a sustainable product,with long EIDL.

The MTTR of a part in a product is the function of corrective maintenance task frequency and maintenance elapsed time including disassembly and assembly. The total MTTR for a product is calculated by the summation of the MTTRs of individual parts in a product. In particular, the corrective maintenance task frequency is determined by the MTBF of a part. A lower MTBF leads to a higher frequency in corrective maintenance. When a part with the low MTBF is disassembled, incurring many other part removals, the MTTR of the part will increase and result in increasing maintenance cost during the product operation. Therefore, a quantitative maintainability analysis is highly demanded at the design stage of product development. For a family set of parts, $\mathcal{DS}$ and a family set ($\mathcal{DF}$) of fasteners corresponding to the $\mathcal{DS}$, the following Equations (7.3) and (7.4) are used for a quantitative maintainability analysis.

$$\mathrm{MTTR} = \frac{\sum_i (\lambda_i \times \mathrm{ET}_i)}{\sum_i \lambda_i} \tag{7.3}$$

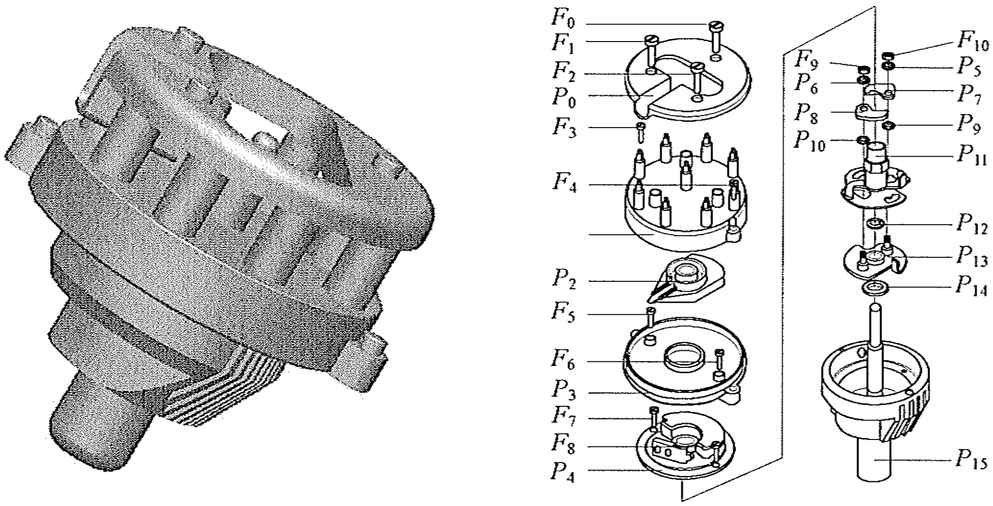
$$\mathrm{ET}_i = \sum_q^{N_{\mathcal{DF}}} \left\{ \sum_{P_j} \left[ \left( d(P_j) \times W(P_j) \right) \times K_p \right] + \sum_{F_k} \left[ \eta(F_k) \times T(F_k) \right] \right\} \text{for } \forall P_j \in \mathcal{DS}_i^q \text{ and } \forall F_k \in \mathcal{DF}_i^q \tag{7.4}$$

Where, $\mathcal{DS}_i$ is the family set for selectively disassembling $P_i$, $N_{\mathcal{DS}}$ is the total number of subsets in $\mathcal{DS}_i$, $\lambda_i \left(= 1/\mathrm{MTBF}_i\right)$ is the failure rate of $P_i$, $\eta(F_k)$ is the number of fasteners with respect to a fastener type $F_k$ in a subset $\mathcal{DF}_i^q$, and $T(F_k)$ is the standard time to disassemble the fastener of type $F_k$. The unit standard time $K_p$ to move $P_j$ with weight $W(P_j)$ and distance $d(P_j)$ is determined by a user via the system developed in this research.

# 7.5 Sample Applications

## 7.5.1 Sample application 1

An example product is tested to execute SD sequence planning. Figures 7.3(a) and (b) illustrate a magnetic breakerless distributor consisting of sixteen parts and eleven fasteners. As shown in Figure 7.3(b), fasteners $F_0$, $F_1$ and $F_2$ are used for fastening parts $P_0$ and $P_1$, fasteners $F_3$ and $F_4$ for parts $P_1$ and $P_3$, and fasteners $F_5$, $F_6$, $F_7$, and $F_8$ are connected to the part $P_{15}$. The fastener details are as follows: (1) $F_0 \sim F_2 \Rightarrow$ 3/8″ slotted hex cap screws (2) $F_3 \sim F_8 \Rightarrow$ 1/4″ slotted hex cap screws (3) $F_9 \sim F_{10} \Rightarrow$ 3/4″ hex nuts.

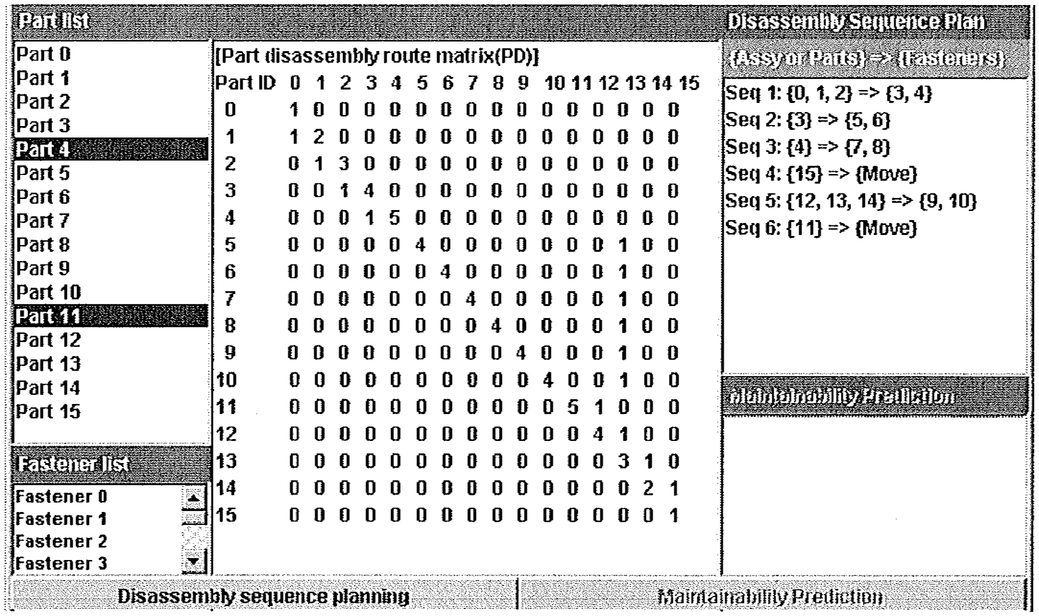


(a) (b)

**Figure 7.3**: A tested example for SD sequence planning (a) 3D model of a magnetic breakerless distributor, and (b) break-down drawing of the tested product.

Figure 7.4 shows the test result via the system developed in this research. As shown in Figure 7.4, the matrix ***PD*** is first defined via the methods developed. For this product, a total of five disassemblability levels are defined in terms of topological part disassemblability. For a randomly selected set ***Ps*** consisting of parts $\{P_4, P_{11}\}$, the family sets $\mathcal{DP}$ and $\mathcal{DF}$ generated in Figure 7.4 are $\{\{P_0, P_1, P_2\}, \{P_3\}, \{P_4\}, \{P_{15}\}, \{P_{12}, P_{13}, P_{14}\}, \{P_{11}\}\}$, and $\{\{F_3, F_4\}, \{F_5, F_6\}, \{F_7, F_8\}, \{\ \}, \{F_9, F_{10}\}, \{\ \}\}$, respectively.

In Figure 7.4, two sequences $\{P_0, P_1, P_2\}$ and $\{P_{12}, P_{13}, P_{14}\}$ respectively represent batch removals of parts. The batch removals are dissembled at one time. This reduces the number of removals in the disassembly of selected parts ***Ps***. In particular, there are two empty subsets in the set $\mathcal{DF}$, which describe no fasteners to remove. Instead of an empty subset in the set $\mathcal{DF}$, the developed system translates it into a $\{Move\}$ statement shown in Figure 7.4, which may be useful for a computer-based graphical simulation.



[Part disassembly route matrix(PD)]

| Part ID | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 1 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

{Assy of Parts} => {Fasteners}

Seq 1: {0, 1, 2} => {3, 4}
Seq 2: {3} => {5, 6}
Seq 3: {4} => {7, 8}
Seq 4: {15} => {Move}
Seq 5: {12, 13, 14} => {9, 10}
Seq 6: {11} => {Move}

**Figure 7.4**: Test result of SD sequence planning for randomly selected parts $\{P_4, P_{11}\}$ from the product shown in Figure 7.3.

### 7.5.2 Sample application 2

As shown in Figure 7.5, a product is here designed to test for SD sequence planning via the developed system. The product consists of 17 parts and 20 fasteners including bolts, screws and nuts. It is assumed that the same types of the fasteners are used in this product, which are 1/4" hex type fasteners. As shown in Figures 7.5(b) and (c), this product has a fastener-based structure that is tightly connected. This makes it hard to obtain a SD sequence with the reduced number of removals. Moreover, it is hard to find a feasible SD sequence for randomly selected parts ***Ps***.

For a set ***Ps*** consisting of parts $\{P_8, P_{10}\}$, the family set $\mathcal{DS}$ and $\mathcal{DF}$ are generated as follows: $\mathcal{DS}=\{\{P_3, P_7\}, \{P_4, P_6\}, \{P_8, P_{10}, P_{11}, P_{12}, P_2\}, \{P_8\}, \{P_{10}\}\}$, and $\mathcal{DF}= \{\{F_7, F_8\}, \{\ \}, \{F_3, F_4\}, \{F_2\}, \{F_{11}\}\}$. In particular, three sub-sequences $\{P_3, P_7\}$, $\{P_4, P_6\}$ and $\{P_8, P_{10}, P_{11}, P_{12}, P_2\}$ represent batch removals of parts. This achieves the direct access to selected parts $\{P_8, P_{10}\}$ with the reduced number of part and fastener removals. Figure 7.6 shows the SD sequence planning and simulation process for this example via the system developed in this research. In Figure 7.6, the architecture and implementation methods of the system are later discussed in Chapter 8.

By Equations (7.3) and (7.4), a quantitative maintainability analysis is executed for the product shown in Figure 7.5. In this test, 14 parts are chosen from the product, and used to form structural significant items (SSIs) and functional significant items (FSIs) for the maintainability analysis. The chosen 14 parts are $P_1$, $P_2$, $P_3$, $P_4$, $P_8$, $P_9$, $P_{10}$, $P_{11}$, $P_{12}$, $P_{13}$, $P_{14}$, $P_{15}$, $P_{16}$ and $P_{17}$. Table 7.2 shows part information used to execute the maintainability analysis in this chapter.

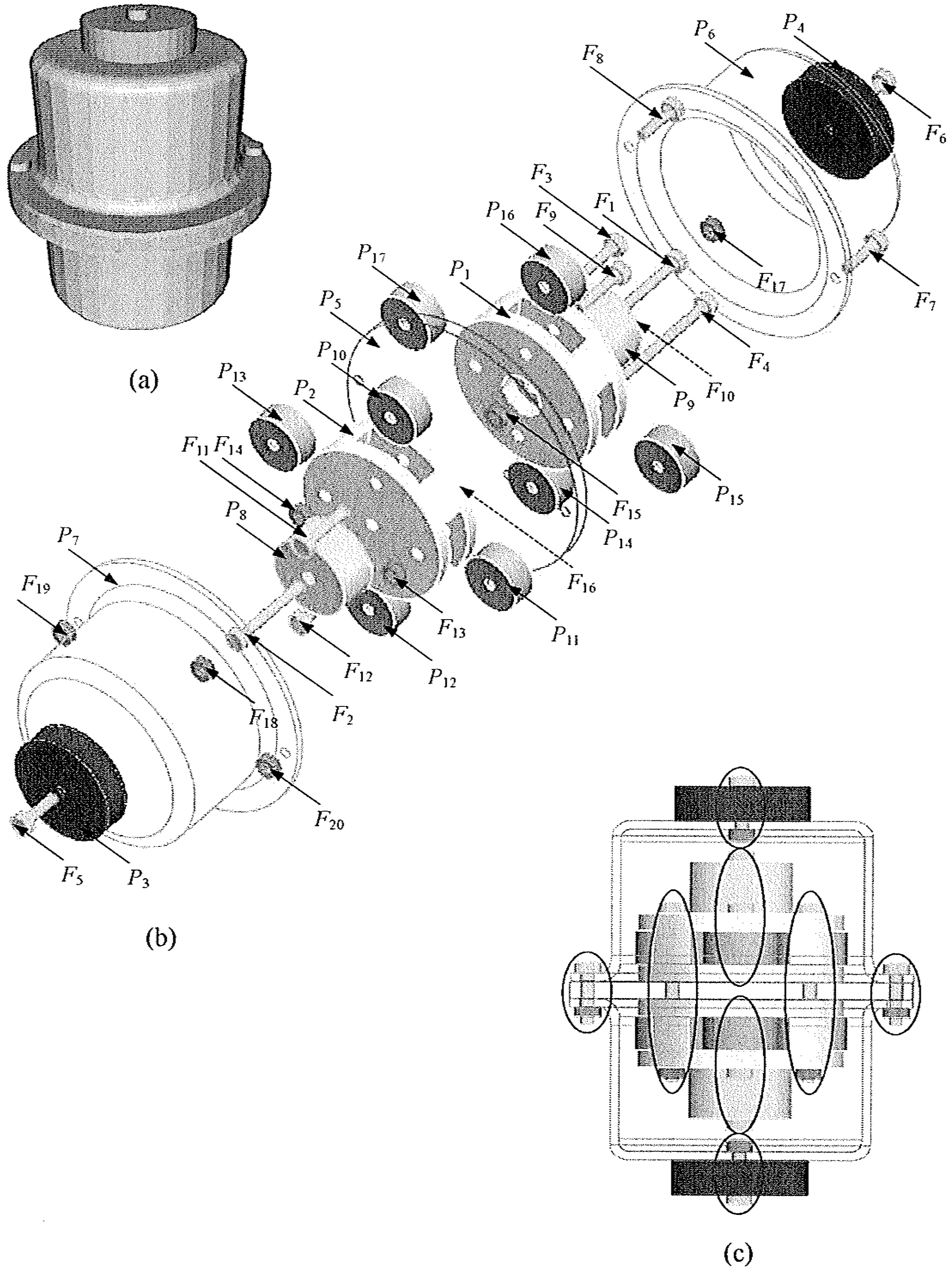
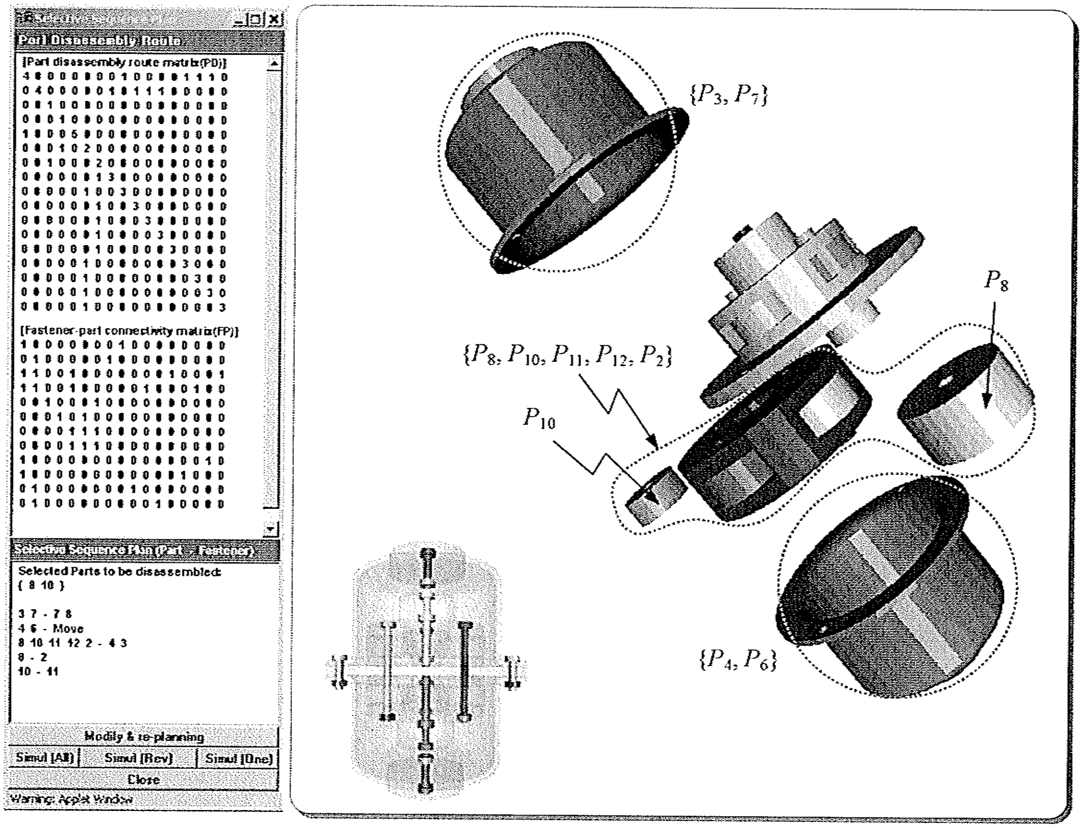


**Figure 7.5**: A tested example for SD sequence planning (a) 3D model of a product (b) break-down drawing of the tested product, and (c) front view of the product.



```
Part Disassembly Route
[Part disassembly route matrix(PD)]
4 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0
0 4 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 3 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 3 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 3 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 3 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 3 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 3 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 3 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 3 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 3 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3

[Fastener-part connectivity matrix(FP)]
1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1
1 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0
0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

Selective Sequence Plan (Part - Fastener)
Selected Parts to be disassembled:
{ 8 10 }

3 7 - 7 8
4 6 - Move
8 10 11 12 2 - 4 3
8 - 2
10 - 11
```

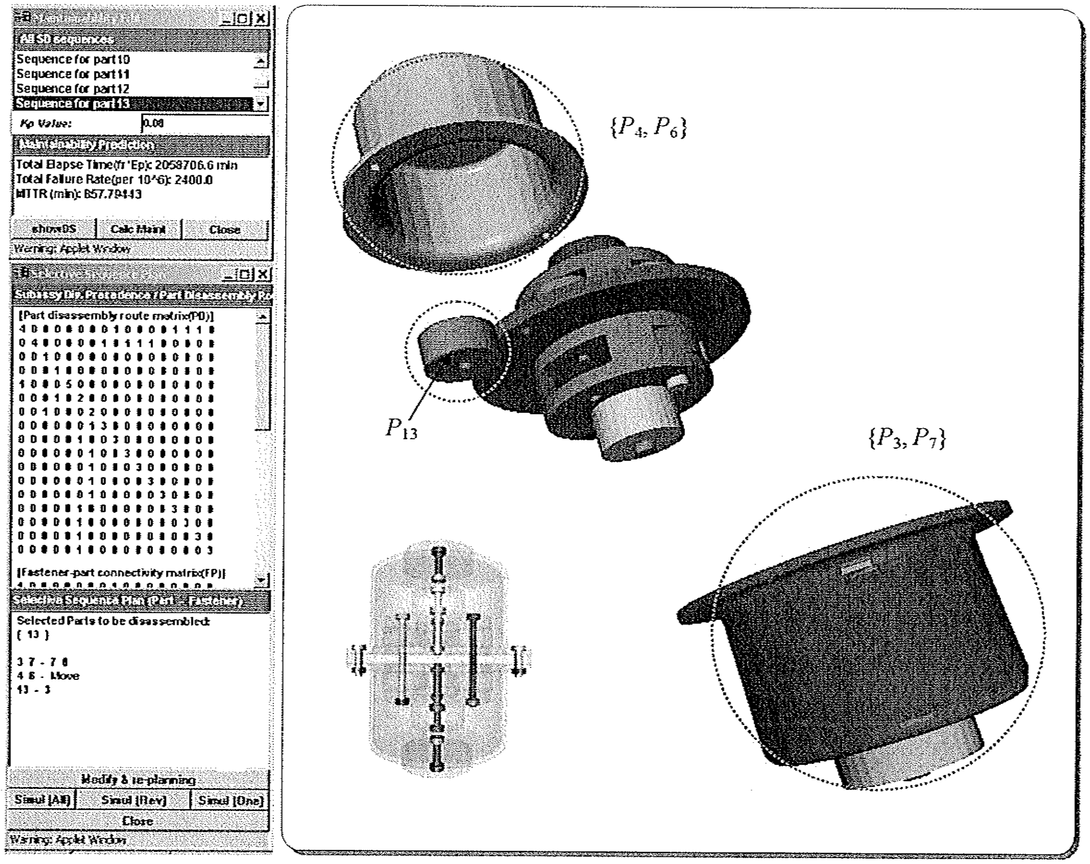
**Figure 7.6**: SD sequence planning and its simulation for the selected parts $\boldsymbol{Ps}=\{P_8, P_{10}\}$.

**Table 7.2**: Part information used to execute the maintainability analysis in this chapter.

| Part ID | Part failure rate ($\times 10^{-6}$) | Part weight (Kgf) |
|---|---|---|
| $P_1$ | 100 | 1.00 |
| $P_2$ | 100 | 1.00 |
| $P_3$ | 200 | 0.20 |
| $P_4$ | 200 | 0.20 |
| $P_8$ | 100 | 0.50 |
| $P_9$ | 100 | 0.50 |
| $P_{10}$ | 200 | 0.15 |
| $P_{11}$ | 200 | 0.15 |
| $P_{12}$ | 200 | 0.15 |
| $P_{13}$ | 200 | 0.15 |
| $P_{14}$ | 200 | 0.15 |
| $P_{15}$ | 200 | 0.15 |
| $P_{16}$ | 200 | 0.15 |
| $P_{17}$ | 200 | 0.15 |

In Table 7.2, the maintenance task frequencies of the chosen SSIs and FSIs are derived from MTBF data. In particular, the distance data of moving a part will be automatically determined by a virtual simulation that requires starting and ending coordinates for a position interpolation while the failure rate and weight for MTTR calculation of a part are retrieved from part DB.

The system shown in Figure 7.7 repeatedly generates 14 $\mathscr{DS}$ based on the task frequencies generated by the failure rates of parts, and calculates the maintainability of the product illustrated in Figure 7.5.



```
All SD sequences
Sequence for part10
Sequence for part11
Sequence for part12
Sequence for part13
Kp Value: 0.08
Maintainability Prediction
Total Elapse Time(fr *Ep): 2058706.6 min
Total Failure Rate(per 10^6): 2400.0
MTTR (min): 657.79443
```

```
Subassy Dir. Precedence / Part Disassembly Ro
[Part disassembly route matrix(PD)]
4 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0
0 4 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 3 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 3 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 3 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 3 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 3 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 3 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 3 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 3 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 3 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3

[Fastener-part connectivity matrix(FP)]
Selective Sequence Plan (Part - Fastener)
Selected Parts to be disassembled:
{ 13 }

3 7 - 7 8
4 8 - Move
13 - 3
```

Figure 7.7: Maintainability analysis and simulation for the selected SSIs or FSIs.

As shown in Figure 7.7, the total MTTR for the example product is 857.79 min. This value relies on the configuration of the product so that it will be minimized by design changes. In addition, the $\mathscr{DS}$ generated for SSIs or FSIs can be individually modified and simulated via the selective-disassembly sequence edit window. Figure 7.7 also shows the simulation for the disassembly of $P_{13}$, which is based on the family sets $\mathscr{DS}_{13}$ and $\mathscr{DF}_{13}$, $\{\{P_3, P_7\}, \{P_4, P_6\}, \{P_{13}\}\}$, and $\{\{F_7, F_8\}, \{\ \}, \{F_3\}\}$, respectively.

## 7.6 Chapter Summary

This chapter presented an approach to non-destructive SD sequence planning in a dynamic DM environment. It was based on the assembly-tool feasibility analysis and the part disassemblability analysis, which are presented in Chapters 4 and 5, respectively.

Based on the matrix ***PA***, the matrix ***PD*** was constructed to form a nearly optimal disassembly chain of a part being disassembled. The disassembly chain was also used to define a module to directly access selected parts with a reduced number of part removals.

A matrix-based heuristic approach was developed for SD sequence planning in this chapter. This is based on the matrix ***PD*** and the assembly-tool feasibility analysis. As a robust and practical approach, this quickly generates a near optimal SD sequence for a given ***Ps***. Several experiments were executed to demonstrate the performance of the developed approach in this chapter.

Based on the approach developed in this chapter, it is possible to achieve dynamic disassembly planning in a Web-based PD&M environment. A feasible SD sequencing plan can be generated in the networked collaborative environment. The generated SD plan can be used for the early involvement of DM enterprises in a temporary enterprise alignment that is seeking manufacturing opportunities for a specific product. It can be used for realizing various disciplines including maintenance, product recycling and disposal at the service and product end-life levels. In addition, the SD plan can provide VP or VM systems with a reduced number of feasible *what-if* scenarios regarding disassembly, and serve at supporting DFD, DFMa, and DFR to execute the practical assessment of de-manufacturability, and the reliable estimation of disassembly time and cost.

With respect to the research on the dynamic disassembly planning, published papers in international journals and conferences are as follows.

**Chung, C.** and Peng, Q. (2005). A hybrid approach to selective-disassembly sequence planning for de-manufacturing and its implementation on the Internet. *Special issue on intelligent disassembly in the de-manufacturing process*, International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-005-0038-5 (First Online).

**Chung, C.** and Peng, Q. (2004). An integrated approach to selective-disassembly sequence planning. Proceedings of the International Conference on Flexible Automation & Intelligent Manufacturing, Toronto, Canada, FAIM 2004, July 12-14, pp. 262-269.

**Chung, C.** and Peng, Q. (2004). An algorithmic approach to quantitative maintainability analysis in product design. Proceedings of the International Conference on Computers and Industrial Engineering, Cheju, South Korea, C&IE 2004, March 25-27, pp. 1-6.

# Chapter 8

# Implementations on the Web

This chapter presents implementations of the developed approaches for dynamic task planning on a three-tier Internet environment. The Web-based systems are developed via VRML-EAI, Java and JDBC. A Web server and a DB server to form a three-tier Internet environment adopt Apache HTTP Server Version 1.3, and MySQL DB server version 3.23, respectively.

## 8.1 Geometry Data Representation

VRML is a language intended to create a 3D virtual world on a Web. VRML was announced by Berners and Reggett at the first annual World Wide Web (WWW) Conference in Geneva, Switzerland in 1994. The current version of VRML is VRML97 (ISO/IEC 14772-1: 1997), which became an international standard in the same year. VRML is designed to be platform independent. Web users need only have a browser and a VRML plug-in to view the VR world, just like viewing Web pages written in HTML format. VRML files are smaller in size than most other 3D file formats, they can be transferred easily through the Internet (Xu *et al.*, 2003). Therefore, VRML is a powerful tool to create solid models on the Web easily.

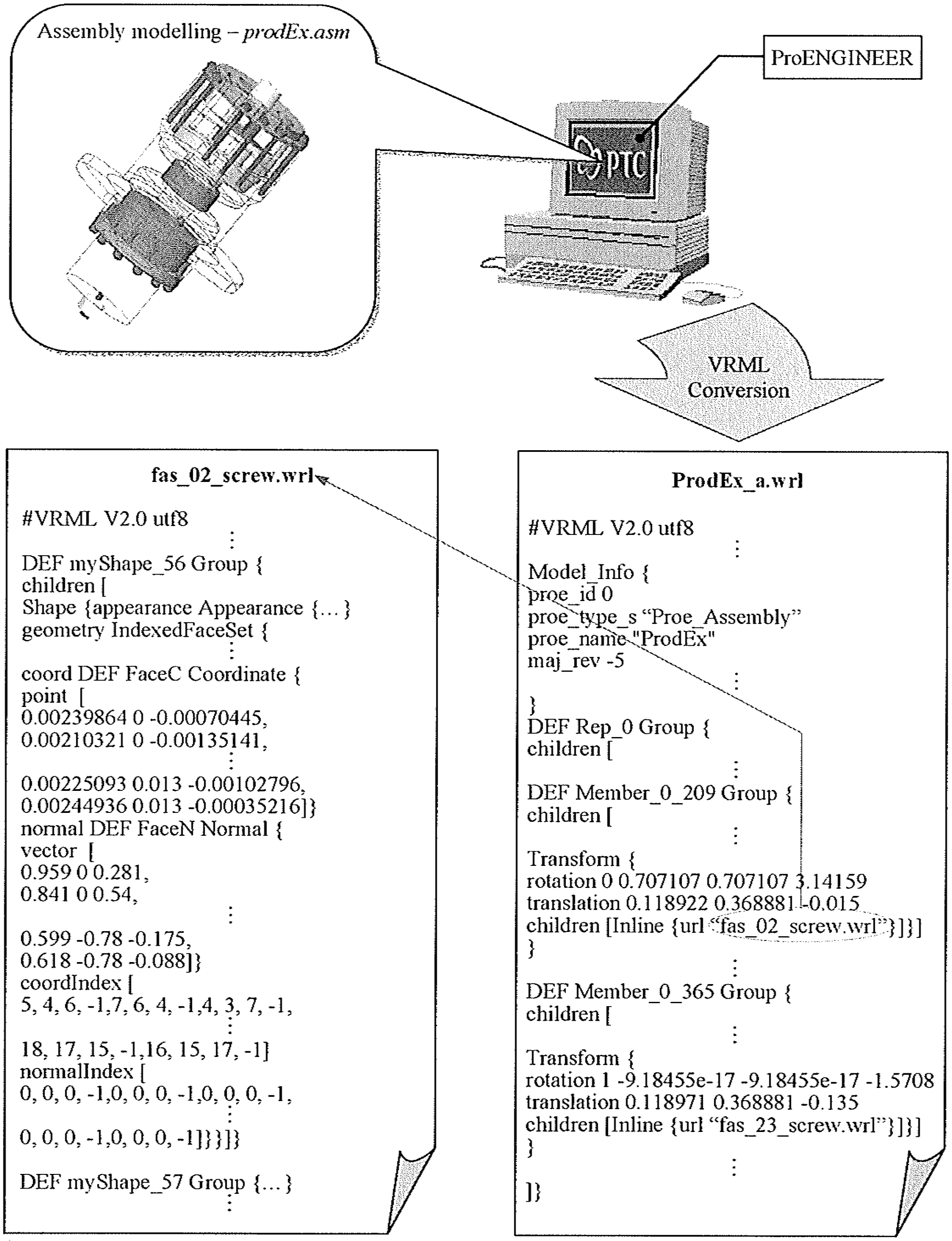
In this research, the geometry data of a component (*i.e.*, a part or fastener) are represented by a VRML node named *IndexedFaceSet*. The following are the data format of the *IndexedFaceSet* VRML node.

```
IndexedFaceSet {
        eventIn         MFInt32   set_colorIndex
        eventIn         MFInt32   set_coordIndex
        eventIn         MFInt32   set_normalIndex
        eventIn         MFInt32   set_texCoordIndex
        exposedField    SFNode    color               NULL
        exposedField    SFNode    coord               NULL
        exposedField    SFNode    normal              NULL
        exposedField    SFNode    texCoord            NULL
        field           SFBool    ccw                 TRUE
        field           MFInt32   colorIndex[]                  # [-1,∝)
        field           SFBool    colorPerVertex      TRUE
        field           SFBool    convex              TRUE
        field           MFInt32   coordIndex[]                  # [-1, ∝)
        field           SFFloat   creaseAngle         0         # [0, ∝)
        field           MFInt32   normalIndex[]                 # [-1, ∝)
        field           SFBool    normalPerVertex     TRUE
        field           SFBool    solid               TRUE
        field           MFInt32   texCoordIndex[]               # [-1, ∝)
}
```

The above representation is automatically allocated when a component solid model is converted by the VRML translator in any commercial CAD system. To represent a component in VRML, the *IndexedFaceSet* node defines a set of triangle patches based on a set of 3D point coordinates (*i.e.*, *coord*), a set of normal vectors (*i.e.*, *normal*), and sets of indexes (*i.e.*, *normalIndex[]* and *coordIndex[]*). An index array in *coordIndex[]* or *normalIndex[]* consists of a set of three elements to define a triangle patch or its orientation. The elements index the positions of points ($x$, $y$, $z$) in *coord*, or the positions of vectors $\left(x\hat{\boldsymbol{i}} + y\hat{\boldsymbol{j}} + z\hat{\boldsymbol{k}}\right)$ in *normal*.

In this research, ProENGINEER, a widely used commercial CAD system, is used to model products, and their VRML data are converted via its VRML translator. In particular, a product in ProENGINEER is normally modelled as an assembly consisting of parts and fasteners. When this assembly is converted into VRML models, ProENGINEER creates a master VRML file (*i.e.*, *_a.wrl*) and its components VRML files (*i.e.*, *.*wrl*). Figure 8.1 shows the VRML conversion process of a product model via the VRML translator employed in ProENGINEER. As shown in Figure 8.1, the master VRML file named *ProdEx_a.wrl* lists all the components of the product, which are sets of parts and fasteners. This also has transformation matrices including 3D rotation and translation for each component.

A DB importer is implemented to collect and store essential information of a component represented in VRML. This reduces the data size, and forms organized and structured data. The DB importer has the following processes: (1) searching a component name and its transformation matrix in the master VRML file; (2) opening the found component VRML file; (3) taking 3D point coordinates, normal vectors, coordinate indexes, and normal vector indexes; (4) transforming the 3D point coordinates via the found transformation matrices and general 3D transformation equations; (5) rearranging the coordinate indexes in the right hand rule via normal vectors and indexes; (6) creating a dynamic SQL query to insert the defined data into the product DB; (7) connecting and transacting to the product DB; and (8) repeating these processes for the next component in the master VRML file.



**fas_02_screw.wrl**

```
#VRML V2.0 utf8
                    ⋮
DEF myShape_56 Group {
children [
Shape {appearance Appearance {...}
geometry IndexedFaceSet {
                    ⋮
coord DEF FaceC Coordinate {
point [
0.00239864 0 -0.00070445,
0.00210321 0 -0.00135141,
                    ⋮
0.00225093 0.013 -0.00102796,
0.00244936 0.013 -0.00035216]}
normal DEF FaceN Normal {
vector [
0.959 0 0.281,
0.841 0 0.54,
                    ⋮
0.599 -0.78 -0.175,
0.618 -0.78 -0.088]}
coordIndex [
5, 4, 6, -1,7, 6, 4, -1,4, 3, 7, -1,
                    ⋮
18, 17, 15, -1,16, 15, 17, -1]
normalIndex [
0, 0, 0, -1,0, 0, 0, -1,0, 0, 0, -1,
                    ⋮
0, 0, 0, -1,0, 0, 0, -1]}}]}

DEF myShape_57 Group {...}
                    ⋮
```

**ProdEx_a.wrl**

```
#VRML V2.0 utf8
                         ⋮
Model_Info {
proe_id 0
proe_type_s "Proe_Assembly"
proe_name "ProdEx"
maj_rev -5
                         ⋮
}
DEF Rep_0 Group {
children [
                         ⋮
DEF Member_0_209 Group {
children [
                         ⋮
Transform {
rotation 0 0.707107 0.707107 3.14159
translation 0.118922 0.368881 -0.015
children [Inline {url "fas_02_screw.wrl"}]}]
}
                         ⋮
DEF Member_0_365 Group {
children [
                         ⋮
Transform {
rotation 1 -9.18455e-17 -9.18455e-17 -1.5708
translation 0.118971 0.368881 -0.135
children [Inline {url "fas_23_screw.wrl"}]}]
}
                         ⋮
]}
```

**Figure 8.1**: VRML conversion process of a product model via the VRML translator in ProENGINEER.

To store product information in VRML, a product DB is defined in this research. The product DB shown in Figure 8.2 consists of five tables and relationships. The tables are designed to store hierarchical product information including system, product, part and fastener. This product DB resides in a DB server that is later discussed in this chapter.

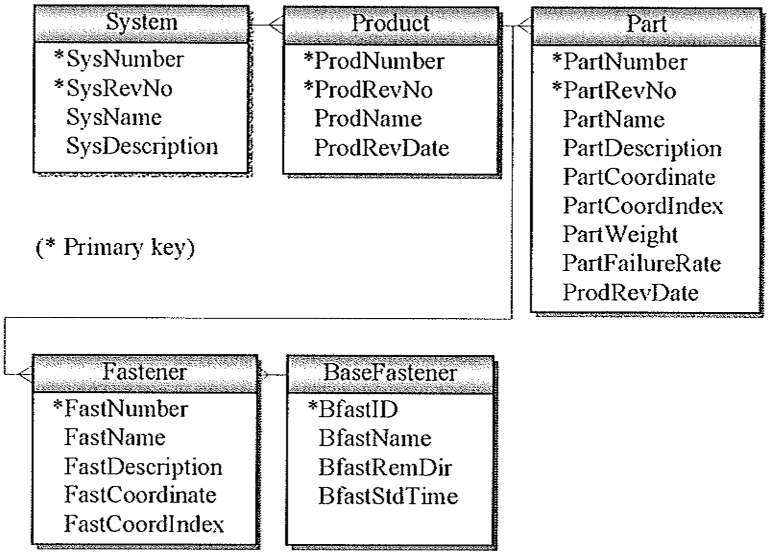


**Figure 8.2**: Relational DB model of product DB used in this research.

The stored component information in Figure 8.2 is used in the task planning presented in the previous chapters. A Java class named *VRMLGeometryData* (Refer to Appendix A.15) is efficiently defined to store the information being retrieved from the product DB. In Appendix A.15, the *VRMLGeometryData* class contains various properties and methods. For a component (*i.e.*, part or fastener), VRML geometry, OBBs or BSs are generated and temporarily stored by these methods and properties. The defined OBBs and BSs are also used for collision detection in determining the primitive matrices presented in Chapter 5.

The following details describe the Java program code of a method defined in Appendix A.15. This method named *getVRMLCode()* in Appendix A.15 is used to dynamically create the VRML geometry of a component via coordinates and indexes retrieved from the product DB.

```
public String getVRMLCode () {
        int i;
        String rs="Shape {\n" +
                "geometry IndexedFaceSet {solid FALSE\n" +
                "coord Coordinate {point [\n";
                for (i=0;i<nCoord;i++) rs=rs+coord[i][0]+" "+coord[i][1]+" "+coord[i][2]+",\n";
                rs=rs+"]}\n";
                rs=rs+"coordIndex [\n";
                for (i=0;i<nCoordIndex;i++)
                rs=rs+coordIndex[i][0]+" "+coordIndex[i][1]+" "+coordIndex[i][2]+" -1,\n";
                rs=rs+"]}\n";
                rs=rs+"appearance Appearance {\n" +
                "material DEF m" +name+ " ";
                rs=rs+super.getAppearance();
                rs=rs+"}";
                return rs;
}
```

In particular, the above program contains *material DEF m**. This makes a component automatically contain the material VRML node when its VRML geometry is defined. This material VRML node is later used to dynamically manipulate material properties such as *diffuseColor* (main shading color), *emissiveColor* (glowing color) and *transparency* (opaque or not) in a VRML object. Such functions as hiding, transparency, highlight and color interpolation are also efficiently implemented by manipulating these properties via Java programs.

## 8.2 Overall System Architecture

As shown in Figure 8.3, a three-tier architecture on the Internet is used to implement Web-based applications for the dynamic task planning in this research. This architecture consists of a Web server, a DB server and clients. The features of these elements are described in the following sections.

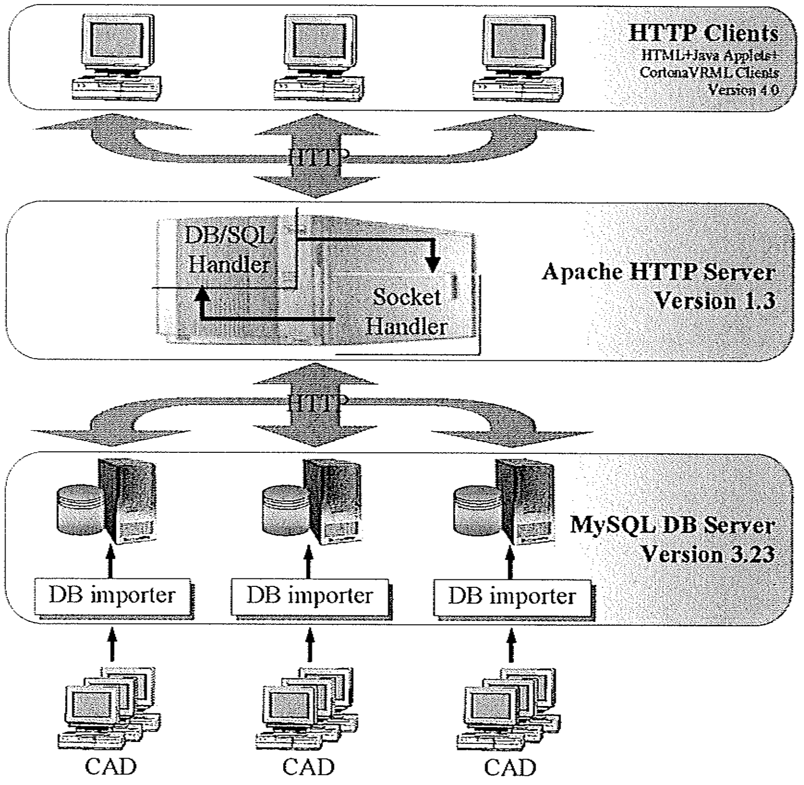


**Figure 8.3**: Three-tier architecture on the Internet used in this research.

## 8.2.1 HTTP Web server

The Web server used in Figure 8.3 adopts Apache HTTP server version 1.3. The Apache HTTP server is the free, open source Web server software developed for Unix, Linux, Windows and other operating systems. This server is widely used all over the world, and more than 60% of Web sites on the Internet use it (Apache, 2005). The Apache HTTP Server serves static and dynamic information including HTML and other multimedia. This information is also transmitted via various standard protocols such as HTTP, common gateway interface (CGI), hypertext transmission protocol, secure (HTTPS) and multipurpose Internet mail extension (MIME). For a user request for a file through a Web browser, this server has the following process steps: (1) accept network connections from the browser; (2) retrieve content from disk; (3) run local CGI programs or application server programs for such common application server technologies as ASP.NET, hypertext pre-processor (PHP), and Java server page (JSP)/Java Servlets; (4) transmit data back to clients; and (5) keep a log of user activity.

In Figure 8.3, server-side programs developed are running on the HTTP Web server. The server-side programs in this research are named as *DB/SQL handler* with JDBC and *socket handler*. Implemented via Java 1.1.8, these programs serve to respectively communicate with the remote DB server and clients over the Internet. Details are later discussed in this chapter.

### 8.2.2 Remote DB server

As shown in Figure 8.3, MySQL server version 3.23, an RDBMS, is used to build the remote DB server in this research. Like the popular Linux operating system, MySQL is released as open source software under the GNU public license (GPL). This MySQL supports most of the functionality in a commercial RDBMS. It ensures that transactions allow the building of indexes, support standard data types, and allow for DB replication, among other features.

Moreover, the MySQL provides standards-based drivers for JDBC, open database connectivity (ODBC), and .Net enabling developers to build DB applications in their languages. In particular, MySQL Connector/J 2.0.14 is used as a JDBC driver in this architecture. The MySQL Connector/J is a native Java driver that converts JDBC calls into the network protocol used by the MySQL DB. It allows developers working with the Java programming language to easily build programs and applets that interact with MySQL and connect all corporate data, even in a heterogeneous environment.

In addition, a DB importer is implemented on the DB server shown in Figure 8.3. This aims to efficiently store VRML data from a CAD system into the product DB that requires organized and structured information.

### 8.2.3 HTTP clients

A Web browser containing a HTML file is an HTTP client because it sends requests to an HTTP Web server that then sends responses back to the client. VRML plug-ins and Java applets are embedded in the HTML file. Thus, the Java applets interact with the 3D objects in the VRML browser. As shown in Figure 8.3, Cortona VRML client version 4.0 is adopted as the VRML plug-in used in this architecture. This is a fast and highly interactive Web3D viewer that is ideal for viewing 3D models on the Web, and serves various Web browsers (Internet Explorer, Netscape Navigator, Mozilla, *etc.*) and office applications (Microsoft PowerPoint, Microsoft Word, *etc.*). In particular, the Cortona VRML plug-in completely supports the VRML 97 specification (ISO/IEC 14772-1: 1997), Java, JavaScript and extended VRML-EAI (ISO/IEC 14772-2: 2004).

## 8.3 Socket-based Communication

In this research, Java socket is used to communicate between a server and clients. This Java socket-based communication is a native Internet communication method, and is similar to performing file input/output (I/O). This communication is programming-language independent so that a socket program written in Java also communicates to a program written in non-Java socket class. Figure 8.4 shows the concept of socket-based communication between a multithreaded server and clients in this research.

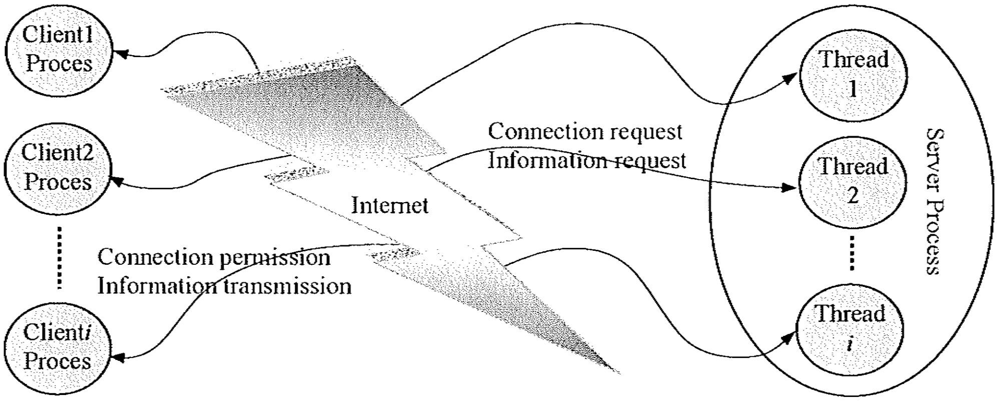


**Figure 8.4**: Concept of socket-based communication between a multithreaded server and clients.

First, the Apache HTTP Web server program runs on a server computer. This server has a socket that is bound to a specific port. Then, a developed server program named *NetServer* is executed to wait and listen to the socket for a client that makes a connection request. *NetServer* Java class is developed by Java 1.1.8 in this research, and its structure is as follows:

```
import java.io.*;
import java.net.*;
public class NetServer extends Thread{
        public NetServer() {super();}
        public void run() {
                try {
                        ServerSocket serverSocket = new ServerSocket(8765); /* Port */
                        while (true) {
                                Socket clientSocket = serverSocket.accept();
                                (new Thread(new SockHandler(clientSocket))).start();
                        }
                }catch(IOException ex) {System.err.println(ex);}
        }
}
```

In the above program, *8765* is the port number used for the Apache HTTP Web server. In particular, a multithreaded server is achieved by the aid of a Java class, *Thread* class. If a connection is requested by a client, the server accepts the connection. Upon acceptance, the server gets a new socket bound to a different port. Thus, the new socket can continue to listen to the original socket for connection requests while serving the connected client.

In this research, the following socket program is developed for a client that makes requests for connection and information. In particular, the information request initiates the Web-server to connect to a remote DB server.

```
import java.io.*;
import java.net.*;
public class ClientSocketProgram {
        Socket socket;
        BufferedReader fromServer = null;
        PrintWriter toServer = null;
        SocketProgram () {
                try {
                        socket = new Socket("130.179.132.34",8765); /* Server IP, port */
                } catch (UnknownHostException e) {
                        System.out.println("Unknown host");
                } catch (IOException e) {
                        System.out.println("IO Exception");
                        return;
                }
        }
        public String[] getProductInformationFromDB (String arg) {...}
        public String[] getPartInformationFromDB (String arg) {...}
                                    ⋮
}
```

The streams used in file I/O operation are also applicable to socket-based I/O. Therefore, the developed public methods including *getPartInformationFromDB* have the following procedure to get information via socket-based I/O operation.

```
try {
        fromServer = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));
        toServer = new PrintWriter(socket.getOutputStream(), true);
        toServer.println(arg);
        String tmpArg=fromServer.readLine();
        /* Data processing and creating: String[] dataDB[] */
                                          ⋮
        return dataDB;
}
```

In the above program, the *arg* String variable stores information that is used to execute a method in the server, and to get results in the String data type. Once the information is formed in a client, it is sent to the server. In this research, a command protocol is defined for the information to efficiently communicate between a client and the server. Figure 8.5 shows this command protocol defined.

| | | | | | |
|---|---|---|---|---|---|
| Method ID | IP address of DB server | DB name | Number of parameters | Parameter 1 | … |

**Figure 8.5**: Defined command protocol for socket communication between a client and the server.

As shown in Figure 8.5, *method ID* specifies a method to be invoked and executed in the server. Internet protocol (*IP*) *address of a DB server* is included in this protocol for designating a DB server in multi-DB server environments. For instance, a Web-based PD&M environment involves many dispersed sub-contractors or business partners with their own DB servers in developing a new product. By *DB name* in Figure 8.5, a specific RDB including various tables and relationships is subsequently selected in the designated DB server. Moreover, parameters in Figure 8.5 are used to form a dynamic SQL query to get intended results in transacting with the RDB.

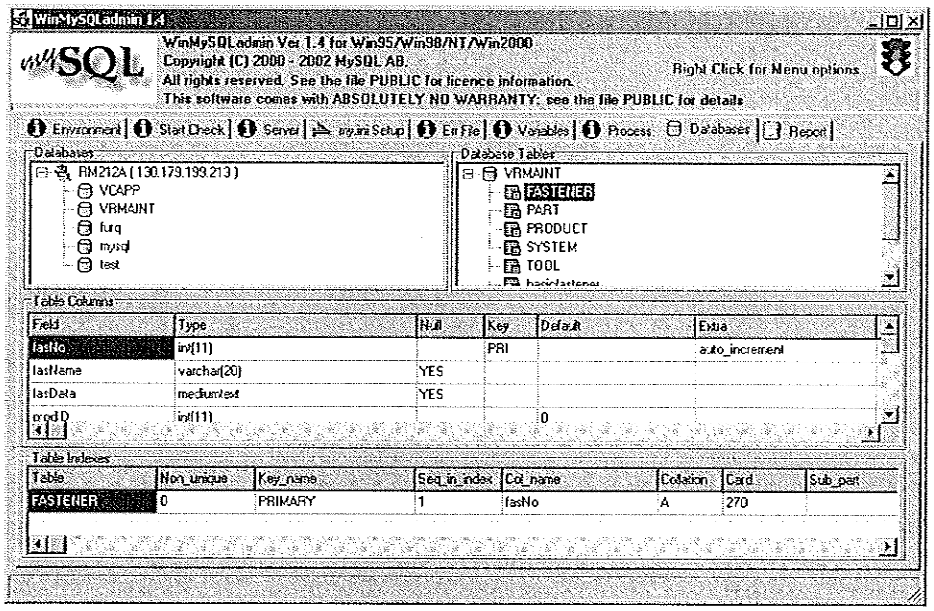
In the server side, a threaded *SocketHandler* runs to process requests from a client. This program interfaces with a Java interface named *Runnable* interface. This defines the behaviour required to execute as a separate thread in a multithreaded program. With the aid of this interface, the *NetServer* program previously presented can be executed for other clients at the same time. The *SocketHandler* class developed in this research is as follows.

```
import java.io.*;
import java.net.*;
import java.util.*;
public class SocketHandler implements Runnable{
        private String[] result;
        private Socket clientSocket = null;
        public SocketHandler (Socket con) {this.clientSocket = con;}
        public void run() {
                try {
                        BufferedReader fromClient = new BufferedReader(
                                new InputStreamReader(clientSocket.getInputStream()));
                        PrintWriter toClient = new PrintWriter(
                                clientSocket.getOutputStream(), true);
                        while (true) {
                                String mesg = new String (fromClient.readLine());
                                int methodID=getMethodID (mesg);
                                String ipAddress=getDBServerIPAddress (mesg);
                                String dbName=getDBName (mesg);
                                String[] param = getParamether (mesg);
                                DBSqlHandler sql = new DBSqlHandler ();
                                sql.open(ipAddress, dbName);
                                if (methodID == 1) {
                                        result=sql.getDataFromDBForOrder1 (param);
                                        for (int i=0;i<=Integer.parseInt(result[0]);i++)
                                                toClient.println(result[i]);
                                }
                                if (methodID == 2) {...}
                                if (methodID == 3) {...}
                                                ⋮
                                sql.close();
                        }
                } catch(IOException ex) {System.err.println(ex);}
        }
        private int getMethodID (String mesg) {...}
        private String getDBServerIPAddress (String mesg) {...}
                                                ⋮
}
```

The above *SocketHandler* program takes information from a received command protocol, executes a corresponding method in the *DBSqlHandler* program, and subsequently sends out the results to the client socket program. The *DBSqlHandler* developed in this research is discussed in the next section.

## 8.4 Java Database Connectivity

The architecture used in this research aims to construct a collaborative and distributed virtual environment with a three-tier Internet architecture. An RDBMS resides on a tier of the multi-tier architecture, which is farthest removed from the end users. JDBC techniques aid clients to connect to a remote DB server through the Internet. MySQL Connector/J 2.0.14 is used as a JDBC driver to interact with the MySQL DB server as shown in Figure 8.6. This driver is also efficiently manipulated by all basic Java interfaces and classes, which are defined in a Java package named *Java.sql*.



WinMySQLadmin Ver 1.4 for Win95/Win98/NT/Win2000
Copyright (C) 2000 - 2002 MySQL AB.
All rights reserved. See the file PUBLIC for licence information.
This software comes with ABSOLUTELY NO WARRANTY: see the file PUBLIC for details

Right Click for Menu options

RM212A ( 130.179.199.213 )
- VCAPP
- VRMAINT
- furq
- mysql
- test

VRMAINT
- FASTENER
- PART
- PRODUCT
- SYSTEM
- TOOL
- basicfastener

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| fasNo | int(11) | | PRI | | auto_increment |
| fasName | varchar(20) | YES | | | |
| fasData | mediumtext | YES | | | |
| prodID | int(11) | | | 0 | |

| Table | Non_unique | Key_name | Seq_in_index | Col_name | Collation | Card | Sub_part |
|---|---|---|---|---|---|---|---|
| FASTENER | 0 | PRIMARY | 1 | fasNo | A | 270 | |

**Figure 8.6**: Administration window of MySQL DB server version 3.23 used.

The following are the major structure of *DBSqlHandler* program developed.

```
import java.sql.*;
import java.util.*;
public class DBSqlHandler {
        private Connection con;
        DBSqlHandler () {
                try {
                        Class.forName("org.gjt.mm.mysql.Driver");
                } catch ( ClassNotFoundException ee) {ee.printStackTrace();}
        }
        public void close() {
                try{
                        con.close();
                }catch (SQLException e) {e.printStackTrace();}
        }
        public void open(String ipAddress, String dbName) {
                try {
                        String url = "jdbc:mysql://"+ipAddress+":3306/"+ dbName;
                        con = DriverManager.getConnection(url, "userID", "password");
                } catch (SQLException e) {e.printStackTrace();}
        }
        public String[] getDataFromDBForOrder1(String[] param) {...}
        public String[] getDataFromDBForOrder2(String[] param) {...}
                                                  :
        private ResultSet selectRecordSet (String query) {...}
        private void insertRecord (String query) {...}
        private void deleteRecord (String query) {...}
                                               :
}
```

In the above program, the MySQL DB server allocates the 3306 port for communication. Once connectivity is established via a DB server IP address and the DB name, public methods including *getDataFromDBForOrder1* are invoked by the *SocketHandler* program previously presented. These public methods generate dynamic SQL queries with the transferred parameters. The generated queries are sent to provide methods based on DML, and subsequently used to retrieve intended information from the designated DB. The provided methods in the program have the following core procedure to interact with the designated DB.

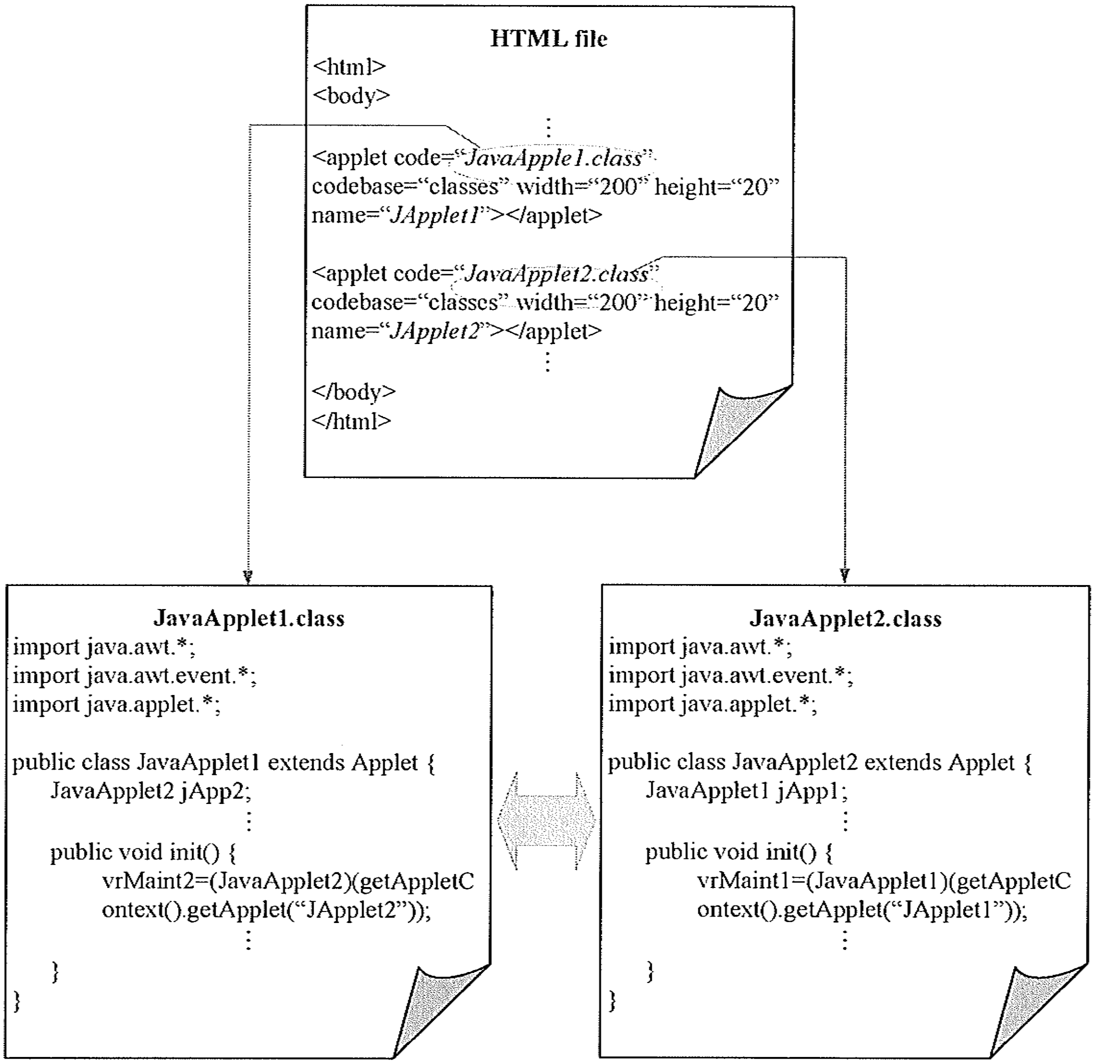
```
try {
        Statement stmt = con.createStatement();
        ResultSet result = stmt.executeQuery(query);
} catch (SQLException e) {e.printStackTrace();}
```

# 8.5 Java Applet and VRML-EAI

On the client side, an HTML file transmitted from the server is running on a Web browser. As a client uses the application, the HTML file embeds several Java applets and VRML browsers serving as VRML plug-ins. A Java applet in this research communicates with the Web server via the Java socket-based communication presented in Section 8.3. This also communicates with other Java applets in the same HTML file, and interacts with the 3D objects in the VRML browsers. Details are described in the following sections.

## 8.5.1 Communication between Java applets

In this research, a client application consists of several Java applets based on their functionality. This approach is advantageous in enhancing the usability of objects built on the OOP concept. Figure 8.7 shows the concept for implementing communication between two applets in a client application. As shown in Figure 8.7, two Java applets with different functionality are embedded in an HTML file, namely *JavaApplet1.class* and *JavaApplet2.class*. To make communication between these two applets, their names are additionally designated in the HTML file with an HTML tag, *<applet>*. Such applet names as *jApplet1* and *jApplet2* in Figure 8.7 are designated for *JavaApplet1.class* and *JavaApplet2.class*, respectively. These names are used by a Java method, *getAppletContext()* that allows an applet to be instantiated in another applet in an HTML file. A communication is built by this method so that an applet can use the methods and properties of another applet, and vice versa.



```
<html>
<body>
        ⋮
<applet code="JavaApple1.class"
codebase="classes" width="200" height="20"
name="JApplet1"></applet>

<applet code="JavaApplet2.class"
codebase="classes" width="200" height="20"
name="JApplet2"></applet>
        ⋮
</body>
</html>
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class JavaApplet1 extends Applet {
    JavaApplet2 jApp2;
        ⋮
    public void init() {
        vrMaint2=(JavaApplet2)(getAppletC
        ontext().getApplet("JApplet2"));
        ⋮
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class JavaApplet2 extends Applet {
    JavaApplet1 jApp1;
        ⋮
    public void init() {
        vrMaint1=(JavaApplet1)(getAppletC
        ontext().getApplet("JApplet1"));
        ⋮
    }
}
```

**Figure 8.7**: Implementation of communication between two applets in a Web-based client application.

In this research, developed applets include server selector, message windows, VRML model handler, and other applets developed for dynamic task planning. These applets are formed as objects or components so that a Web-based client application can be easily implemented via the *plug and play* concept.

### 8.5.2 Communication between Java applet and VRML-EAI

The EAI used in this research is a programming interface for communication between VRML and Java applets. This is based on an ISO standard, VRML 97 international standard part 2 published in 2004. The EAI specification has developed through the following three steps (Lee, 2004): (1) previous *de facto* EAI (Web3D Org., 2005) in 1997; (2) EAI draft standard in 1997; and (3) VRML 97 international standard part 2 (ISO/IEC 14772-2: 2004) in 2004.

The first two EAI specifications use an interface package named *External Package*. The *External Package* is supported by many VRML plug-ins such as ParallelGraphics Cortona (ParallelGraphics Co., 2005), Blaxxun Contact (Blaxxun Co., 2005), Nexternet Pivoron Player (Lee, 2004) and Cosmo Player (Lee, 2004). This is also supported by such Web browsers as Internet Explorer and Netscape. The final version of EAI released in 2004 uses an interface package named *EAI Package* that is extended from *External Package*. In particular, only ParallelGraphics Cortona VRML plug-in and Internet Explorer currently support this extended interface (Lee, 2004). This is partly because the specification has recently been approved and recommended as an international standard (ISO, 2005).

Figure 8.8 shows the main notion of implementing communication between a Java applet and a VRML plug-in in a Web-based client application. With the EAI, it is possible for an applet method in a standard Web browser to access the scene graph of an embedded

VRML plug-in. As shown in Figure 8.8, the EAI provides two important services: (1) enabling external programs to read and change the scene graph, and (2) enabling external programs to register some of their functions as callbacks (Diehl, 2001). Every callback is bound to an event. Whenever this event is generated in the VRML scene, the browser invokes the associated callback function and passes the current value of the event as an argument. To provide a Java applet with the above two services, a Java program is developed in this research.

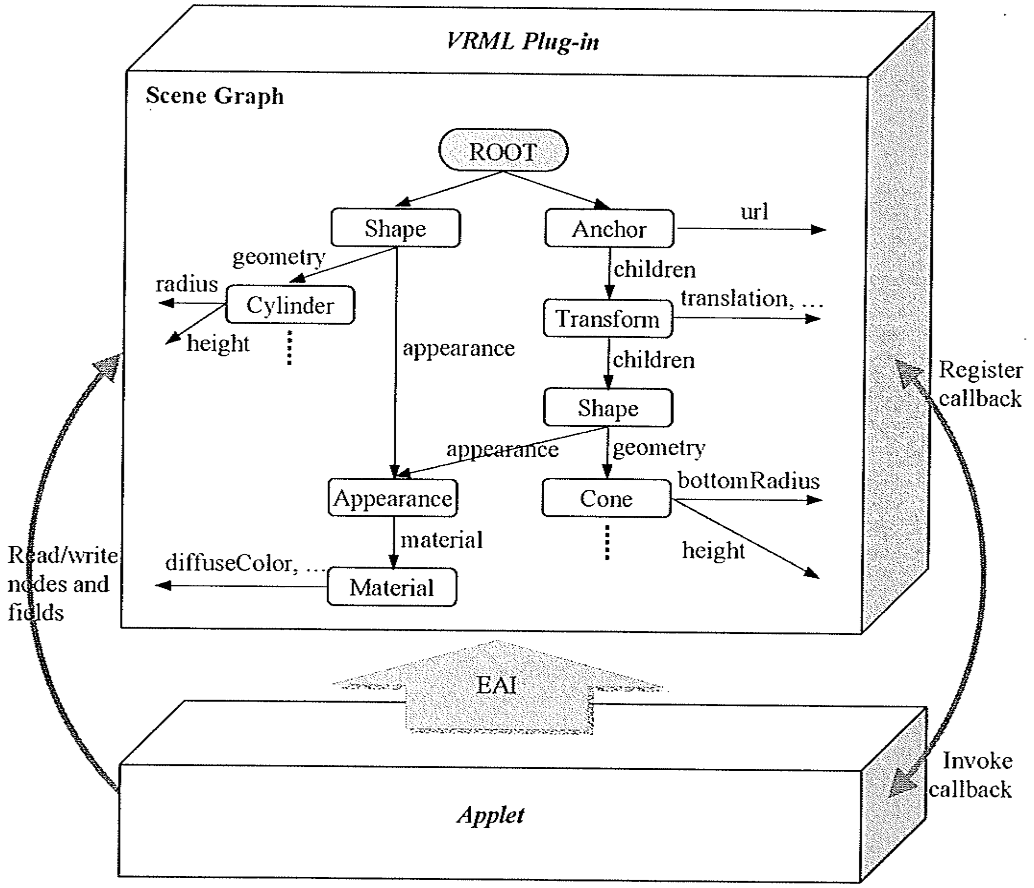


**Figure 8.8**: Implementation of communication between a Java applet and a VRML plug-in in a Web-based client application.

The following are a part of the developed program, and describe declarations.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import vrml.eai.*;
import vrml.eai.event.*;
import vrml.eai.field.*;
public class VR_EAI_Applet1 extends Applet implements VrmlEventListener{
        Browser browser0;
                                        ⋮
        private EventOutSFBool [] a_TouchSens_changed=new EventOutSFBool
                [nTotalNode]; /* Total number of nodes consisting of parts and fasteners */
        private EventOutSFVec3f[] a_translation_changed=new EventOutSFVec3f
                [nTotalNode];
                                        ⋮
}
```

The above program first imports a Java package named *vrml.eai* that is provided by the Cortona VRML plug-in based on ISO/IEC 14772-2: 2004. This package contains the classes and sub packages of the EAI. Then, an EAI interface named *VrmlEventListener* is implemented in the above applet. This allows a VRML browser to invoke a callback occurring at a VRML object, and subsequently, the applet receives an event message from the VRML browser. Inside the program above, a VRML browser is declared with EAI variables including *EventOutSFBool* and *EventOutSFVec3f*. In particular, these variables are assigned to parts and fasteners in a retrieved product from a DB, and used to get event messages from the VRML browser. If an object in the VRML browser is touched by a user, for instance, the value of its *EventOutSFBool* variable changes and this result is signalled to the applet.

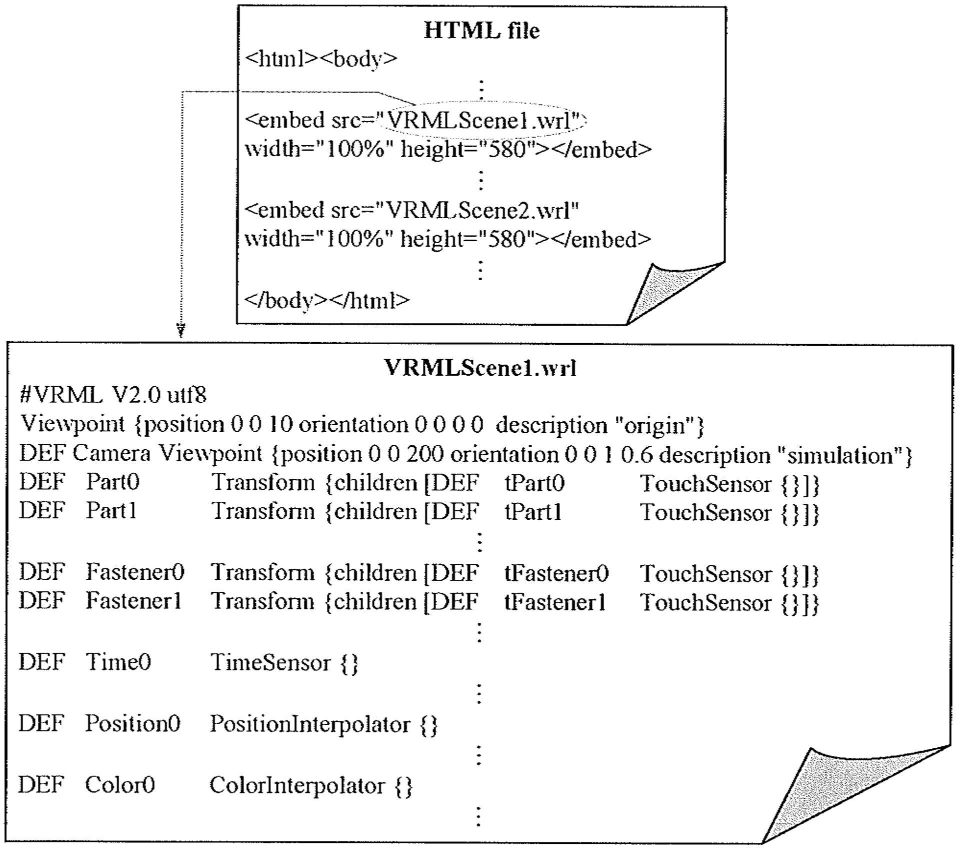
The following are Java program codes written in a Java method named *start()* where VRML browsers, VRML nodes and accessible VRML variables are sequentially defined.

This program shows how to define the touch sensors of components in a Java applet. The touch sensors are efficiently used for an interaction between a user and VRML objects.

```
public void start(){
        try {
                int i, j;
                browser0=BrowserFactory.getBrowser(this,null,0);
                if(browser0!=null){System.out.println("Accessing Browser0");}
                                                                ⋮
                for (i=0, j=0;i<nTotalNode;i++) {
                        Node a;
                        if (i<nTotalPartNode)   /* total number of parts retrieved */
                                a=browser0.getNode("tPart"+i);
                        else
                                a=browser0.getNode("tFastener"+j++);
                        a_TouchSens_changed[i]=(EventOutSFBool)a.getEventOut
                                ("isActive");
                        a_TouchSens_changed[i].setUserData(new Integer(i));
                        a_TouchSens_changed[i].addVrmlEventListener(this);
                                                                ⋮
                }
                                                                ⋮
        } catch(VrmlException e){System.out.println("error occurred : "+e);}
}
```

For instance, a VRML browser, *browser0*, is obtained within a Java applet by calling up the *BrowserFactory.getBrowser()* method. Then, its nodes are defined by invoking the *getNode()* method. These nodes here are VRML touch sensor nodes, each of which is attached to a component, either a part or a fastener. An *eventOut* variable (*i.e.*, *isActive*) of the defined VRML node is assigned to an *a_TouchSens_changed* variable declared in a Java applet. Subsequently, this *a_TouchSens_changed* variable interfaces with *VrmlEventListener* to await an event message. Moreover, each *a_TouchSens_changed* variable is given a serial number by invoking the *setUserData()* method in this research. This allows quick identification of a VRML object generating an event message during user-VRML interactions.

In the above program code, a defined touch sensor node is the one bound with *DEF* to the node named $tPart_i$ or $tFastener_i$ in a VRML file. *DEF* in VRML is a VRML keyword and enables a VRML node to be given a specific name that is used for other nodes by just calling the defined name. Node names such as $tPart_i$ and $tFastener_i$ should be predefined within an empty *Transform* node in a VRML file (**.wrl*). This VRML file is also designated in the same HTML file embedding Java applets. Figure 8.9 shows a predefined VRML file in this research, which is integrated into an HTML file with the HTML tag *<embed>*.



**HTML file**

```
<html><body>
                    :
<embed src="VRMLScene1.wrl"
width="100%" height="580"></embed>
                    :
<embed src="VRMLScene2.wrl"
width="100%" height="580"></embed>
                    :
</body></html>
```

**VRMLScene1.wrl**

```
#VRML V2.0 utf8
Viewpoint {position 0 0 10 orientation 0 0 0 0  description "origin"}
DEF Camera Viewpoint {position 0 0 200 orientation 0 0 1 0.6 description "simulation"}
DEF  Part0       Transform {children [DEF  tPart0       TouchSensor {}]}
DEF  Part1       Transform {children [DEF  tPart1       TouchSensor {}]}
                                          :
DEF  Fastener0   Transform {children [DEF  tFastener0   TouchSensor {}]}
DEF  Fastener1   Transform {children [DEF  tFastener1   TouchSensor {}]}
                                          :
DEF  Time0       TimeSensor {}
                                          :
DEF  Position0   PositionInterpolator {}
                                          :
DEF  Color0      ColorInterpolator {}
                                          :
```

**Figure 8.9**: Predefined VRML file integrated into an HTML file with the HTML tag *<embed>*.

As shown in Figure 8.9, a VRML node bound with *DEF* can be dynamically controlled by a Java program. For instance, *Part0* node in the *VRMLScene1.wrl* is used to represent part geometry, appearance and transformations, while its embedded touch sensor named *tPart0* is used to generate an event message during user-VRML interactions. Moreover, other empty nodes in Figure 8.9 are predefined in the VRML file, which include time sensors, position interpolators and color interpolators. These nodes are used to dynamically implement various 3D animations in executing dynamic task planning developed in this research.

Once VRML browsers, VRML nodes and accessible VRML variables are defined in a Java program, it waits to get an event message generated in a VRML browser. A VRML-EAI method named *eventOutChanged()* accomplishes this function in the Java program. The following Java program code describes how this method is applied in this research.

```
VRMLGeometryData[] vr;
VRMLModelHandler vh=new VRMLModelHandler();
                                                  ⋮
public void eventOutChanged(VrmlEvent event){
        ItemEvent a=null;
        String nodeName;
        Integer infor=(Integer)event.getData();
        int inf=infor.intValue();
        float[] transVal=new float[3];
        if (inf<nTotalNode) {
                lstPartList_itemStateChanged(a);
                lstFastList_itemStateChanged(a);
                this.lstPartList.select(inf);
                nodeName=lstPartList.getSelectedItem();
                if (funcKey==0) vh.removeComponent(browser0, vr, nodeName);
                if (funcKey==1) vh.LoadComponent(browser0, vr, nodeName);
                if (funcKey==2) vh.highLightOneComponent (browser0, vr, nodeName);
                if (funcKey==3) vh.hideComponent (browser0, vr, nodeName);
                if (funcKey==4) vh.showComponent (browser0, vr, nodeName);
                if (funcKey==5) {
                        vh.moveComponent (browser0, nodeName);
                        transVal=a_translation_changed[inf].getValue();
                        vh.movingAngleCone(transVal);
                }
                                          ⋮
        }
                                          ⋮
}
```

In the above program, an event message is delivered from a VRML browser to *eventOutChanged()* method in the Java applet program. Then, the program finds the assigned number of the VRML object that generates the event. Based on this number and a function key triggered by a user, the program in this research invokes various interactions including loading, hiding, deleting, showing, highlighting and moving a VRML object. The following sections describe two major interactions implemented in this chapter, which are static and dynamic VRML-EAI-Java interactions.

### *8.5.2.1. Static VRML-EAI-Java interaction*

The static interaction is intended to manipulate three properties of a VRML object, which are its geometry, appearance (*i.e.*, colors, lighting and textures) and the transformations of its position, orientation and scaling. The *getEventIn()* method of a defined node is effectively used to achieve this interaction. This method allows an external program to change the properties of a defined VRML node.

VRML geometry definition and deletion are achieved by manipulating the *addChildren* and *deleteChildren* property of a target VRML node. *EventIn* typed variables in VRML are used to set a value into the VRML node. For instance, the following Java program codes describe how to load a VRML model into a VRML browser. In the program code, the *EventInMFNode* typed variable named *set_ROOT_addChildren* is declared to handle the *addChildren* property of the target node. Then, the *Node[]* typed variable named *node* initially stores the actual VRML code of a component to be loaded. This VRML code is text-based and dynamically generated via Java. The defined *node* variable is transferred to the VRML node via the *setValue()* method.

```
public void LoadComponent(Browser b, VRMLGeometryData[] vr, String nodeName) {
        browser=b;
        int arrayNo;
        /* Find the array number of the VRMLGeometryData with the nodeName */
        Node root=browser.getNode(nodeName);
        EventInMFNode set_ROOT_addChildren=(EventInMFNode)root.getEventIn
                ("addChildren");
        Node[] node=browser.createVrmlFromString("Transform { \n" +
                " translation 0 0 0\n" + " children [\n" + vr[i].getVRMLCode() +"]}");
        set_ROOT_addChildren.setValue(node);
}
```

Similarly, the appearance and transformations of a node are also set or changed by manipulating node properties via *getEventIn()* and *setValue()* methods. For instance, node appearance including color, lighting and texture changes by manipulating such properties as *transparency*, *diffuseColor*, *emissiveColor*, *shininess* and *specularColor*. In this research, this appearance manipulation is being used for several user-interactive effects including selecting, hiding and showing a VRML object. The following Java program codes describe how to highlight a VRML object by a user selection.

```
public void highLightOneComponent (Browser b, VRMLGeometryData[] v, String
nodeName) {
        int i;
        browser=b;
        EventInSFColor set_HightLight_Color;
        float[] fColor=v[0].getSelectionColor();
        for (i=0;i<v[0].nTVRML;i++) {
                Node root=browser.getNode("m"+v[i].name);
                set_HightLight_Color=(EventInSFColor)root.getEventIn("diffuseColor");
                set_HightLight_Color.setValue(v[i].getOrigColor());
        }
        Node root=browser.getNode("m"+nodeName);
        set_HightLight_Color=(EventInSFColor)root.getEventIn("diffuseColor");
        set_HightLight_Color.setValue(fColor);
}
```

In the above program code, all VRML objects in a VRML browser are changed to their original colors by setting the *diffuseColor* property via the *getEventIn()* and *setValue()* methods. Then, a pre-defined color is assigned to a selected VRML object by *nodeName*.

### *8.5.2.2. Dynamic VRML-EAI-Java interaction*

Dynamic interactive animations are implemented by VRML interpolators including *PositionInterpolator* and *ColorInterpolator*. These VRML interpolators create time-dependant animations with a VRML object. The basic animation concept via these interpolators is illustrated in Figure 8.10.

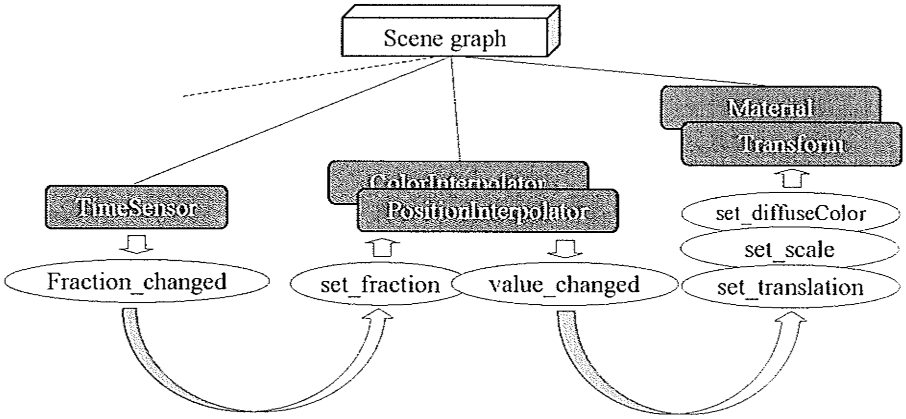
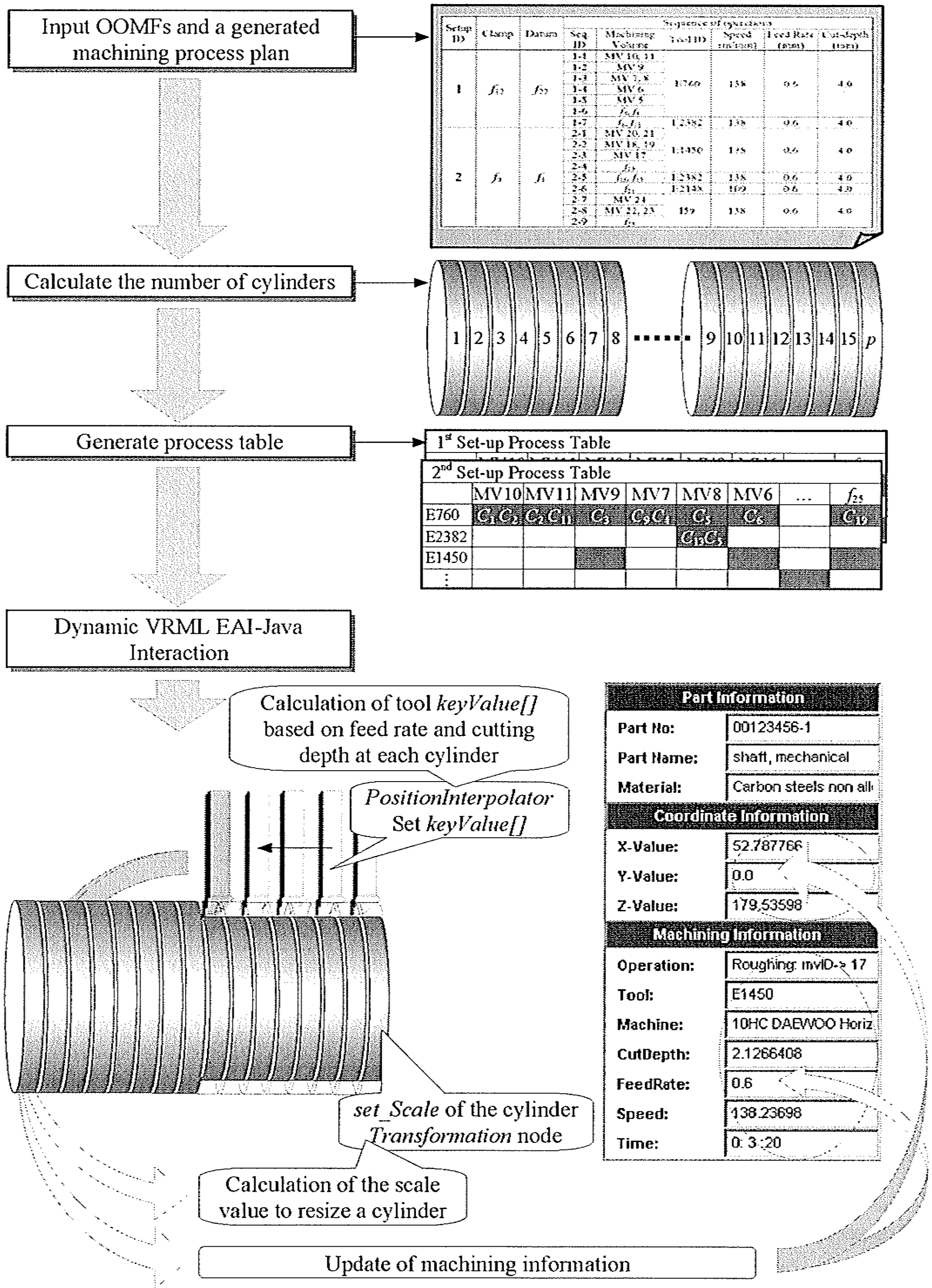


**Figure 8.10**: Basic animation concept via the VRML interpolators.

As shown in Figure 8.10, a clock based on *TimeSensor* starts generating events directly after it is created. The value of its *eventOut fraction_changed* is a type of *SFFloat* (single float data type) and its value is in the interval key value [0, ..., 1]. Subsequent events have increasing values. By default, if the end of the interval is reached, no more events are generated. If the field loop has true value, then, when reaching the end of the interval, the clock starts over at the beginning of the interval. As shown in Figure 8.10, the event *fraction_changed* is routed to the *event_fraction* of a node of type either *PositionInterpolator* or *ColorInterpolator*. This node generates the *eventOut value_changed* of type *SFVec3f* (*i.e.*, positions in 3D space or RGB color codes). These are sent to such *eventIn* properties as *set_translation*, *set_scale* and *set_diffuseColor*.

The following describes a dynamic VRML-EAI-Java interaction for the machining process presented in Chapter 3. Figure 8.11 shows the concept of this interaction process in brief. As shown in Figure 8.11, OOMFs and a generated process plan are imported into the VRML simulation module. The OOMFs are used to define a number of cylinders because this research uses a set of VRML cylinder nodes for simulating a lathe machining process. The radius of each VRML cylinder node is the same size as that of a chosen workpiece, and its thickness is assigned a default value, 0.5mm. For instance, there are 480 VRML cylinders defined when selecting an initial workpiece with 240 mm.

Process tables in Figure 8.11 are formed by the defined cylinders and the generated process plan. At a sequence, each MV consists of a series of VRML cylinders. When a tool moves along an MV, its cylinders are resized by manipulating the *set_scale* property of the *Transformation* node. Once process tables are completely formed, a dynamic VRML-EAI-Java interaction is executed as shown in Figure 8.11. At each cylinder that belongs to an MV to be machined, the following processes are repeatedly executed, which are: (1) calculation of *keyValue [Current_Tool_Position, Next_Tool_Position]*; (2) initiation of a *PositionInterpolator*; (3) transmission of the *keyValue[]* to the *PositionInterpolator*; (4) a tool move and stop; and (5) an update of coordinate and machining information including accumulated machining time.


Input OOMFs and a generated machining process plan
Calculate the number of cylinders
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
$p$
Generate process table
Dynamic VRML EAI-Java Interaction
Calculation of tool *keyValue[]* based on feed rate and cutting depth at each cylinder
*PositionInterpolator* Set *keyValue[]*
*set_Scale* of the cylinder *Transformation* node
Calculation of the scale value to resize a cylinder
Update of machining information

| Setup ID | Clamp | Datum | Sequence of operations | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Seq. ID | Machining Volume | Tool ID | Speed (m/min) | Feed Rate (mm) | Cut-depth (mm) |
| 1 | $f_{12}$ | $f_{22}$ | 1-1 | MV 10, 11 | E760 | 138 | 0.6 | 4.0 |
| | | | 1-2 | MV 9 | | | | |
| | | | 1-3 | MV 7, 8 | | | | |
| | | | 1-4 | MV 6 | | | | |
| | | | 1-5 | MV 5 | | | | |
| | | | 1-6 | $f_6, f_1$ | | | | |
| | | | 1-7 | $f_2, f_3$ | E2382 | 138 | 0.6 | 4.0 |
| 2 | $f_3$ | $f_1$ | 2-1 | MV 20, 21 | E1450 | 138 | 0.6 | 4.0 |
| | | | 2-2 | MV 18, 19 | | | | |
| | | | 2-3 | MV 17 | | | | |
| | | | 2-4 | $f_{17}$ | | | | |
| | | | 2-5 | $f_{16}, f_{18}$ | E2382 | 138 | 0.6 | 4.0 |
| | | | 2-6 | $f_{21}$ | E2148 | 109 | 0.6 | 4.0 |
| | | | 2-7 | MV 24 | 159 | 138 | 0.6 | 4.0 |
| | | | 2-8 | MV 22, 23 | | | | |
| | | | 2-9 | $f_{23}$ | | | | |

1st Set-up Process Table

2nd Set-up Process Table

| | MV10 | MV11 | MV9 | MV7 | MV8 | MV6 | … | $f_{25}$ |
|---|---|---|---|---|---|---|---|---|
| E760 | $C_1C_2$ | $C_2C_{11}$ | $C_3$ | $C_9C_4$ | $C_5$ | $C_6$ | | $C_{19}$ |
| E2382 | | | | | $C_{13}C_5$ | | | |
| E1450 | | | ▨ | | | ▨ | | ▨ |
| ⋮ | | | | | | | ▨ | |

**Part Information**

| | |
|---|---|
| Part No: | 00123456-1 |
| Part Name: | shaft, mechanical |
| Material: | Carbon steels non all |

**Coordinate Information**

| | |
|---|---|
| X-Value: | 52.787766 |
| Y-Value: | 0.0 |
| Z-Value: | 179.53598 |

**Machining Information**

| | |
|---|---|
| Operation: | Roughing: mvID-> 17 |
| Tool: | E1450 |
| Machine: | 10HC DAEWOO Horiz |
| CutDepth: | 2.1266408 |
| FeedRate: | 0.6 |
| Speed: | 138.23698 |
| Time: | 0: 3:20 |

**Figure 8.11**: Dynamic VRML-EAI-Java interaction for the machining process presented in Chapter 3.

The following Java program code describes how to implement the tool position interpolation and to resize a VRML cylinder in the process shown in Figure 8.11.

```
boolean isContinue;
Browser b=browser0
float[] fkey={0.0f,1.0f};
float[][] keyValue=new float[2][3];          /* (x_s, y_s, z_s) and (x_e, y_e, z_e) */
Node tool = b.getNode("Tool0");              /* Tool */
Node timer = b.getNode("Timer0");            /* Timer node */
Node position=b.getNode("Position0");        /* PositionInterpolator node */
                                             ⋮
EventInSFTime interval = (EventInSFTime) timer.getEventIn("cycleInterval");
isActive= (EventOutSFBool) timer.getEventOut("isActive");
EventInSFBool loop=(EventInSFBool)timer.getEventIn("loop");
EventInMFFloat key=(EventInMFFloat) position.getEventIn("key");
key.setValue(fkey);
EventInMFVec3f keyVal=(EventInMFVec3f) position.getEventIn("keyValue");
b.addRoute(timer,"fraction_changed",position,"set_fraction");
b.addRoute(position,"value_changed",tool,"set_translation");
                                             ⋮
/* Calculation of keyValue[] */
keyVal.setValue(keyValue);   /* Transmission of KeyValue to PositionInterpolator */
interval.setValue(speed);    /* Determination of animation speed */
loop.setValue(true);         /* Tool activation (Loop mode) */
loop.setValue(false);        /* Turn off the loop mode */
do {
        isContinue=isActive.getValue();
} while (isContinue);
                                             ⋮
/* Find a cylinder k to be resized */
/* Calculate the new radius of the cylinder: newRadius=current radius-cutting depth */
Node cylinder = b.getNode("Cyl"+k);
EventInSFVec3f t_scale = (EventInSFVec3f) cylinder.getEventIn("set_scale");
s_xyz[0]=newRadius/radius;
s_xyz[1]=1f;
s_xyz[2]=newRadius/radius;
t_scale.setValue(s_xyz);
                                             ⋮
b.deleteRoute(timer,"fraction_changed",position,"set_fraction");
b.deleteRoute(position,"value_changed",tool,"set_translation");
```

Similarly, the *PositionInterpolator* is efficiently used in dynamic VRML-EAI-Java interactions for assembly and disassembly VRML simulations. Moreover, this is combined with *ColorInterpolator* to generate a more effective 3D simulation during assembly or disassembly.

## 8.6 Developed Web-based Systems

Three Web-based systems are implemented for dynamic task planning discussed in this dissertation. Figure 8.12 shows the schematic of the entire system architecture.

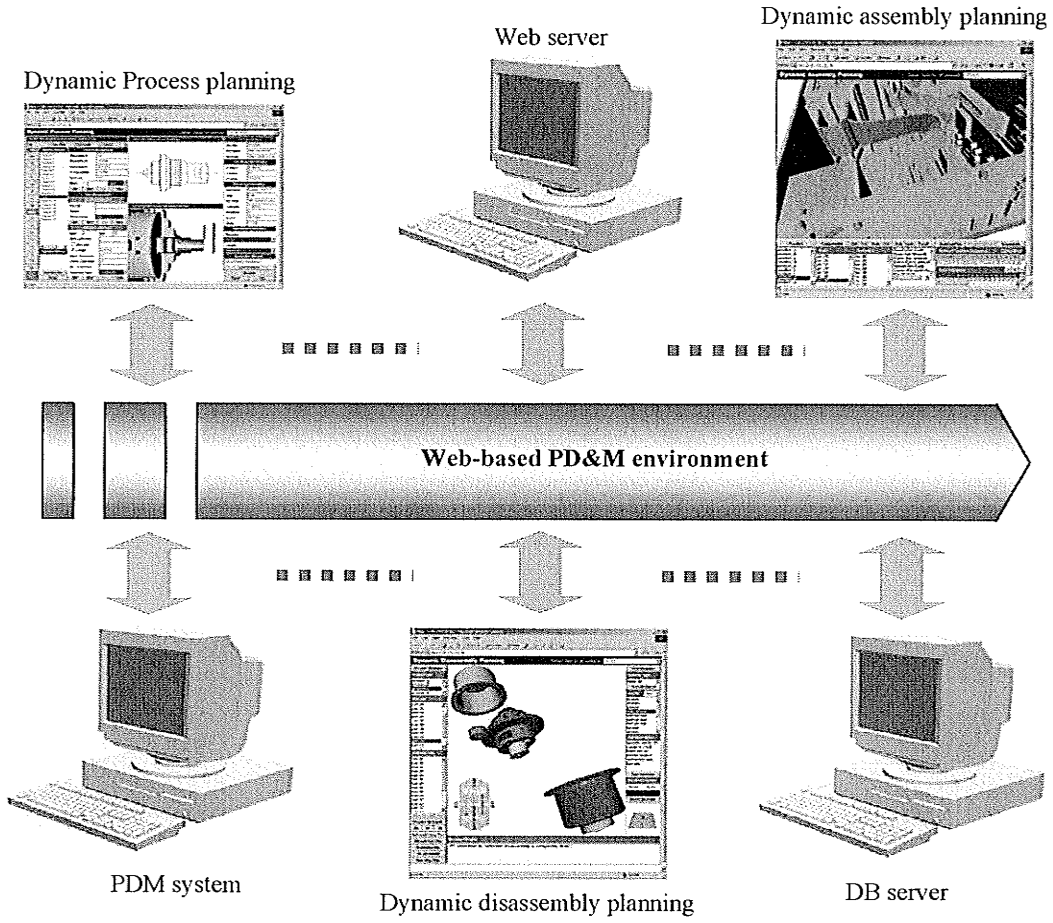

Dynamic assembly planning
Web server
Dynamic Process planning
Web-based PD&M environment
PDM system
Dynamic disassembly planning
DB server

**Figure 8.12**: Schematic of entire system architecture.

In a Web-based PD&M environment, the developed Web-based systems are accessed by users on the client side. The users may be geographically dispersed and work for different companies in an enterprise alignment. They may be CAD designers, process planners, and assembly planners. In planning tasks via the developed Web-based systems, a user designates a DB server location, which stores appropriate manufacturing or DM resources in an enterprise alignment. Then, dynamic task planning processes are executed with the retrieved information from a DB. Generated task plans are subsequently stored in a DB that can be used by other users. Most of the processing performed for dynamic task planning occurs on the client side so that the Web server serves many other users simultaneously.

The developed three Web-based systems are here implemented by Java 1.1.8. This Java version is chosen for matching the currently used Microsoft Java virtual machine (JVM) supporting Internet Explorer. There is no additional Java plug-in required when these Web-based applications run on Internet Explorer. Moreover, the implementation of the three Web-based systems is based on the concept of the OOP paradigm. As an example, the Web-based system for dynamic assembly planning is illustrated in Figure 8.13. Figure 8.13 shows the sequence diagram to show how to model this system, which is represented by unified modelling language (UML).

UML is a widely used visual language for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems. This provides eight different modelling techniques including the use-case, class, and sequence diagrams. Among the modelling techniques provided by UML, the sequence diagram explicitly represents the sequence of messages for a particular case from a use case diagram. In Figure 8.13, client side processes are particularly focused to show the sequence of all messages for dynamic assembly planning.

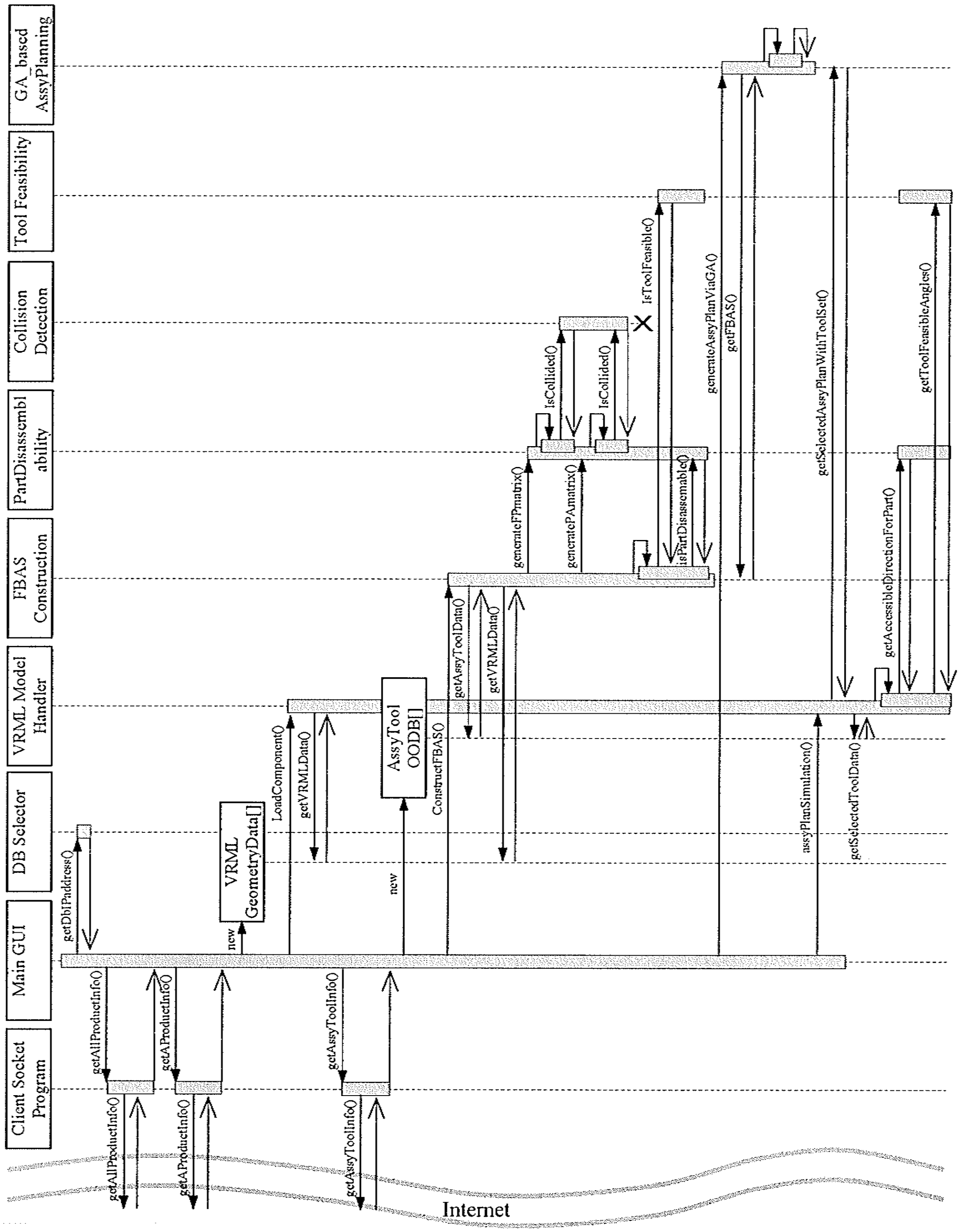


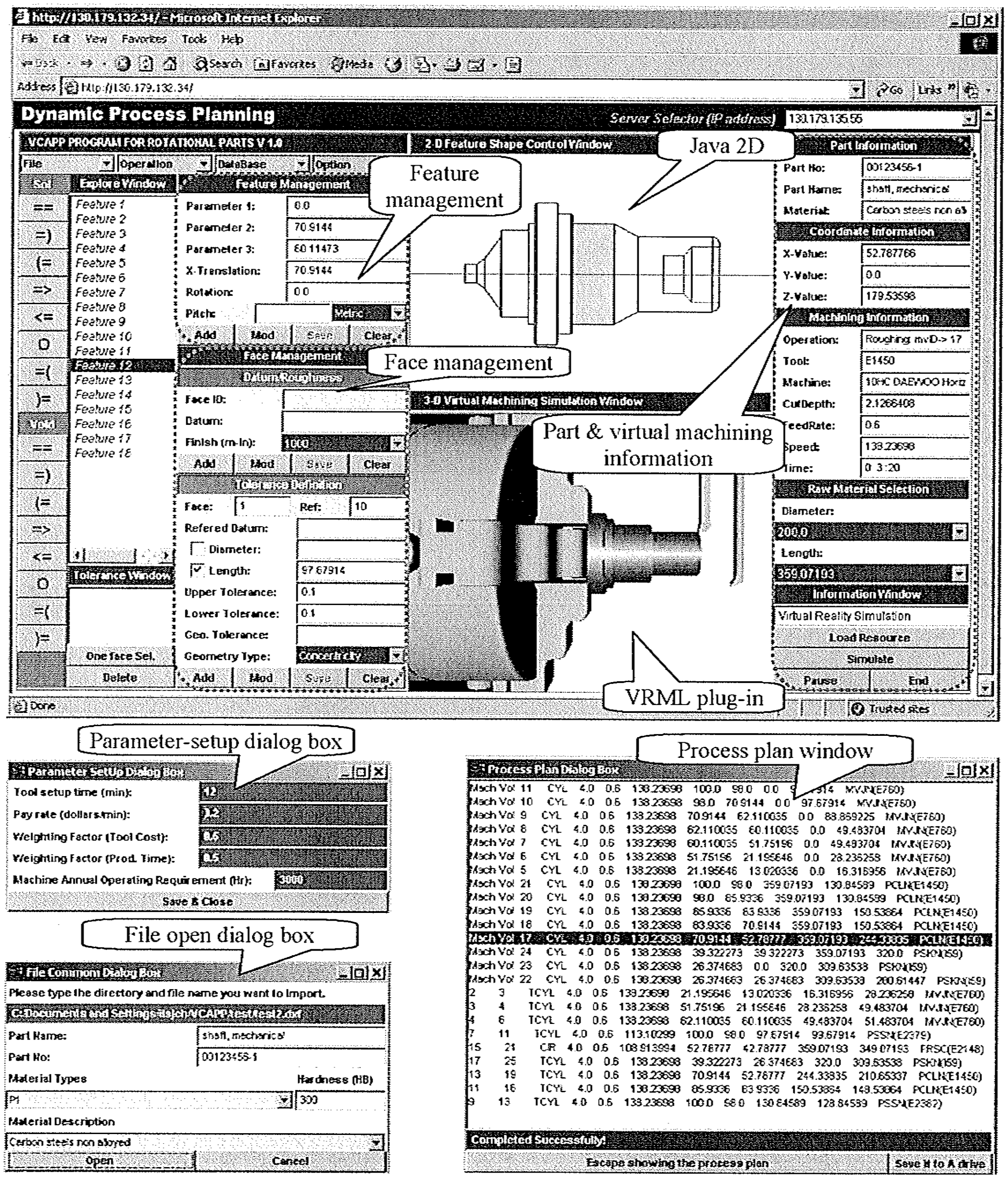
**Figure 8.13**: UML sequence diagram of modelling the Web-based system for dynamic assembly planning.

As shown in Figure 8.13, a message is generated by selecting a DB server location, which is selected by the user. This message is sent to the client socket program to retrieve all product information from the product DB. Then, a chosen product is again sent to the client socket program to retrieve its parts and fasteners. By a Java class named *VRMLModelHandler*, the retrieved parts and fasteners are displayed on a VRML browser where user interactions with a VRML object occur. In particular, the fastener information is used to retrieve all available assembly tools that are related to the retrieved fasteners. The message of assembly tools and part VRML models is sent to the *FBAS_Construction* class to form an FBAS by repeatedly analyzing two aspects: topological part disassemblability and assembly-tool feasibility. Subsequently, the FBAS is used to execute GA-based assembly planning with tool selection.

The following sections describe GUIs of the developed Web-based systems in this research. The GUIs running on the client side are a tool for user interactions with the developed systems. These GUIs consist of Java applets and VRML browsers.

### 8.6.1 Web-based system for dynamic process planning

As shown in Figure 8.14, several GUIs are implemented for dynamic process planning in Web-based PD&M environments. The main GUI in Figure 8.14 consists of five components: (1) an applet for server selector; (2) an applet for feature and face managements; (3) an applet for Java 2D; (4) an applet for part and machining information; and (5) a VRML browser. Moreover, three sub GUIs based on Java *Frame* class support users to: (1) select a part DXF file and choose its material; (2) input parameters used in tools and machines selection; and (3) show a generated process plan that is used for virtual machining simulation.



| | |
|---|---|
| Parameter 1: | 0.0 |
| Parameter 2: | 70.9144 |
| Parameter 3: | 60.11473 |
| X-Translation: | 70.9144 |
| Rotation: | 0.0 |
| Pitch: | Metric |

| | |
|---|---|
| Face ID: | |
| Datum: | |
| Finish (m-In): | 1000 |

| | | | |
|---|---|---|---|
| Face: | 1 | Ref: | 10 |
| Refered Datum: | | | |
| □ Diameter: | | | |
| ☑ Length: | 97.67914 | | |
| Upper Tolerance: | 0.1 | | |
| Lower Tolerance: | 0.1 | | |
| Geo. Tolerance: | | | |
| Geometry Type: | Concentricity | | |

| | |
|---|---|
| Part No: | 00123456-1 |
| Part Name: | shaft, mechanical |
| Material: | Carbon steels non a3 |
| X-Value: | 52.787766 |
| Y-Value: | 0.0 |
| Z-Value: | 179.53598 |
| Operation: | Roughing mvD-> 17 |
| Tool: | E1450 |
| Machine: | 10HC DAEWOO Horiz |
| CutDepth: | 2.1266408 |
| FeedRate: | 0.6 |
| Speed: | 138.23698 |
| Time: | 0 3:20 |
| Diameter: | 200.0 |
| Length: | 359.07193 |

Virtual Reality Simulation

| | |
|---|---|
| Tool setup time (min): | 12 |
| Pay rate (dollars/min): | 1.2 |
| Weighting Factor (Tool Cost): | 0.5 |
| Weighting Factor (Prod. Time): | 0.5 |
| Machine Annual Operating Requirement (Hr): | 3000 |

Please type the directory and file name you want to import.

C:\Documents and Settings\dsjch\VCAPP\test\test2.dxf

| | |
|---|---|
| Part Name: | shaft, mechanical |
| Part No: | 00123456-1 |
| Material Types | Hardness (HB) |
| P1 | 300 |
| Material Description | |
| Carbon steels non alloyed | |

```
Mach Vol 11   CYL  4.0  0.6  138.23698  100.0  98.0  0.0  9  914  MVJN(E760)
Mach Vol 10   CYL  4.0  0.6  138.23698  98.0  70.9144  0.0  97.67914  MVJN(E760)
Mach Vol 9   CYL  4.0  0.6  138.23698  70.9144  62.110035  0.0  83.869225  MVJN(E760)
Mach Vol 8   CYL  4.0  0.6  138.23698  62.110035  60.110035  0.0  49.483704  MVJN(E760)
Mach Vol 7   CYL  4.0  0.6  138.23698  60.110035  51.75196  0.0  49.483704  MVJN(E760)
Mach Vol 6   CYL  4.0  0.6  138.23698  51.75196  21.195646  0.0  28.236258  MVJN(E760)
Mach Vol 5   CYL  4.0  0.6  138.23698  21.195646  13.020336  0.0  16.316956  MVJN(E760)
Mach Vol 21   CYL  4.0  0.6  138.23698  100.0  98.0  359.07193  130.84599  PCLN(E1450)
Mach Vol 20   CYL  4.0  0.6  138.23698  98.0  65.9336  359.07193  130.84599  PCLN(E1450)
Mach Vol 19   CYL  4.0  0.6  138.23698  65.9336  63.9336  359.07193  150.53864  PCLN(E1450)
Mach Vol 18   CYL  4.0  0.6  138.23698  63.9336  70.9144  359.07193  150.53864  PCLN(E1450)
Mach Vol 17   CYL  4.0  0.6  138.23698  70.9144  52.78777  359.07193  244.33835  PCLN(E1450)
Mach Vol 24   CYL  4.0  0.6  138.23698  39.322273  39.322273  359.07193  320.0  PSKN(59)
Mach Vol 23   CYL  4.0  0.6  138.23698  26.374683  0.0  320.0  309.63539  PSKN(59)
Mach Vol 22   CYL  4.0  0.6  138.23698  26.374683  26.374683  309.63539  280.61447  PSKN(59)
2     3    TCYL  4.0  0.6  138.23698  21.195646  13.020336  16.316956  28.236258  MVJN(E760)
3     4    TCYL  4.0  0.6  138.23698  51.75196  21.195646  28.236258  49.483704  MVJN(E760)
4     6    TCYL  4.0  0.6  138.23698  62.110035  60.110035  49.483704  51.483704  MVJN(E760)
7     11    TCYL  4.0  0.6  113.10299  100.0  98.0  97.67914  93.67914  PSSN(E2379)
15     21    CIR  4.0  0.6  108.913994  52.78777  42.78777  359.07193  349.07193  FRSC(E2148)
17     25    TCYL  4.0  0.6  138.23698  39.322273  26.374683  320.0  309.63538  PSKN(59)
13     19    TCYL  4.0  0.6  138.23698  70.9144  52.78777  244.33835  210.65337  PCLN(E1450)
11     16    TCYL  4.0  0.6  138.23698  65.9336  63.9336  150.53864  148.53864  PCLN(E1450)
9     13    TCYL  4.0  0.6  138.23698  100.0  98.0  130.84589  128.84599  PSSN(E2382)

Completed Successfully!
```

**Figure 8.14**: GUIs implemented for dynamic process planning in Web-based PD&M environments.

## 8.6.2 Web-based system for dynamic assembly planning

As shown in Figure 8.15, three GUIs are implemented for dynamic assembly planning in Web-based PD&M environments. The main GUI in Figure 8.15 consists of three components: (1) an applet for server selector; (2) an applet for all control boxes and command buttons; and (3) a VRML browser.

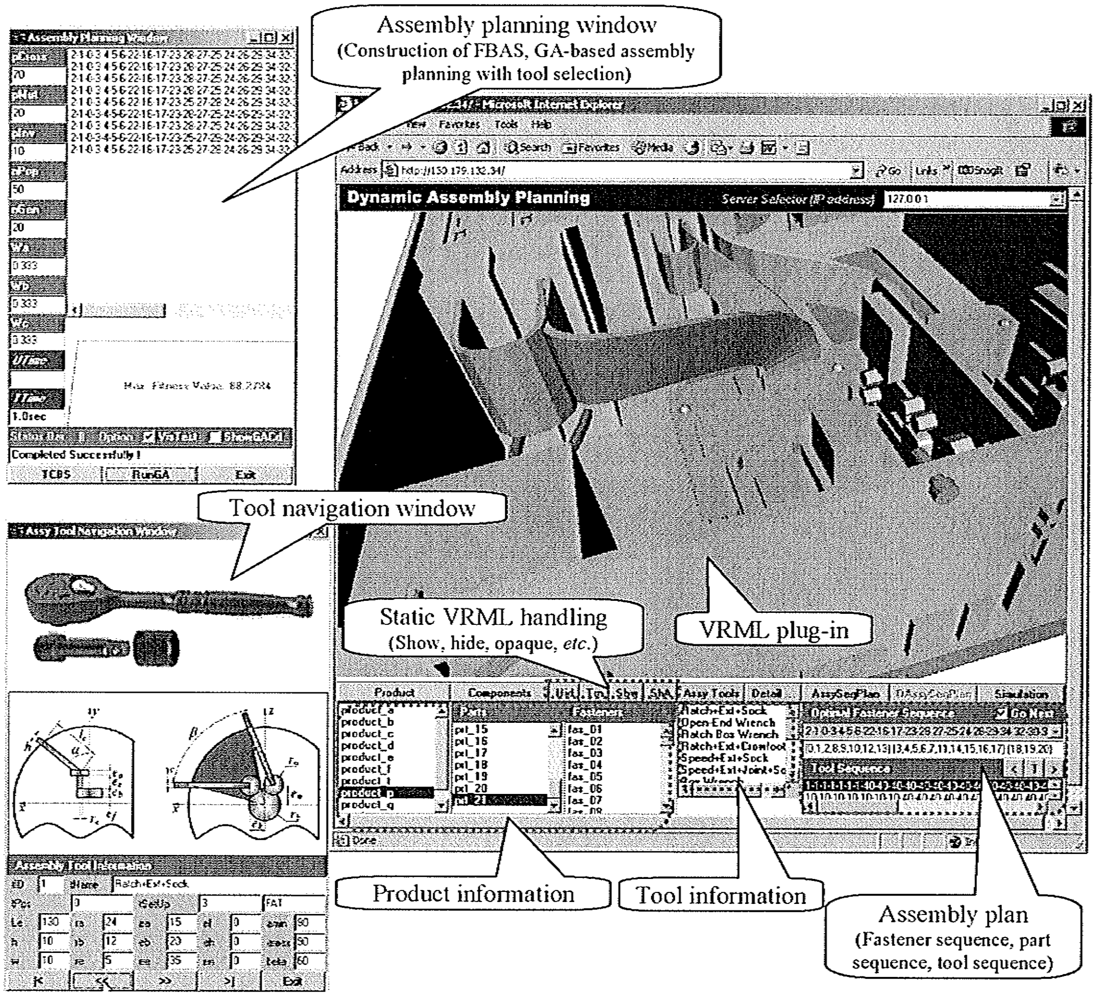


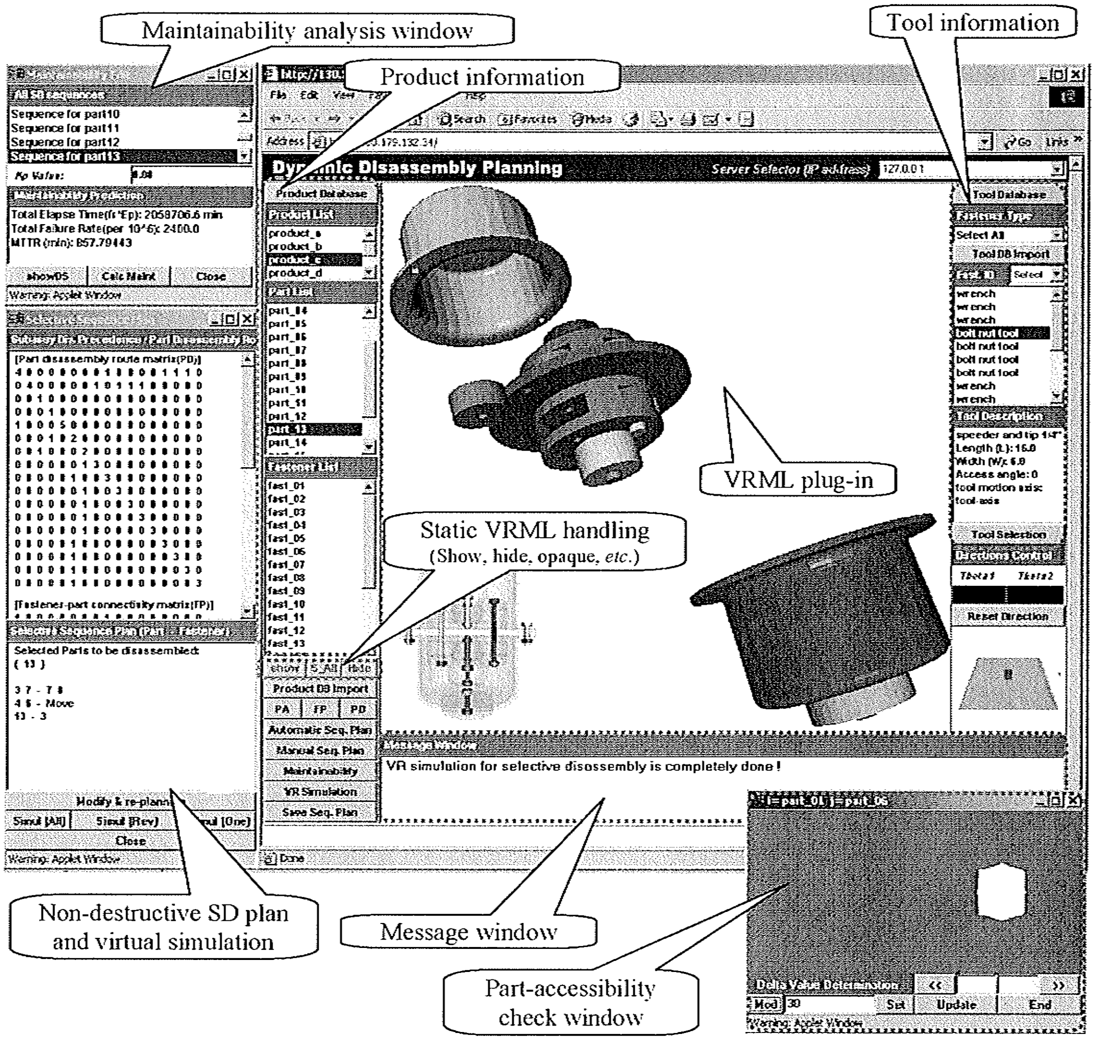
**Figure 8.15**: GUIs implemented for dynamic assembly planning in Web-based PD&M environments.

Moreover, two sub GUIs based on Java *Frame* class support users to: (1) create optimal assembly plans via construing an FBAS and two-tier GA-based assembly planning, and (2) show retrieved or selected assembly tool information. By a user request, the *Tool navigation window* shown in Figure 8.15 retrieves additional tool information from the assembly-tool DB, which includes tool images. Thus, a DB connection occurs when a user clicks on a navigation button in the *Tool navigation window*.

### 8.6.3 Web-based system for dynamic disassembly planning

Figure 8.16 shows several GUIs implemented for dynamic disassembly planning in Web-based PD&M environments. The main GUI in Figure 8.16 consists of six components: (1) an applet for server selector; (2) an applet for product retrievals or displaying product information; (3) an applet for tool retrievals or displaying tool information; (4) an applet for displaying messages; and (5) two VRML browsers including a VRML model navigation plug-in.

In addition to the *Tool navigation window* shown in Figure 8.15, this application has three sub GUIs: (1) an *SD plan edit window* allowing for modification of a generated SD plan and execution of its virtual simulation; (2) a *maintainability analysis window* that can invoke the *SD plan edit window* for each SD plan; and (3) a *part accessibility check window* that can monitor or modify defined part accessibility, $PA^{i,j}$.



Total Elapse Time(f*Ep): 2058706.6 min
Total Failure Rate(per 10^6): 2400.0
MTTR (min): 857.79443

Selected Parts to be disassembled:
{ 13 }

VR simulation for selective disassembly is completely done!

**Figure 8.16**: GUIs implemented for dynamic disassembly planning in Web-based PD&M environments.

## 8.7 Chapter Summary

This chapter presented the methodology to implement the Web-based systems, their structure and functions, and GUIs. As the backbone of a Web-based PD&M environment, a three-tier architecture on the Internet was implemented in this research. Thus, the

)

developed Web-based systems for dynamic task planning run on this three-tier Internet-based architecture. This architecture consisting of a Web server, a DB server and clients is advantageous in implementing a distributed and collaborative Web-based application.

This chapter also presented various communication techniques used between Web server and clients, Web server and DB servers, and Java applets and VRML-EAI in a Web browser. In particular, the communication between Java applets and VRML-EAI were discussed in detail. Based on this communication, various static and dynamic interactions were implemented in developing Web-based systems for the proposed dynamic task planning of this research. Subsequently, several sections at the end of this chapter included practical issues in developing Web-based software, which include a UML sequence diagram and GUI functions.

With respect to the research on the Web-based implementation of the dynamic task planning, published papers in international journals and conferences are as follows.

Peng, Q., **Chung, C.**, Yu, C., and Luan, T. (2006). A networked virtual manufacturing systems for SMEs, International Journal of Computer Integrated Manufacturing (In Press).

**Chung, C.** and Peng, Q. (2006). A dynamic tasks planning system for Web-based collaborative product development. 2006 CIRP International Design Seminar on Design and Innovation for Sustainable Society. Kananaskis, Alberta, Canada, July 16-19.

**Chung, C.** and Peng, Q. (2005). Selective disassembly sequence planning on the Internet. International Journal of Engineering Simulation (IJES) 6(1): 10-16, ISSN 1468-1137.

# Chapter 9

# Conclusions and Future Work

This chapter concludes the dissertation and discusses the contributions of this research. In particular, the dynamic task-planning requirements presented in Chapter 1 are reviewed. Possible future directions for extending the work presented in this dissertation are also discussed.

## 9.1 Research Summary

Web-based PD&M is regarded as a key to achieving AM in the 21st century business environment. The main thrusts of the Web-based PD&M are concurrency, attention to the product life cycle, collaboration among multidisciplinary members including suppliers and customers, high-fidelity simulation and validation, and integration with other Internet-based business systems. Dynamic task planning is performed based on resource sharing and involvement in planning tasks at the different product life-cycle stages. It is critical to expedite Web-based PD&M execution of such activities as: (1) exploiting business opportunities in a virtually formed enterprise alignment; (2) executing reliable assessment of product design alternatives in a discipline point of view; and (3) realizing disciplines that occur at the latter stages of the product life cycle.

The objective of this research was to develop efficient approaches to dynamic task planning, and to integrate them into a Web-based PD&M environment. Among the task-planning activities of tasks at different product life-cycle stages, the three major activities developed were: (1) process (fabrication) planning at the shop floor level; (2) assembly planning at the production level; and (3) disassembly planning at the product end-life level.

As a research strategy, seven key issues were suggested in Chapter 1, which are: (1) modelling knowledge and resource DBs; (2) developing DB search algorithms; (3) developing operation methods; (4) defining OODBs; (5) developing task optimization methods; (6) developing 3D visualization methods; and (7) implementing Web-based integration. Based on the research strategy, the objective of this research was effectively achieved.

In particular, the developed approaches for three task-planning activities were *generic* to deal with various products manufactured by SMEs to LMEs. They were not product-oriented but task-oriented approaches. The developed approaches in this research can be applied if a product simply needs a turning process, or if it needs a meticulous examination in terms of part disassemblability and assembly-tool feasibility. The used knowledge and DB were based on standards including ISO, so that they can be used in a manufacturing company without significant modification. Moreover, the developed Web-based systems were built on neutral and standardised technologies including Java, VRML, RDB and JDBC. This allows platform-independent and cost-effective solutions for SMEs to LMEs, not requiring application development and extra infrastructure.

The developed approaches were *dynamic* to timely respond to a manufacturing or DM environment. With organized and structured generic DBs, DB search engines based on SQL queries worked at dynamically retrieving available tools and machines from a remote DB server. Depending on the retrieved resources, the approaches can generate different timely task plans with the aid of the developed elements including OODBs, and operation and optimization methods.

The developed elements also helped to achieve the *rapid* generation of task plans that are used as economical and optimal *what-if* scenarios in Web-based PD&M environments. Several novel methods, including the assembly-tool feasibility analysis, efficiently supported resource-involved task planning. They generate optimal and feasible plans, and the required resources rapidly. Thus, the resource-involved task planning, with resource sharing over the Internet, helped to generate *precise* task plans. The plans can minimize potential errors that are related to product life cycle disciplines to be performed within an enterprise alignment. Reliable time and cost estimation of the intended tasks can be achieved by the selected timely available resources including tools and machines. Later, the *precise* task plans can be used for supporting core activities in Web-based PD&M to reduce *time-to-market* significantly.

It was accomplished to generate feasible and economical task plans at the shop floor, production and product end-life levels via the Web-based dynamic task-planning systems developed. The generated task plans can be used for quickly exploiting manufacturing opportunities for a specific product at the inter-enterprise level. They can help to realize such product life-cycle disciplines as *fabrication*, *assembly*, *maintenance*, *product reuse* and *product recycling*. In addition, the task plans can provide Web-based VM systems with a reduced number of feasible *what-if* scenarios regarding the disciplines, and serve at supporting such DFX systems as DFM, DFA, DFD, DFR and DFMa.

## 9.2 Summary of Contributions

The following are the summary of contributions made through this research. They render useful service to the development of efficient approaches to dynamic task planning, and to the implementation of the developed approaches in a Web-based PD&M environment.

***A fast assembly-tool reasoning method based on a geometric accessibility analysis***. The appropriate consideration of tool feasibility is critical in complete assembly or disassembly planning. The developed method is proficient in achieving this. It is based on a parameterized assembly tool and a $GAC^d$ that approximates the obstacles facing assembly or disassembly of the fastener. It avoids the used of complex collision-detection methods. In particular, the developed method uses a simple algebra in the defined searching range of a $GAC^d$, which can quickly determine the geometric feasibility of an assembly tool. The method can simply deal with complex variations in a fastener movement and a tool access angle, so that it is possible to execute a practical feasibility analysis for an assembly-tool.

***A topological part disassemblability analysis via several triangle patch-based collision detection techniques***. A systematic method was developed for part disassemblability analysis. It is based on the geometric accessibility analysis of parts. With the tool feasibility, the topological part disassemblability is critical in determining the priority of parts being assembled or disassembled. Several collision detection methods based on triangle patches were efficaciously applied, namely BS-, OBB-, and triangle patch-based collision detection methods. In particular, a D$^3$C was introduced to represent a part accessibility $PA^{i,j}$. In determining complicated topological disassemblability, the D$^3$C allows mapping 3D geometric constraints between parts into a 2D directionality map via the two mapping algorithms developed. This is advantageous in not only executing operations including merging all $PA^{i,j}$ in the matrix $PA$, but also searching available directions for a part assembly or disassembly.

***A two-tier GA-based approach to dynamic assembly planning with a number of tool alternatives***. A two-tier GA was applied for the tool selection-embedded optimal assembly planning in this research. The two-tier GA was executed on a constructed FBAS. It consists of two GA-based optimizations: (1) an assembly level based GA for creating the initial population and (2) a product based GA for generating an optimal assembly sequence. This approach effectively works in solving a large-scale optimization problem. In this research, it helped to simultaneously optimize fastener-based assembly sequences and assembly tool sequences with a reduced number of generations or iterations.

***A matrix-based heuristic approach to near-optimal SD sequence planning***. An approach was developed to support near-optimal SD sequence planning based on a dynamic DM environment. It is executed with primitive matrices that are generated from a given product. Two analyses presented in Chapters 4 and 5 are incorporated into this approach. For randomly selected parts, the developed approach generates an efficient and practical SD plan of the product. The SD plan defines a partial and incomplete disassembly sequence, minimizing SD efforts via forming modularized removals that satisfy assembly tool feasibility and part disassemblability.

***The extensive use of RDB and JDBC technologies in modeling, managing and searching knowledge and resources***. Standard DB technologies including RDB and JDBC were extensively applied in this research. They were used to appropriately model, manage and search various knowledge and resources related to manufacturing and DM. Based on dynamic SQL queries and search criteria defined, DB search algorithms were also developed. They minimize a number of transactions with a remote DB server over the Internet and retrieve a reduced amount of data. The reduced amount of data subsequently helps to achieve resource-involved task planning efficiently. In particular, JDBC provides easy DB connectivity for other resource-related systems such as inventory and scheduling systems. These systems in an enterprise can concurrently work with dynamic resources that are used by dynamic task planning systems.

***The efficient use of the OOP paradigm in defining OODBs to appropriately manipulate data retrieved on the client***. Various OODBs were utilized to efficiently manipulate information that is retrieved from DBs in a remote DB server, or newly generated during dynamic task planning. The OODBs in this research are defined as a set of instances. The instances are based on the Java classes developed, consisting of encapsulated attributes and methods. A set of essential data formed from DBs is set to the attributes in each OODB. The developed operation methods are mainly interfaced to the methods of the OODB. Therefore, the use of OODBs provides an easy and effective manipulation of various data used for dynamic task planning.

***Web-based Interactions of Java and VRML-EAI***. In this research, two VRML-based interactions, static and dynamic interactions, were implemented by Java and VRML-EAI on the Internet. The static interaction is intended to manipulate and acquire three properties of a VRML object, which are geometry, appearance (*i.e.*, colors, lighting and textures) and transformations of position, orientation and scaling. The dynamic interaction is implemented by using VRML interpolators including *PositionInterpolator* and *ColorInterpolator*. It is used to create time-dependant VRML-based animations with a VRML object. This is advantageous in implementing a realistic 3D simulation and acquiring real time information from the object. The acquired information is used for other computations including the estimation of machining time and required distance to move a part being disassembled. Finally, the two VRML-based interactions produce efficient platform-independent, cost-effective and visualized information sharing in a Web-based PD&M environment.

## 9.3 Future Work

For extending the work presented in this dissertation, possible future directions are as follows: (1) the development of the approaches to other aspects of task planning; (2) a seamless link with scheduling; (3) the place of human involvement in dynamic task planning; and (4) the use of advanced Web-technologies.

This research discussed the following three task planning activities: (1) process (fabrication) planning at the shop floor level; (2) assembly planning at the production level; and (3) disassembly planning at the product end-life level. A direction to go in further research would be the development of dynamic task planning systems for other product life-cycle disciplines. The disciplines can include quality control, product distribution, product disposal, and materials recycling. In Web-based PD&M environments, this allows DFLC to generally assess product-life cycle issues, and precisely estimate PLCC. The PLCC are cost of product development and manufacturing, cost of operation, maintenance and service, and cost of product DM.

Although there is an abundance of work in the *integration* of task planning and scheduling, it would be impractical because of their different scopes and aspects. Task planning emphasizes the technological requirements of only a product being designed while scheduling involves the aggregate requirements of various products to manufacture and the timing aspects. Moreover, the nature of scheduling is so dynamic. Although an economical machine in dynamic task planning is selected based on its timely availability at a dynamic shop floor, for instance, the machine can break down or its utilization can be delayed unexpectedly. This can lead to the generation of an infeasible or sub-optimal production plan by a scheduling process executed just before getting to production. A seamless link between dynamic task planning and scheduling is highly demanded in a Web-based PD&M environment. It covers the efficient collaborations including monitoring and control between scheduling and dynamic task planning.

Resource involvement and sharing were mainly emphasised in achieving dynamic task planning in this research. However, they only considered tools and machines in a manufacturing or DM environment. Among the resources in enterprises, it is natural that people are the most valuable, flexible and complex one. Unlike a machine or tool that is designed to perform intended functions, humans have various potentials. They can do better work for a given job, but they can also bring more problems. Sometimes, a *synergic* effect occurs when human beings work together in enterprises. In order to completely minimize uncertainty issues in a Web-based PD&M environment, it is highly recommended that people be involved as a resource in the process of dynamic task planning. Human-involved task-planning covers not only the generation of realistic, economical and optimal task plans, but also the analysis of potential risks occurring by people. Human involvement can be represented as micro-ergonomics (*i.e.*, human-machine interfaces) and macro-ergonomics (*i.e.*, human-human interfaces with organisational and cultural factors).

A technical direction to implement in Web-based dynamic task planning systems could be in the use of advanced Web technologies including extensible mark-up language (XML), Web-service, and agents. The Web technologies enhance interoperability between heterogeneous business applications, security, and distributed autonomy. In particular, a new standard, extensible 3D (X3D) has been developed by the Web3D Consortium. This new standard received ISO approvals in November 2005 (ISO, 2005), including (1) X3D encodings: XML and classic VRML (ISO/IEC 19776: 2005), and (2) X3D language bindings: Java (ISO/IEC 10777-2: 2005). As an application of XML, X3D has an XML encoding that makes it easier to mange, control, validate, and exchange information. It provides adequate specifications in such a way that scenes and environments can interoperate between browsers or users.

# Bibliography

Ahmad, N., Haque, A., and Hasin, A., "Current trend in computer aided process planning", Proceedings of the 7th Annual Paper Meeting and 2nd International Conference, pp. 81-92, 2001.

Akagi, F., Osaki, H., and Kikuchi, S., "The method of analysis of assembly work based on the fastener method", Bulletin of the JSME, Vol. 23, pp. 1670-1675, 1980.

Apache, Apache HTTP Server Project, http://www.httpd.apache.org, 2005.

Basdere, B., and Seliger, B., "Disassembly factories for electrical and electronic products to recover resources in product and material cycles", Environmental Science and Technology, Vol. 37, No. 23, pp. 5354-5362, 2003.

Blaxxun Co., http://www.blaxxun.com/home/index.php?option=com_content&task=view&id=37&Itemid=140, 2005.

Bras, B., Design for Recycling, www.srl.gatech.edu/education/ME4171/DFR-Intro.ppt, 2004.

Cao, D., Chen, M., and Wan, G., "Parallel machine selection and job scheduling to minimize machine cost and job tardiness", Computers and Operations Research, Vol. 32, pp. 1995-2012, 2005.

Cao, Q., and Dowlatshahi, S., "The impact of alignment between virtual enterprise and information technology on business performance in an agile manufacturing environment", Journal of Operations Management, Vol. 23, pp. 531-550, 2005.

Chang, T., Wysk, R. A., and Wang, H., Computer-Aided Manufacturing (2nd Edition), Prentice Hall, 1997.

Chen, S. F., and Liao, X., "Stable assembly sequence planning using a genetic algorithm" Proceedings of ASME Design Engineering Technical Conferences, pp. 1-7, 1999.

Chen, S. F., and Liu, Y. J., "An adaptive genetic assembly-sequence planner", International Journal of Computer Integrated Manufacturing, Vol. 14, pp. 489-500, 2001.

Choi, C. K. A., Chan, S. K. D., and Yuen, M. F. A., "Application of virtual assembly tools for improving product", International Journal of Advanced Manufacturing Technology, Vol. 19, pp. 377-383, 2002.

Cui, J., and Forssberg, E., "Mechanical recycling of waste electric and electronic equipment: a review", Journal of Hazardous Materials, Vol. 99, pp. 243-263, 2003.

Davis, M. J., Keys, K. L., Chen, J. I., and Petersen, L. P., Collaborative Product Development in an R&D Environment, NASA/TM 2004-212967, 2004.

Dereli, T., Development of a Process Planning System for Prismatic Parts, PhD thesis, University of Gaziantep, Turkey, 1998.

Desai, A., and Mital, A., "Evaluation of disassemblability to enable design for disassembly in mass production", International Journal of Industrial Ergonomics, Vol. 32, pp. 265–281, 2003.

Diehl, S., Distributed Virtual Worlds-Foundations and Implementation Techniques Using VRML, Java, and CORBA, Springer-Verlag, Berlin, 2001.

Dowlatshahi, S., and Cao, Q., "The relationships among virtual enterprise, information technology, and business performance in agile manufacturing: An industry perspective", European Journal of Operational Research, (Online available), 2005.

Fernandes, J. K., and Raja, H. V., "Incorporated tool selection system using object technology", International Journal of Machine Tools and Manufacture, Vol. 40, pp. 1547-1555, 2000.

Garcia, M. A., Larre, A., Lopez, B., and Oller, A., "Reducing the complexity of geometric selective disassembly", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1474-1479, 2000.

Gen, M., and Cheng, R. W., Genetic Algorithms and Engineering Design, New York, Wiley, 1997.

Gottschalk, S., Lin, M. C., and Manocha, D., "OBB-Tree: A hierarchical structure for rapid interference detection", Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, pp. 171-180, 1996.

Guan, Q., Liu, J. H., and Zhong, Y. F., "A concurrent hierarchical evolution approach to assembly process planning", International Journal of Production Research, Vol. 40, pp. 3357-3374, 2002.

Gui, J. K., and Mantyla, M., "Functional understanding of assembly modeling", Computer Aided Design, Vol. 26, No. 6, pp. 435-451, 1994.

Gupta, S., Paredis, C., and Brown, F. P., "Micro Planning for Mechanical Assembly Operations", Proceedings of IEEE International Conference on Robotics and Automation, pp. 239-246, 1998.

Hao, Q., Shen, W., and Wang, L., "Towards a cooperative distributed manufacturing management framework", Computers in Industry, Vol. 56, pp. 71-84, 2005.

Harris, J. W. and Stocker, H., Handbook of Mathematics and Computational Science, Springer, New York, 1998.

Homem De Mello, S. L., and Sanderson, C. A., "A correct and complete algorithm for the generation of mechanical assembly sequences", IEEE Transactions on Robotics and Automation, Vol. 7, pp. 228-240, 1991.

Huang, M. Y., Lin, Y. J., and Xu, H., "A framework for web-based product data management using J2EE", International Journal of Advanced Manufacturing Technology, Vol. 24, pp. 847-852, 2004.

Huang, Q. G., and Mak, L. K., "Design for manufacture and assembly on the Internet", Computers in Industry, Vol. 38, pp. 17-30, 1999.

Huang, Q. G., and Mak, L. K., "Issues in the development and implementation of Web applications for product design and manufacture", International Journal of Computer Integrated Manufacturing, Vol. 14, No. 1, pp. 125-135, 2001.

Huang, Q. G., Lee, W. S., and Mak, L. K., "Web-based product and process data modelling in concurrent design for X", Robotics and Computer-Integrated Manufacturing, Vol. 15, No. 1, pp. 53-63, 1999.

Huang, Q. G., Nie, M., and Mak, L. K., "Web-based failure mode and effect analysis (FMEA)", Computers and Industrial Engineering, Vol. 37, pp. 177-180, 1999.

Huang, S. H., "Automated set-up planning for lathe machining", International Journal of Manufacturing Systems, Vol. 17, No. 3, pp. 196-208, 1998.

Huang, S. H., and Zhang, H., "Tolerance analysis in set-up planning for rotational parts", International Journal of Manufacturing Systems, Vol. 15, No. 5, pp. 340-350, 1996.

ISO (international organization for standardization), http://www.iso.org/iso/en/CombinedQueryResult.CombinedQueryResult?queryString=ISO%2FIEC+14772, 2005.

Jackman, J., and Park, K. D., "Probe orientation for coordinate measuring machine systems using design models", Robotics and Computer-Integrated Manufacturing, Vol. 14, pp. 229-236, 1998.

Koç, M., Ni, J., Lee, J., and Bandyopadhyay, P., "Introduction of e-manufacturing", NAMRC 2003 E-Manufacturing Panel, pp. 1-13, 2003.

Koelsch, R. J., "Work smart", Manufacturing Engineering, pp. 65-67, 1994.

Kong, S., Park, J., Han, Y., Kim, G., and Lee, K., "The Internet-based virtual machining system using CORBA", Integrated Manufacturing Systems, Vol. 13, No. 5, pp. 340-344, 2002.

Kumara, M., and Rajotia, S., "Integration of scheduling with computer aided process planning", Journal of Materials Processing Technology, Vol. 138, pp. 297-300, 2003.

Kuo, C. T., "Disassembly sequence and cost analysis for electromechanical products", Robotics and Computer-Integrated Manufacturing, Vol. 16, pp. 43-54, 2000.

Kuo, C. T., Huang, H. S., and Zhang, C. H., "Design for manufacture and design for 'X': concepts, applications and perspectives", Computers and Industrial Engineering, Vol. 41, pp. 241-260, 2001.

La, J., Bateman, T., and Wild, D., "Design for manufacturing: use of a spreadsheet model of manufacturability to optimize product design and development", Research in Engineering Design, Vol. 14, pp. 107-117, 2003.

Lambert, A. J. D., "Disassembly sequencing: a survey", International Journal of Production Research. Vol. 41, No. 16, 3721-3759, 2003.

Lau, Y. K. H., Mak, L. K., and Lu, T. H. M., "A virtual design platform for interactive product design and visualization", Journal of Materials Processing Technology, Vol. 139, pp. 402-407, 2003.

Lazzerini, B., and Marcelloni, F., "A genetic algorithm for generating optimal assembly plans", Artificial Intelligence in Engineering, Vol. 14, pp. 319-329, 2000.

Lee, S., External Authoring Interface, http://www.cubizen.net/tt/board/ttboard.cgi?act=read&db=eai&page=1&idx=1, 2004.

Lee, J., "E-manufacturing-fundamental, tools, and transformation", Robotics and Computer-Integrated Manufacturing, Vol. 19, No. 6, pp. 501-507, 2003.

Léon, C. J., Rejneri, N., and Debarbouillé, G., "Assembly/disassembly simulation early during a design process", Proceedings of ASME Design Engineering Technical Conference, pp. 1-9, 2001.

Lim, P. C., and Menq, H. C., "CMM feature accessibility and path generation", Robotics and Computer-Integrated Manufacturing, Vol. 32, No. 3, pp. 597-618, 1994.

Limaiem, A., and ElMaraghy, A. H., "A general method for analyzing the accessibility of features using concentric spherical shells", International Journal of Advanced Manufacturing Technology, Vol. 13, pp. 101-108, 1997.

Mascle, C., and Balasoiu, B., "Algorithmic selection of a disassembly sequence of a component by a wave propagation method", Robotics and Computer-Integrated Manufacturing, Vol. 19, pp. 439-448, 2003.

MIL-HDBK-472, Maintainability prediction. Department of Defense (DoD), USA, 1966.

Miller, M. J., and Hoffman, L. R., "Automatic assembly planning with fasteners", Proceedings of IEEE International Conference on Robotics and Automation, pp. 69-74, 1989.

MIL-STD-470B, Maintainability program for systems and equipment, DoD, USA, 1989.

MIL-STD-756B, Reliability Modeling and Prediction, DoD, USA, 1981.

Mo, J., Zhang, Q., and Gadh, R., "Virtual disassembly", International Journal of CAD/CAM, Vol. 2, No. 1, pp. 29-37, 2002.

Moon, C., Lee, M., Seo, Y., and Lee, Y. H., "Integrated machine tool selection and operation sequencing with capacity and precedence constraints using genetic algorithm", Computers and Industrial Engineering, Vol. 43, pp. 605-621, 2002.

Moon, C., and Seo, Y., "Evolutionary algorithm for advanced process planning and scheduling in a multi-plant", Computers and Industrial Engineering, Vol. 48, pp. 311-325, 2005.

Möller, T., "A fast triangle-triangle intersection test", Journal of Graphics Tools, Vol. 2, pp. 25-30, 1997.

Nagalingam, V. S., and Lin, C. I. G., "Latest developments in CIM", Robotics and Computer Integrated Manufacturing, Vol. 15, pp. 423-430, 1999.

NASA-STD-8729.1, Planning, developing and managing an effective reliability and maintainability (R&M) program. National Aeronautics and Space Administration (NASA), USA, 1998.

Noh, D. S., Park, J. Y., Kong, H. S., Han, G. Y., Kim, G., and Lee, I. K., "Concurrent and collaborative process planning for automotive general assembly", International Journal of Advanced Manufacturing Technology, DOI: 10.1007/S00170-004-2092-9, 2004.

Onwubiko, C. O., Introduction to Engineering Design Optimization, New Jersey, Prentice-Hall, 2000.

Ong, K. S., Jiang, L., and Nee, Y. C. A., "An Internet-based virtual CNC milling system", International Journal of Advanced Manufacturing Technology, Vol. 20, pp. 20-30, 2002.

Pak, K. G., and Sodhi, R., "Destructive disassembly of bolts and screws by impact facture", Journal of Manufacturing Systems, Vol. 21, No. 4, pp. 316-324, 2002.

Parallelgraphics Co., http://www.parallelgraphics.com/products/cortona/, 2005.

Peng, Q., "Integrated design and manufacturing strategy: networked virtual manufacturing for SMEs", Proceedings of the 14th International Conference on Flexible Automation and Intelligent Manufacturing, pp. 1218-1225. July 12-14, 2004, Toronto, Canada.

Peng, Q., Hall, R. F., and Lister, M. P., "Application and evaluation of VR-based CAPP system", Journal of Materials Processing Technology, Vol. 107, pp. 153-159, 2000.

Peng, Q., VR and its applications in Design and manufacturing, A Lecture Note, 2002.

Pomares, J., Puente, S. T., Torres, F., Candelas, F. A., and Gil, P., "Virtual disassembly of products based on geometric models", Computers in Industry, Vol. 55, pp. 1-14, 2004.

Pramet co., Lathe turning tool handbook, 2001.

Pramet co., Lathe turning tool catalogue, 2001.

Qin, F. S., Harrison, R., West, A. A., and Wright, K. D., "Development of a novel 3D simulation modelling system for distributed manufacturing", Computers in Industry, Vol. 54, pp. 69-81, 2004.

Qiu, M. Z., Chen, P. Y., Zhou, D. Z., Ong, K. S., and Nee, Y. C. A., "Multi-user NC machining simulation over the WWW", International Journal of Advanced Manufacturing Technology, Vol. 18, pp. 1-6, 2001.

Seo, Y., Kim, D., and Suh, S., "Development of Web-based CAM system", International Journal of Advanced Manufacturing Technology, DOI 10.1007/s00170-004-2422-y, 2005.

Shu, H. L., and Flowers, C. W., "Application of a design-for-remanufacture framework to the selection of product life-cycle fastening and joining methods", Robotics and Computer Integrated Manufacturing, Vol. 15, pp. 179-190, 1999.

Singh, N., System Approach to Computer-Integrated Design and Manufacturing, John wiley & Sons, Inc., New York, USA, 1996.

Smith, G. C., and Smith, S. S. F., "An enhanced genetic algorithm for automated assembly planning", Robotics and Computer Integrated Manufacturing, Vol. 18, pp. 355-364, 2002.

Smith, S. C., and Wright, K. P., "Cyber cut: a world wide web based design-to-fabrication tool", Journal of Manufacturing System, Vol. 15, pp. 432-442, 1996.

Snap-On tools, Web catalog, http://buy1.snapon.com/catalog /catalog.asp?id=1, 2004.

Soromaz, D., and Khosknevis, B., "Machine and tool constraint specification for integrated process planning system", Proceedings of the 7th Industrial Engineering Research Conference, pp. 901-906, 1997.

Spitz, N. S., and Requicha, A. G. A., "Accessibility analysis using computer graphics hardware", IEEE Transactions on Visualization and Computer Graphics, Vol. 6, No. 3, pp. 208-219, 2000.

Spitz, N. S., Spyridi, J. A., and Requicha, A. G. A., "Accessibility analysis for planning of dimensional inspection with coordinate measuring machines", IEEE Transactions on Robotics and Automation, Vol. 15, No. 6, pp. 714-727, 1999.

Spyridi, J. A., and Requicha, A. G. A., "Accessibility analysis for the automatic inspection of mechanical parts by coordinate measuring machines", Proceedings of IEEE International Conference on Robotics and Automation, pp. 1284-1289, 1990.

Srinivasan, H., and Gadh, R., "A geometric algorithm for single selective disassembly using the wave propagation abstraction", Computer Aided Design, Vol. 30, pp. 603-613, 1998.

Srinivasan, H., Figueroa, R., and Gadh, R., "Selective disassembly for virtual prototyping as applied to de-manufacturing", Robotics and Computer-Integrated Manufacturing, Vol. 15, pp. 231-245, 1999.

Subrahmanyam, G., Gunasekaran, A., Arunachalam, S., and Radhakrishnan, P., "Development of a tool database management system", International Journal of Advanced Manufacturing Technology Vol. 15, pp. 562-565, 1999.

Subramaniam, V., Lee, K. D., Ramesh, T., Hong, S. D., and Wong, S. Y., "Machine selection rules in a dynamic job shop", International Journal of Advanced Manufacturing Technology, Vol. 16, pp. 902-908, 2000.

Tang, Y., Zhou, M. C., Zussman, E., and Caudill, R., "Disassembly modelling, planning, and application: a review", Proceedings of IEEE International Conference on Robotics and Automation, pp. 2197-2202, 2000.

Tseng, E. H., and Li, K. R., "A novel means of generating assembly sequences using the connector concept", Journal of Intelligent Manufacturing, Vol. 10, pp. 423-435, 1999.

Tseng, E. H., Li, D. J., and Chang, H. Y., "Connector-based approach to assembly planning using a genetic algorithm", International Journal of Production Research, Vol. 42, No. 11, pp. 2243-2261, 2004.

Usher, M. J., and Fernandes, J. K., "An object-oriented application of tool selection in dynamic process planning", International Journal of Product Research, Vol. 37, No. 13, pp. 2879-2894, 1999.

Vince, J., Virtual Reality System, Addison-Wesley, 1995.

Wang, L., Orban, P., Cunningham, A., and Lang, S., "Remote real-time CNC machining for web-based manufacturing", Robotics and Computer-Integrated Manufacturing, Vol. 20, pp. 563-571, 2004.

Watson, T. R., Data Management (Database and Organization), John Wiley & Son Inc., 1999.

Web3D Org, http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/part2/backComp.html, 2005.

Webster, M., and Sugden, D., "Implementation of virtual manufacturing by a technology licensing company", International Journal of Operations and Production Management, Vol. 23, No. 5, pp. 448-469, 2003.

Wilson, H. R., "Geometric Reasoning about Assembly Tools", Artificial Intelligence, Vol. 98, No. 1-2, pp. 237-279, 1998.

Wilson, W. R. D., Tolerance, fits and finish. Lecture note (ME395 introduction to mechanical design), Http://faculty.washington.edu/wilsonw/ME395/tolfitfin.ppt#1, 2005.

Xu, H. T., Lin, J. Y., and Chan, C. C., "A Web-based product modelling tool-a preliminary development", International Journal of Advanced Manufacturing Technology, Vol. 21, pp. 669-677, 2003.

Xu, W. X., and Liu, T., "A Web-enabled PDM system in a collaborative design environment", Robotics and Computer-Integrated Manufacturing, Vol. 19, pp. 315-328, 2003.

Yang, H., and Xue, D., "Recent research on developing Web-based manufacturing systems: a review", International Journal of Production Research, Vol. 41, No. 15, pp. 3601-3629, 2003.

Yin, P. Z., Ding, H., Li, X. H., and Xiong, L. Y., "A connector-based hierarchical approach to assembly sequence planning for mechanical assemblies", Computer Aided Design, Vol. 35, pp. 37-56, 2003.

Zeng, J., Chen, W., and Ding, Q., "A Web-based CAD system, Journal of Materials Processing Technology", Journal of Materials Processing Technology, Vol. 139, pp. 229-232, 2003.

Zha, F. X., "A web-based advisory system for process and material selection in concurrent product design for a manufacturing environment", DOI: 10.1007/ s00170-003-1838-0, International Journal of Advanced Manufacturing Technology, Vol. 25, pp. 233-243. 2005.

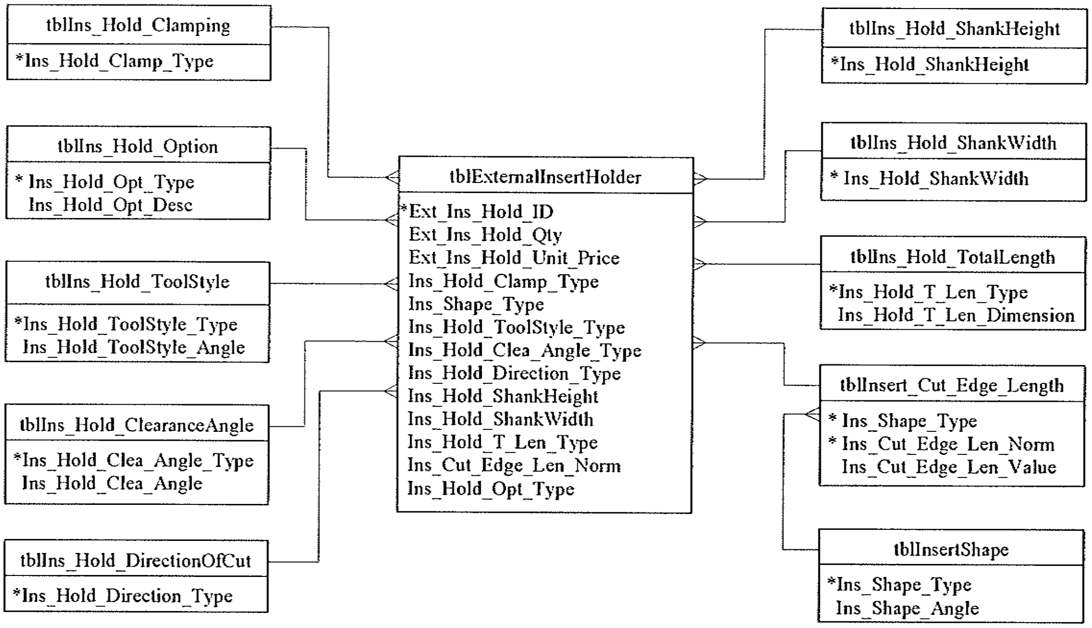
Zhang, S., Shen, W., and Ghenniwa, H., "A review of Internet-based product information sharing and visualization", Computers in Industry, Vol. 54, pp. 1-15, 2004.

Zhao, Z., and Shah, J. J., "Domain independent shell for DFM and its application to sheet metal forming and injection molding", Computer Aided Design, Vol. XX, pp. 1-18, 2005.

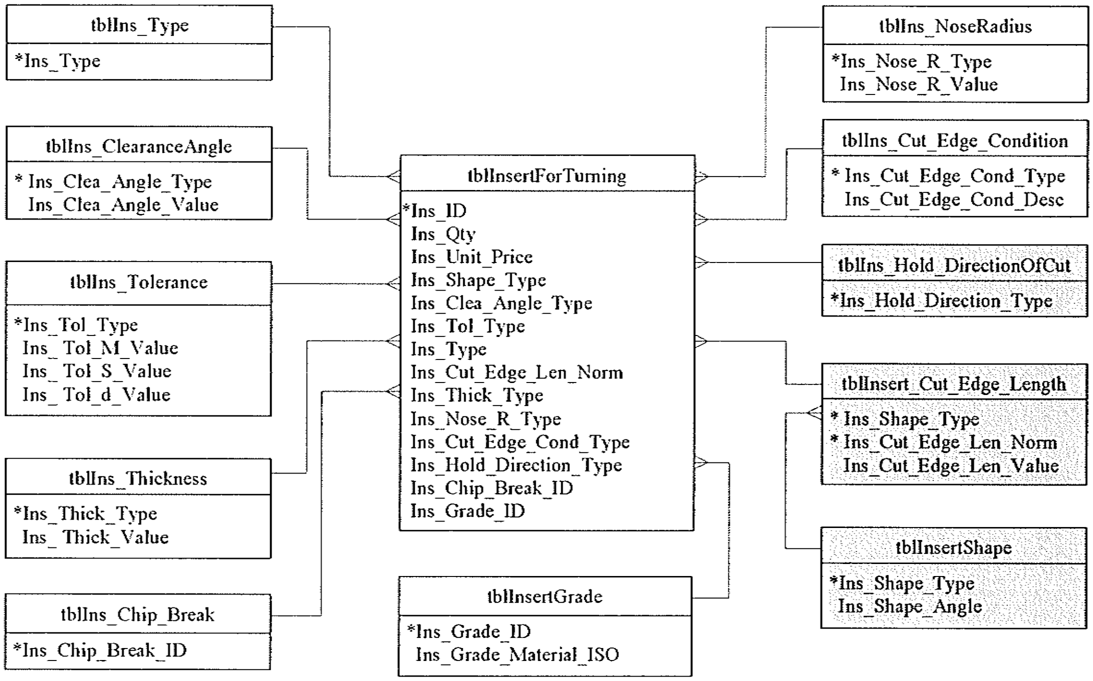
# Appendix

## A.1 Several database models for dynamic process planning

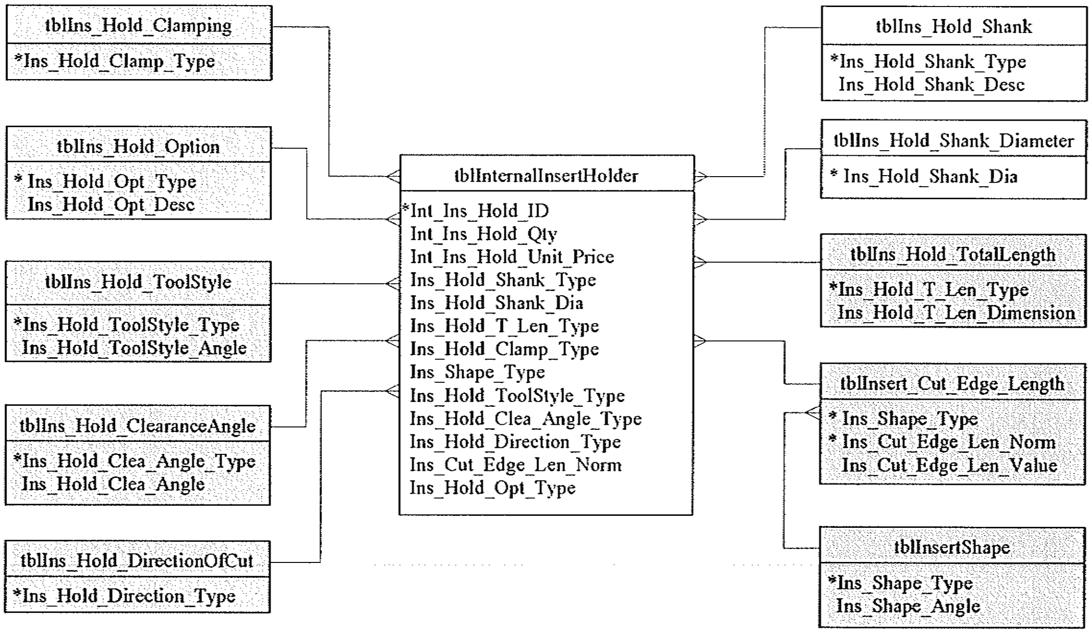
- Data Modelling for External Insert Holder



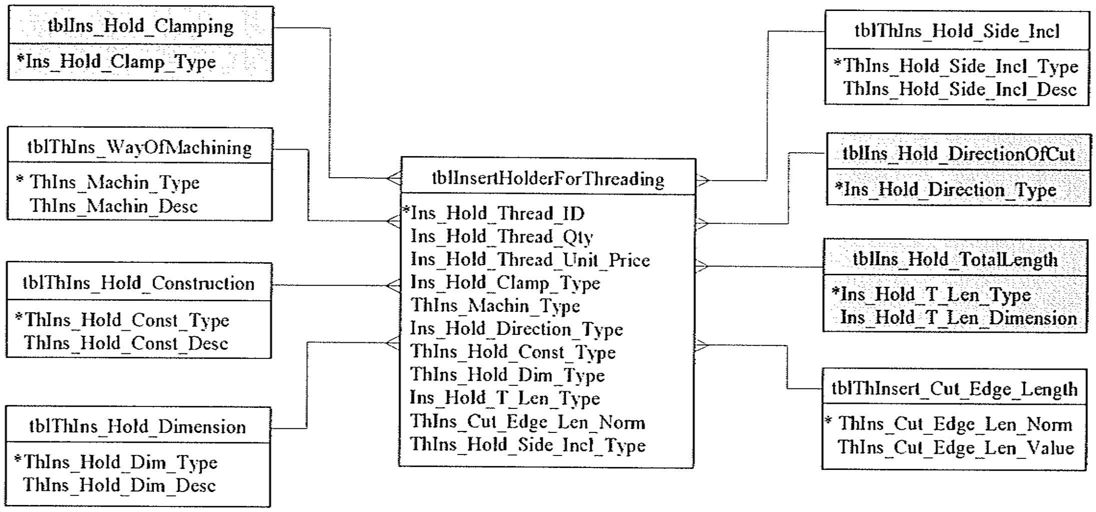
- Data Modelling of Insert for Turning



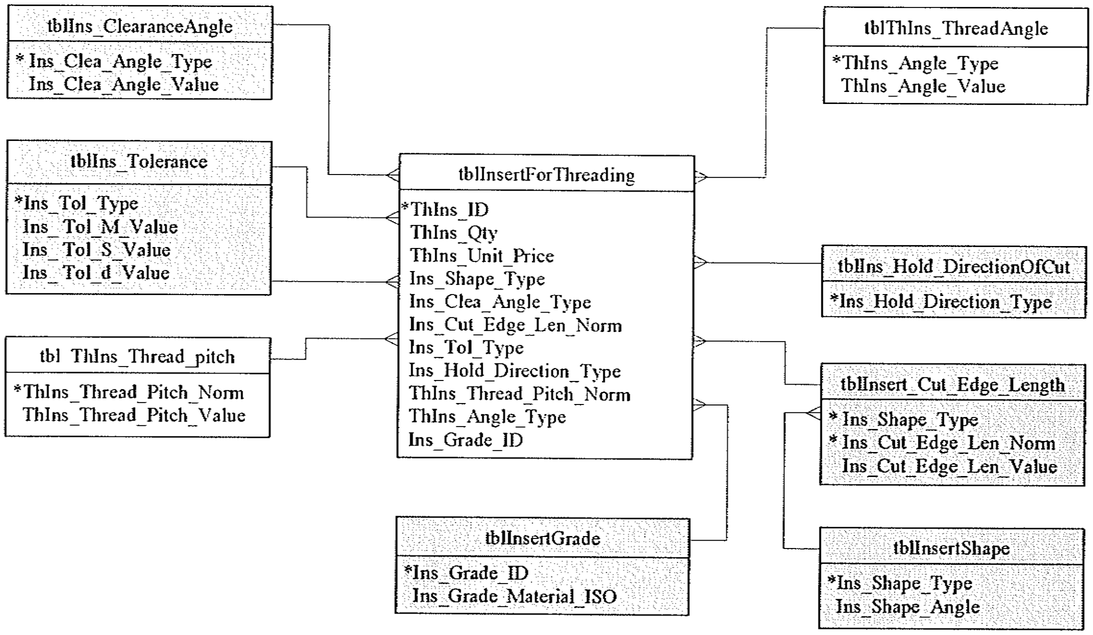
- Data Modelling for Internal Insert Holder



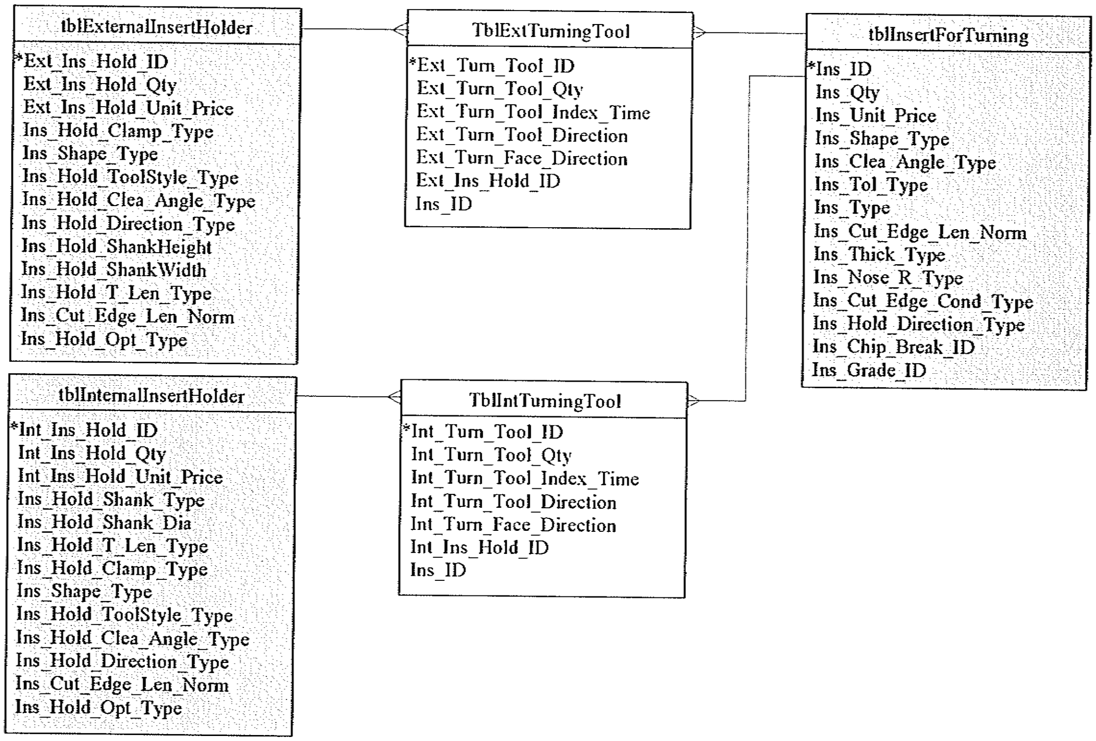
- Data Modelling of Insert Holder for Threading



- Data Modelling of Insert for Threading



- Data Modelling of Turning Tool



tblExternalInsertHolder

*Ext_Ins_Hold_ID
Ext_Ins_Hold_Qty
Ext_Ins_Hold_Unit_Price
Ins_Hold_Clamp_Type
Ins_Shape_Type
Ins_Hold_ToolStyle_Type
Ins_Hold_Clea_Angle_Type
Ins_Hold_Direction_Type
Ins_Hold_ShankHeight
Ins_Hold_ShankWidth
Ins_Hold_T_Len_Type
Ins_Cut_Edge_Len_Norm
Ins_Hold_Opt_Type

TblExtTurningTool

*Ext_Turn_Tool_ID
Ext_Turn_Tool_Qty
Ext_Turn_Tool_Index_Time
Ext_Turn_Tool_Direction
Ext_Turn_Face_Direction
Ext_Ins_Hold_ID
Ins_ID

tblInsertForTurning

*Ins_ID
Ins_Qty
Ins_Unit_Price
Ins_Shape_Type
Ins_Clea_Angle_Type
Ins_Tol_Type
Ins_Type
Ins_Cut_Edge_Len_Norm
Ins_Thick_Type
Ins_Nose_R_Type
Ins_Cut_Edge_Cond_Type
Ins_Hold_Direction_Type
Ins_Chip_Break_ID
Ins_Grade_ID

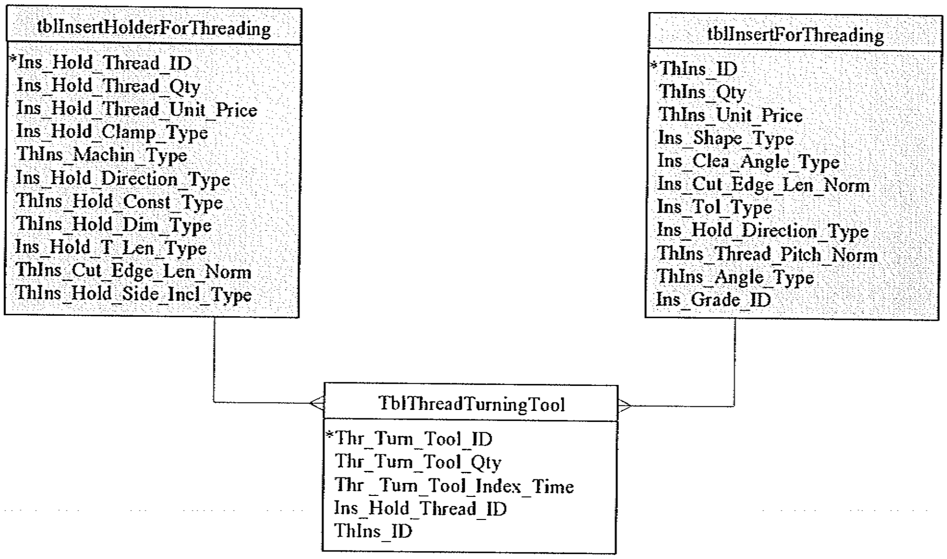
tblInternalInsertHolder

*Int_Ins_Hold_ID
Int_Ins_Hold_Qty
Int_Ins_Hold_Unit_Price
Ins_Hold_Shank_Type
Ins_Hold_Shank_Dia
Ins_Hold_T_Len_Type
Ins_Hold_Clamp_Type
Ins_Shape_Type
Ins_Hold_ToolStyle_Type
Ins_Hold_Clea_Angle_Type
Ins_Hold_Direction_Type
Ins_Cut_Edge_Len_Norm
Ins_Hold_Opt_Type

TblIntTurningTool

*Int_Turn_Tool_ID
Int_Turn_Tool_Qty
Int_Turn_Tool_Index_Time
Int_Turn_Tool_Direction
Int_Turn_Face_Direction
Int_Ins_Hold_ID
Ins_ID

- Data Modelling of Threading Tool



tblInsertHolderForThreading

*Ins_Hold_Thread_ID
Ins_Hold_Thread_Qty
Ins_Hold_Thread_Unit_Price
Ins_Hold_Clamp_Type
ThIns_Machin_Type
Ins_Hold_Direction_Type
ThIns_Hold_Const_Type
ThIns_Hold_Dim_Type
Ins_Hold_T_Len_Type
ThIns_Cut_Edge_Len_Norm
ThIns_Hold_Side_Incl_Type

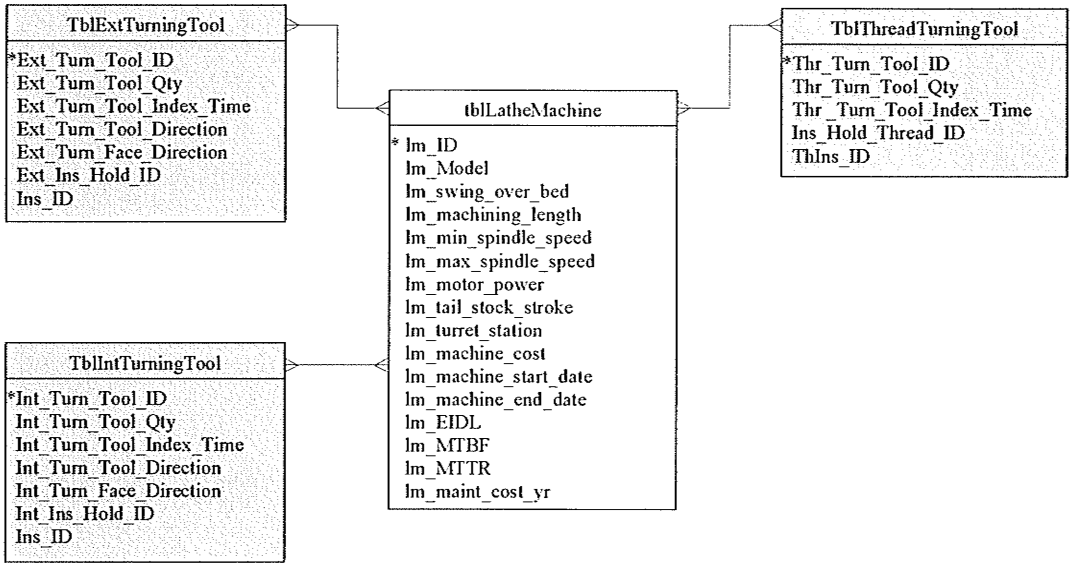
tblInsertForThreading

*ThIns_ID
ThIns_Qty
ThIns_Unit_Price
Ins_Shape_Type
Ins_Clea_Angle_Type
Ins_Cut_Edge_Len_Norm
Ins_Tol_Type
Ins_Hold_Direction_Type
ThIns_Thread_Pitch_Norm
ThIns_Angle_Type
Ins_Grade_ID

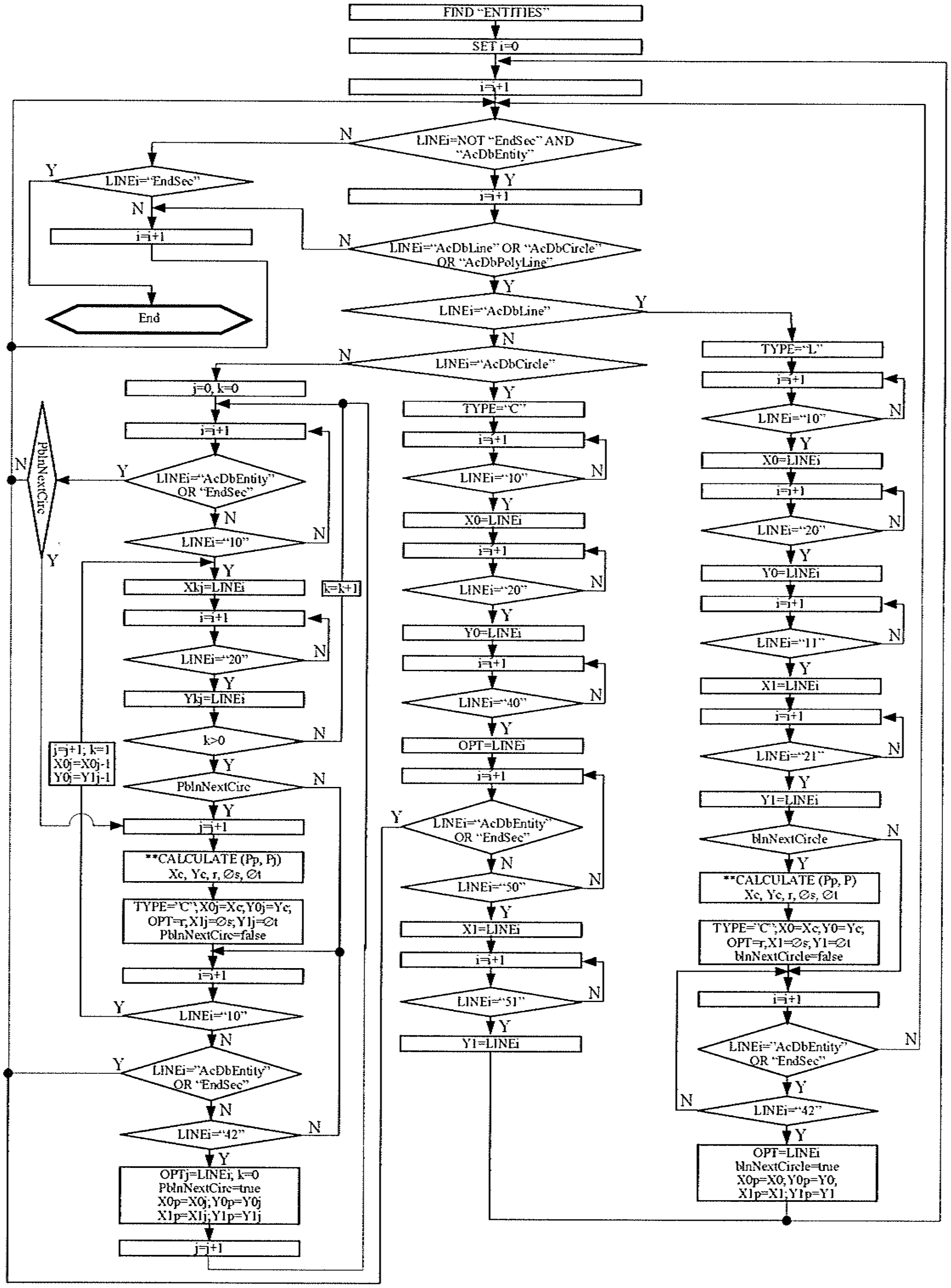
TblThreadTurningTool

*Thr_Turn_Tool_ID
Thr_Turn_Tool_Qty
Thr _Turn_Tool_Index_Time
Ins_Hold_Thread_ID
ThIns_ID

- Data Modelling of NC Lathe Machine

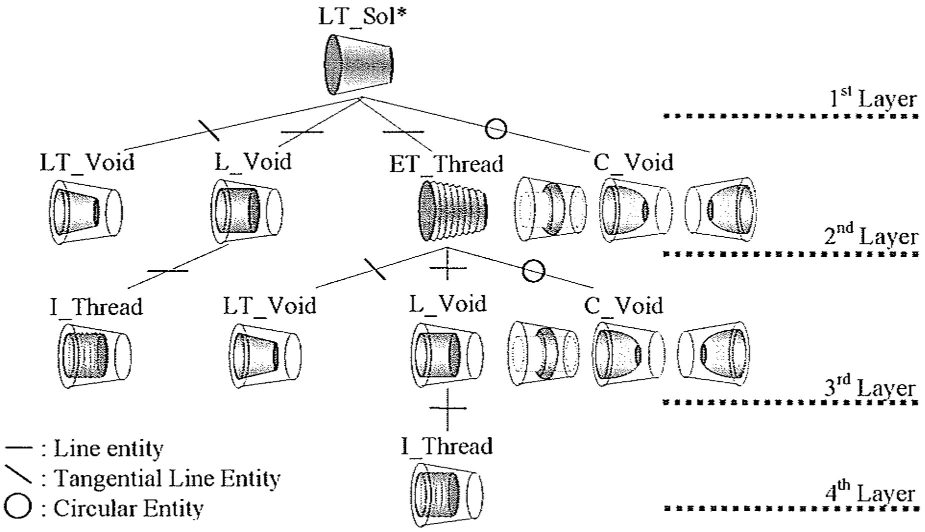


## A.2 Flow chart of obtaining geometry entities from DXF representations

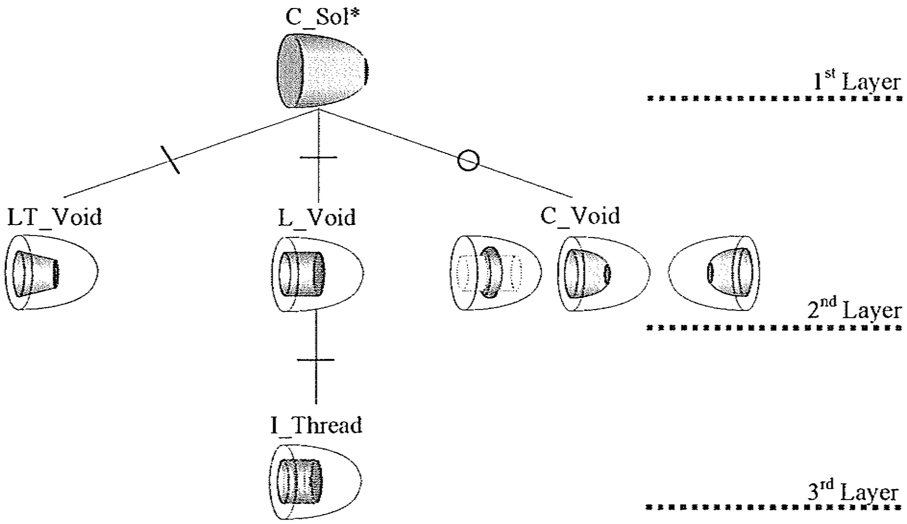


## A.3 Decision making diagrams for feature definition based on obtained geometric entities

- Tangent line entity-based decision making diagram for feature definition



- Circular entity-based decision making diagram for feature definition



## A.4 Structure of ManufFeature Java class

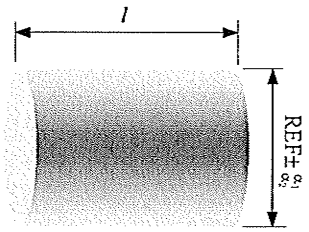
```
Public class ManufFeature implements FeatureManagement {

        /* General information-related properties */
        private int projectID;              //project ID
        private String projectName;         //project Name
        private String dxfID;               //DXF file (drawing) number
        private String partNumber;          //part number
        private String partName;            //part name

        /* Feature information-related properties */
        private int featureID;              //feature ID
        private String featureShape;        //feature shape in the same inheritance
                                            //level
        private float xStart;               //feature position in x-coordinate
        private float[] parameter;          //parameter for a feature defined in this
                                            //dissertation
        private float[][] transf2D;         //transformation matrix of a feature
        private String fillingType;         //'void' or 'solid'
        private MafFace[] faceList;         //generated manufacturing faces of this
                                            //feature
        private int prevFeatureID;          //an adjacent feature at the left side
        private String ingredientList;      //embedded features in this feature

        /* methods */
        public void setProjectID (int pID) {...}
        public void setProjectName (String pName) {...}
        public void setDxfID (String dxfID) {...}

                        ⋮

        public MafFace[] getMafFace () {...}
        public int getPrevFeatureID () {...}
        public int[] getIngredientFeature () {...}
        public MafFace[] getManufacturingFace () {...}

  }
```

## A.5 Several examples of tolerance factors defined in this dissertation

### 1) Dimensional tolerance

$$t = \frac{(\alpha_1 + \alpha_2)}{l} \text{ in which,}$$

$$\begin{pmatrix} \alpha_1 \text{ : demensioning tolerance (upper limit)} \\ \alpha_2 \text{ : demensioning tolerance (lower limit)} \\ l \text{ : machinable area (length)} \end{pmatrix}$$

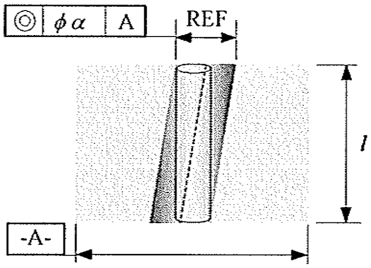


### 2) Geometric tolerance

- Concentricity

$$t = \frac{\alpha}{l} \text{ in which,}$$

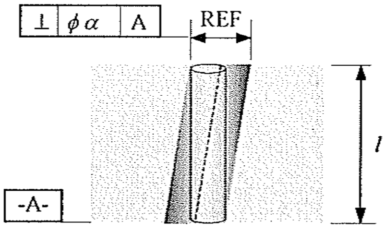
$$\begin{pmatrix} \alpha \text{ : tolerance} \\ l \text{ : machinable area (length)} \end{pmatrix}$$



- Perpendicularity

$$t = \frac{\alpha}{l} \text{ in which,}$$

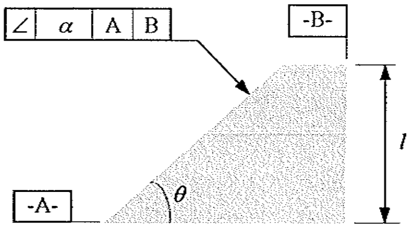
$$\begin{pmatrix} \alpha \text{ : tolerance} \\ l \text{ : machinable area (length)} \end{pmatrix}$$



- Angularity

$$t = \frac{\alpha \sin\theta}{l} \text{ in which,}$$

$$\begin{pmatrix} \alpha \text{ : tolerance} \\ l \text{ : machinable area (length)} \end{pmatrix}$$



## A.6 Set-up planning and sequencing algorithm

### 1) Nomenclatures

| | |
|---|---|
| $n$ | Number of manufacturing faces within a rotational component |
| $f_i$ | Manufacturing faces, $i = 1,2,\cdots,n$ |
| $K_0$ | Set of manufacturing faces that exist on the stock |
| $K_i$ | Set of manufacturing faces that exist after the workpiece was machined in the $i$-th set-up |
| $X$ | Set of manufacturing faces that are suitable for location or clamping |
| $X(K_i)$ | Set of manufacturing faces that are suitable for location or clamping after the workpiece was machined in the $i$-th set-up |
| $C$ | Set of manufacturing faces that are cylindrical surfaces |
| $P$ | Set of manufacturing faces that are plane surfaces |
| $O$ | Set of manufacturing faces that are cone surfaces |
| $A_1$ | Set of manufacturing faces that are can be machined from the left side of the component |
| $A_2$ | Set of manufacturing faces that are can be machined from the right side of the component |
| $A_3$ | Set of manufacturing faces that are can be machined from either the right side or left side |
| $T = [t_{ij}]$ | Adjacency matrix of the tolerance factor graph, $i, j = 1,2,\cdots,n$ |

### 2) Mathematical formulation

Set of manufacturing faces: $F = \{f_1, f_2, \cdots, f_n\}$

Stock geometry vector: $K = \{k_1, k_2, \cdots, k_n\}$

$$k_i = \begin{cases} 1 & \textit{if face } f_i \textit{ exists on the stock} \\ 0 & \textit{otherwise} \qquad i = 1,2,\cdots,n \end{cases}$$

Fixturing vector: $\boldsymbol{X}=\{x_1,x_2,\cdots,x_n\}$

$$x_i=\begin{cases}1 & \textit{if face } f_i \textit{ is suitable for locating or clamping}\\ 0 & \textit{otherwise}\end{cases}$$

$$(i=1,2,\cdots,n)$$

Cylindrical vector: $\boldsymbol{C}=\{c_1,c_2,\cdots,c_n\}$

$$c_i=\begin{cases}1 & \textit{if face } f_i \textit{ is a cylindrical surface}\\ 0 & \textit{otherwise}\end{cases}$$

$$(i=1,2,\cdots,n)$$

Plane vector: $\boldsymbol{P}=\{p_1,p_2,\cdots,p_n\}$

$$p_i=\begin{cases}1 & \textit{if face } f_i \textit{ is a plane surface}\\ 0 & \textit{otherwise}\end{cases}$$

$$(i=1,2,\cdots,n)$$

Cone vector: $\boldsymbol{O}=\{o_1,o_2,\cdots,o_n\}$

$$o_i=\begin{cases}1 & \textit{if face } f_i \textit{ is a cone surface}\\ 0 & \textit{otherwise}\end{cases}$$

$$(i=1,2,\cdots,n)$$

Tool approach vector: $\boldsymbol{A}=\{\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_n\}^T$

$$a_i=\begin{cases}[1,0] & \textit{if } f_i \textit{ can be machined only from the left side}\\ [0,1] & \textit{if } f_i \textit{ can be machined only from the right side}\\ [1,1] & \textit{if } f_i \textit{ can be machined either from the left side}\\ & \quad \textit{or from the right side}\end{cases}$$

Tolerance factor (Relative tolerance):

$$\boldsymbol{T}=[t_{ij}],\quad t=\frac{1}{\sum_{i=1}^{m}\frac{1}{t_i}}$$

### 3) Set-up planning and sequencing algorithm

**Algorithm A.6.1**: Set-up Formation

```
Let $\boldsymbol{S}' = A_1 - A_3, \boldsymbol{S}'' = A_2 - A_3, \boldsymbol{S}^* = A_3$
If $\boldsymbol{S}^* = \phi$   Then
    Exit
Else
    Find $t_{xy} = \min[t_{ij}]$, in which $f_i \in A_3, f_j \notin A_3, t_{ij} \neq 0, i, j = 1,2,\cdots,n$
    If such a $t_{xy}$ is found then
        If $f_y \in A_1$ Then
            Let $\boldsymbol{S}' = \boldsymbol{S}' \cup \{f_x\}$
        Else
            Let $\boldsymbol{S}'' = \boldsymbol{S}'' \cup \{f_x\}$
        Let $\boldsymbol{S}^* = \boldsymbol{S}^* - \{f_x\}$
        Goto statement "If $\boldsymbol{S}^* = \phi$   Then..."
    Else
        Let $a$ and $b$ denote the number of elements in $\boldsymbol{S}'$ and $\boldsymbol{S}''$, respectively
        If $a \geq b$ Then
            Let $\boldsymbol{S}' = \boldsymbol{S}' \cup \boldsymbol{S}^*, \boldsymbol{S}^* = \phi$,   Exit
        Else
            Let $\boldsymbol{S}'' = \boldsymbol{S}'' \cup \boldsymbol{S}^*, \boldsymbol{S}^* = \phi$,   Exit
```

**Algorithm A.6.2**: Datum selection-clamping position (c′)

```
Let $\boldsymbol{H} = \boldsymbol{C} \cap \boldsymbol{X} \cap \boldsymbol{S}''$
If $\boldsymbol{H} = \phi$   Then
    Let $\boldsymbol{H} = \boldsymbol{O} \cap \boldsymbol{X} \cap \boldsymbol{S}''$
    If $\boldsymbol{H} \neq \phi$   Then
        Goto statement "Find $t_{xy} = \min[t_{ij}]$..."
    Else
        Find $f_x$ such that $\sum_{j=1}^{n} t_{xy} = 0$, in which $f_x \in \boldsymbol{S}' \cap A_3 \cap \boldsymbol{C} \cap \boldsymbol{X}, f_j \in \boldsymbol{S}'$
        If such a $f_x$ is found then
            Let $\boldsymbol{S}' = \boldsymbol{S}' - \{f_x\}, \boldsymbol{S}'' = \boldsymbol{S}'' + \{f_x\}$
            Let $c' = f_x$, Exit
```

**Algorithm A.6.2**: Datum selection-clamping position (c′) (*Cont'*)

```
          Else
               Let $q(f_i) = \min[t_{ij}], \forall f_i \in \boldsymbol{S}' \cap A_3 \cap \boldsymbol{C} \cap \boldsymbol{X}$, in which $f_j \in \boldsymbol{S}', t_{ij} \neq 0$
               Find $q(f_x) = \max[\, q(f_i)]$
               Let $\boldsymbol{S}' = \boldsymbol{S}' - \{f_x\}, \boldsymbol{S}'' = \boldsymbol{S}'' + \{f_x\}$
               Let $c' = f_x$, Exit
Else
     Find $t_{xy} = \min[t_{ij}]$, in which $f_i \in \boldsymbol{H}, f_j \in \boldsymbol{S}', t_{ij} \neq 0; i, j = 1,2,\cdots,n$
     If such a $t_{xy}$ is found then
          Let $c' = f_x$, Exit
     Else
          Let $c' = f_u$, in which $f_i \in \boldsymbol{H}$ and its diameter is the largest, Exit
```

**Algorithm A.6.3**: Datum selection-stopping position (p′)

```
Let $\boldsymbol{H} = \boldsymbol{P} \cap \boldsymbol{X} \cap \boldsymbol{S}''$
If $\boldsymbol{H} \neq \phi$  Then
     Find $t_{xy} = \min[t_{ij}]$, in which $f_i \in \boldsymbol{H}, f_j \in \boldsymbol{S}', t_{ij} \neq 0; i, j = 1,2,\cdots,n$
     If such a $t_{xy}$ is found then
          Let $p' = f_x$, Exit
```

**Algorithm A.6.4**: Set-up Sequencing

```
Let $d(f_i) = \sum_{j=1}^{n} w_{ij}$,     $\forall f_i \in \{c', p', c'', p''\}$, in which
     $w_{ij} = \begin{cases} 1 & \text{if } t_{ij} \neq 0, \{f_i, f_j\} \not\subset \boldsymbol{S}', \{f_i, f_j\} \not\subset \boldsymbol{S}'' \\ 0 & \text{otherwise} \end{cases}$
Find $d(f_x) = \max[d(f_i)]$, in which $f_i \in \{c', p', c'', p''\}$
If $f_x \in \{c'', p''\}$ then
     Let $\boldsymbol{S}_1 = \boldsymbol{S}', c_1 = c', p_1 = p'$
     Let $\boldsymbol{S}_2 = \boldsymbol{S}'', c_2 = c'', p_2 = p''$
Else
     Let $\boldsymbol{S}_1 = \boldsymbol{S}'', c_1 = c'', p_1 = p''$
     Let $\boldsymbol{S}_2 = \boldsymbol{S}', c_2 = c', p_2 = p'$
```

**Algorithm A.6.4**: Set-up Sequencing (*Cont'*)

```
If $c_2 \notin X(K_1)$ or $p_2 \notin X(K_1)$ then
    Swap $\boldsymbol{S}_1$ with $\boldsymbol{S}_2$, $c_1$ with $c_2$, and $c_1$ with $c_2$
If $c_1 \notin \boldsymbol{K}_0 \cap \boldsymbol{X}$ then
    Let $\boldsymbol{H}_c = \boldsymbol{C} \cap \boldsymbol{X} \cap \boldsymbol{S}_2 \cap \boldsymbol{K}_0$
    If $\boldsymbol{H}_c \neq \phi$ then
        Let $c_1 = f_x$, in which $f_x \in \boldsymbol{H}_c$ and its diameter is the largest
    Else
        Let $\boldsymbol{S}_3 = \boldsymbol{S}_2$, $c_3 = c_2$, $p_3 = p_2$
        Let $\boldsymbol{S}_2 = \boldsymbol{S}_1$, $c_2 = c_1$, $p_2 = p_1$
        Let $\boldsymbol{S}_1 = \{c_2\}$
        Let $c_1 = f_x$, in which $f_x \in \boldsymbol{C} \cap \boldsymbol{X} \cap \boldsymbol{K}_0$ and it is practical for machining $\boldsymbol{S}_1$
        Let $p_1 = f_y$, in which $f_y \in \boldsymbol{P} \cap \boldsymbol{X} \cap \boldsymbol{K}_0$, $\{f_y, p_2\} \not\subset A_1$, $\{f_y, p_2\} \not\subset A_2$
If $p_1 \notin \boldsymbol{K}_0 \cap \boldsymbol{X}$ then
    Let $p_1 = f_u$, in which $f_u \in \boldsymbol{P} \cap \boldsymbol{X} \cap \boldsymbol{S}_2 \cap \boldsymbol{K}_0$ and it is an end face
```

## A.7 Tolerance-grade conversion table and the corresponding machining processes

### 1) IT-numbers based on tolerance coefficients

| Grade | Tolerance coefficient | Grade | Tolerance coefficient |
|---|---|---|---|
| IT5 | 7 | IT11 | 100 |
| IT6 | 10 | IT12 | 160 |
| IT7 | 16 | IT13 | 250 |
| IT8 | 25 | IT14 | 400 |
| IT9 | 40 | IT15 | 640 |
| IT10 | 64 | IT16 | 1000 |

### 2) Machining processes corresponding to IT-numbers

| Machining process | Tolerance grade (IT-number) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | IT4 | IT5 | IT6 | IT7 | IT8 | IT9 | IT10 | IT11 | ... |
| Lapping and honing | ⊕ | ⊕ | | | | | | | |
| Cylindrical grinding | | ⊕ | ⊕ | ⊕ | | | | | |
| Surface grinding | | ⊕ | ⊕ | ⊕ | ⊕ | | | | |
| Diamond turning and boring | | ⊕ | ⊕ | ⊕ | | | | | |
| Broaching | | ⊕ | ⊕ | ⊕ | ⊕ | | | | |
| Reaming | | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | | |
| Fine turning | | | | ⊕ | ⊕ | | | | |
| Finishing turning | | | | | ⊕ | ⊕ | | | |
| Semi-rough turning | | | | | | ⊕ | ⊕ | | |
| Rough turning | | | | | | | ⊕ | ⊕ | |
| Milling | | | | | | | ⊕ | ⊕ | |
| ⋮ | | | | | | | | | |

## A.8 Summary of knowledge tables for calculating machining parameters

### 1) Material classifications according to ISO 513

| Material | Description |
|---|---|
| $P_{I}$ | Carbon steels non alloyed<br>Carbon cast steels<br>Carbon tool steels<br>Low alloyed steels |
| $P_{II}$ | Alloyed and medium alloyed steels<br>Low and medium alloyed steels<br>Alloyed tool steels<br>Ferritic and martensitic corrosion-resistant steels |
| $M_{I}$ | Austenitic and Ferritic-Austenitic corrosion-resistant, heat-resistant and creep-resistant steels<br>Nonmagnetic and abrasive resistant steels |
| $M_{II}$ | Special creep-resistant Ni, Co, Fe, and Ti based alloys |
| $M_{III}$ | Heat-treated steels with hardness 48 to 60 HRC<br>Hardened ingot-mould iron with hardness 55 to 85 HSH |
| $K_{I}$ | Grey cast iron alloyed and non alloyed<br>Nodular cast iron<br>Malleable cast iron |
| $K_{II}$ | Non-ferrous metals<br>Al alloys<br>Cu alloys |

### 2) Insert grades according to ISO 513

| Chemical vapour deposition (CVD) | | | | | Physical vapour deposition (PVD) | | Uncoated | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 320P | 210K | 525P | 530P | 535P | 816 | 836 | S10 | S20 | HF7 | HF10 |
| TiC/TiCN/TiN Multi-Layer Coated | | | | | TiN Coated | | | | | |

### 3) Basic cutting speed in turning

| M | Turning | | Tool Materials/cutting speed, $v_{c15}$ (m•min$^{-1}$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Grade<br>Type | 320P | 210K | 525P | 530P | 535P | 816 | S10 | S20 | 836 | HF7 | HF10 |
| $P_I$ | Fine turning | **S** | 340 | 290 | 280 | - | - | 220 | 210 | 180 | - | - | - |
| | | **C, W** | 340 | 290 | 280 | - | - | 220 | 210 | 180 | - | - | - |
| | | **T** | 340 | 290 | 260 | - | - | 220 | 190 | 170 | - | - | - |
| | | **D** | 330 | 270 | 260 | - | - | 210 | 190 | 170 | - | - | - |
| | | **V** | 330 | 270 | 260 | - | - | 200 | 190 | 160 | - | - | - |
| | | **R** | 340 | 290 | 280 | - | - | 220 | 210 | 180 | - | - | - |
| | Finishing | **S** | 300 | 250 | 245 | 240 | - | - | - | - | - | - | - |
| | | **C, W** | 300 | 250 | 245 | 240 | - | - | - | - | - | - | - |
| | | **T** | 290 | 240 | 230 | 225 | - | - | - | - | - | - | - |
| | | **D** | 280 | 240 | 230 | 225 | - | - | - | - | - | - | - |
| | | **V** | 290 | 230 | 220 | 210 | - | - | - | - | - | - | - |
| | | **R** | 300 | 250 | 245 | 240 | - | - | - | - | - | - | - |
| | Semi roughing | **S** | 235 | 195 | 180 | 175 | - | - | - | - | - | - | - |
| | | **C, W** | 235 | 195 | 180 | 175 | - | - | - | - | - | - | - |
| | | **T** | 225 | 185 | 170 | 165 | - | - | - | - | - | - | - |
| | | **D** | 225 | 185 | 170 | 165 | - | - | - | - | - | - | - |
| | | **V** | 215 | 175 | 160 | 155 | - | - | - | - | - | - | - |
| | | **R** | 235 | 195 | 180 | 175 | - | - | - | - | - | - | - |
| | Roughing | **S** | 165 | - | 135 | 130 | 120 | - | - | - | - | - | - |
| | | **C, W** | 165 | - | 135 | 130 | 120 | - | - | - | - | - | - |
| | | **T** | 155 | - | 125 | 125 | 110 | - | - | - | - | - | - |
| | | **D** | 155 | - | 125 | 125 | 110 | - | - | - | - | - | - |
| | | **V** | 155 | - | 125 | 125 | 110 | - | - | - | - | - | - |
| | | **R** | 165 | - | 135 | 130 | 120 | - | - | - | - | - | - |
| $P_{II}$ | Fine turning | **S** | 260 | 225 | 210 | - | - | 165 | 155 | 135 | - | - | - |
| | | **C, W** | 260 | 225 | 210 | - | - | 165 | 155 | 135 | - | - | - |
| | | **T** | 260 | 225 | 195 | - | - | 165 | 140 | 125 | - | - | - |
| | | **D** | 240 | 210 | 195 | - | - | 155 | 140 | 125 | - | - | - |
| | | **V** | 240 | 210 | 195 | - | - | 150 | 130 | 115 | - | - | - |
| | | **R** | 260 | 225 | 210 | - | - | 165 | 155 | 135 | - | - | - |
| | Finishing | **S** | 230 | 190 | 180 | 175 | - | - | - | - | - | - | - |
| | | **C, W** | 230 | 190 | 180 | 175 | - | - | - | - | - | - | - |
| | | **T** | 220 | 180 | 170 | 165 | - | - | - | - | - | - | - |
| | | **D** | 220 | 180 | 170 | 165 | - | - | - | - | - | - | - |
| | | **V** | 220 | 180 | 165 | 160 | - | - | - | - | - | - | - |
| | | **R** | 230 | 190 | 180 | 175 | - | - | - | - | - | - | - |

| M | Turning | Grade<br>Type | Tool Materials/cutting speed, $v_{c15}$ (m•min$^{-1}$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 320P | 210K | 525P | 530P | 535P | 816 | S10 | S20 | 836 | HF7 | HF10 |
| $P_{II}$ | Semi roughing | **S** | 175 | 145 | 135 | 130 | - | - | - | - | - | - | - |
| | | **C, W** | 175 | 145 | 135 | 130 | - | - | - | - | - | - | - |
| | | **T** | 170 | 140 | 130 | 125 | - | - | - | - | - | - | - |
| | | **D** | 170 | 140 | 130 | 125 | - | - | - | - | - | - | - |
| | | **V** | 160 | 130 | 120 | 115 | - | - | - | - | - | - | - |
| | | **R** | 175 | 145 | 135 | 130 | - | - | - | - | - | - | - |
| | Roughing | **S** | 125 | - | 100 | 95 | 90 | - | - | - | - | - | - |
| | | **C, W** | 125 | - | 100 | 95 | 90 | - | - | - | - | - | - |
| | | **T** | 115 | - | 95 | 90 | 85 | - | - | - | - | - | - |
| | | **D** | 115 | - | 95 | 90 | 85 | - | - | - | - | - | - |
| | | **V** | 115 | - | 95 | 90 | 85 | - | - | - | - | - | - |
| | | **R** | 125 | - | 100 | 95 | 90 | - | - | - | - | - | - |
| $M_I$ | Fine turning | **S** | 230 | - | - | 180 | - | 100 | - | - | 80 | 60 | 45 |
| | | **C, W** | 230 | - | - | 180 | - | 100 | - | - | 80 | 60 | 45 |
| | | **T** | 210 | - | - | 170 | - | 85 | - | - | 80 | 60 | 45 |
| | | **D** | 210 | - | - | 170 | - | 85 | - | - | 70 | 55 | 40 |
| | | **V** | 200 | - | - | 160 | - | 80 | - | - | 70 | 55 | 40 |
| | | **R** | 230 | - | - | 180 | - | 100 | - | - | 80 | 60 | 45 |
| | Finishing | **S** | - | 160 | - | 140 | - | 100 | - | - | 80 | - | - |
| | | **C, W** | - | 160 | - | 140 | - | 100 | - | - | 80 | - | - |
| | | **T** | - | 155 | - | 135 | - | 90 | - | - | 75 | - | - |
| | | **D** | - | 150 | - | 135 | - | 90 | - | - | 75 | - | - |
| | | **V** | - | 150 | - | 130 | - | 90 | - | - | 70 | - | - |
| | | **R** | - | 160 | - | 140 | - | 100 | - | - | 80 | - | - |
| | Semi roughing | **S** | - | 120 | 110 | 105 | - | 85 | - | - | 55 | - | - |
| | | **C, W** | - | 120 | 110 | 105 | - | 85 | - | - | 55 | - | - |
| | | **T** | - | 115 | 100 | 95 | - | 80 | - | - | 50 | - | - |
| | | **D** | - | 115 | 100 | 95 | - | 80 | - | - | 50 | - | - |
| | | **V** | - | 105 | 95 | 90 | - | 75 | - | - | 45 | - | - |
| | | **R** | - | 120 | 110 | 105 | - | 85 | - | - | 55 | - | - |
| | Roughing | **S** | - | - | - | 80 | - | 50 | - | - | 43 | - | - |
| | | **C, W** | - | - | - | 80 | - | 50 | - | - | 43 | - | - |
| | | **T** | - | - | - | 75 | - | 45 | - | - | 38 | - | - |
| | | **D** | - | - | - | 75 | - | 45 | - | - | 38 | - | - |
| | | **V** | - | - | - | 75 | - | 45 | - | - | 38 | - | - |
| | | **R** | - | - | - | 80 | - | 50 | - | - | 43 | - | - |

| M | Turning | | Tool Materials/cutting speed, $v_{c15}$ (m•min$^{-1}$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Grade<br>Type | 320P | 210K | 525P | 530P | 535P | 816 | S10 | S20 | 836 | HF7 | HF10 |
| $M_{II}$ | Finishing | **S** | - | - | - | - | - | 50 | - | - | 45 | 40 | 35 |
| | | **C, W** | - | - | - | - | - | 50 | - | - | 45 | 40 | 35 |
| | | **T** | - | - | - | - | - | 45 | - | - | 40 | 35 | 30 |
| | | **D** | - | - | - | - | - | 45 | - | - | 40 | 35 | 30 |
| | | **V** | - | - | - | - | - | 40 | - | - | 35 | 30 | 25 |
| | | **R** | - | - | - | - | - | 50 | - | - | 45 | 40 | 35 |
| | Semi roughing | **S** | - | - | - | - | - | 35 | - | - | 30 | 27 | 20 |
| | | **C, W** | - | - | - | - | - | 35 | - | - | 30 | 27 | 20 |
| | | **T** | - | - | - | - | - | 30 | - | - | 25 | 20 | 18 |
| | | **D** | - | - | - | - | - | 30 | - | - | 25 | 20 | 18 |
| | | **V** | - | - | - | - | - | 25 | - | - | 20 | 15 | 12 |
| | | **R** | - | - | - | - | - | 35 | - | - | 30 | 27 | 20 |
| | Roughing | **S** | - | - | - | - | - | 30 | - | - | 25 | 20 | 18 |
| | | **C, W** | - | - | - | - | - | 30 | - | - | 25 | 20 | 18 |
| | | **T** | - | - | - | - | - | 25 | - | - | 20 | 18 | 15 |
| | | **D** | - | - | - | - | - | 25 | - | - | 20 | 18 | 15 |
| | | **V** | - | - | - | - | - | 25 | - | - | 20 | 18 | 15 |
| | | **R** | - | - | - | - | - | 30 | - | - | 25 | 20 | 18 |
| $M_{III}$ | Finishing | **S** | - | 50 | - | - | - | 50 | - | - | 35 | 45 | 30 |
| | | **C, W** | - | 50 | - | - | - | 50 | - | - | 35 | 45 | 30 |
| | | **T** | - | 45 | - | - | - | 45 | - | - | 35 | 40 | 20 |
| | | **D** | - | 45 | - | - | - | 45 | - | - | 35 | 40 | 20 |
| | | **V** | - | 40 | - | - | - | 40 | - | - | 30 | 35 | 20 |
| | | **R** | - | 50 | - | - | - | 50 | - | - | 35 | 45 | 30 |
| | Semi roughing | **S** | - | 35 | - | - | - | 35 | - | - | 22 | 30 | 18 |
| | | **C, W** | - | 35 | - | - | - | 35 | - | - | 22 | 30 | 18 |
| | | **T** | - | 30 | - | - | - | 30 | - | - | 18 | 25 | 15 |
| | | **D** | - | 30 | - | - | - | 30 | - | - | 18 | 25 | 15 |
| | | **V** | - | 25 | - | - | - | 25 | - | - | 15 | 20 | 10 |
| | | **R** | - | 35 | - | - | - | 35 | - | - | 22 | 30 | 18 |
| $K_I$ | Fine turning | **S** | - | 250 | - | 210 | - | 170 | - | - | 145 | 140 | 125 |
| | | **C, W** | - | 250 | - | 210 | - | 170 | - | - | 145 | 140 | 125 |
| | | **T** | - | 230 | - | 200 | - | 160 | - | - | 135 | 135 | 120 |
| | | **D** | - | 230 | - | 200 | - | 160 | - | - | 135 | 135 | 120 |
| | | **V** | - | 225 | - | 195 | - | 155 | - | - | 130 | 130 | 115 |
| | | **R** | - | 250 | - | 210 | - | 170 | - | - | 145 | 140 | 125 |

| M | Turning | Grade<br>Type | Tool Materials/cutting speed, $v_{c15}$ (m•min$^{-1}$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 320P | 210K | 525P | 530P | 535P | 816 | S10 | S20 | 836 | HF7 | HF10 |
| K$_I$ | Finishing | **S** | - | 220 | - | 185 | - | 145 | - | - | 125 | - | - |
| | | **C, W** | - | 220 | - | 185 | - | 145 | - | - | 125 | - | - |
| | | **T** | - | 210 | - | 175 | - | 135 | - | - | 115 | - | - |
| | | **D** | - | 210 | - | 175 | - | 135 | - | - | 115 | - | - |
| | | **V** | - | 200 | - | 160 | - | 125 | - | - | 105 | - | - |
| | | **R** | - | 220 | - | 185 | - | 145 | - | - | 125 | - | - |
| | Semi roughing | **S** | - | 175 | - | 150 | - | 120 | - | - | 105 | - | - |
| | | **C, W** | - | 175 | - | 150 | - | 120 | - | - | 105 | - | - |
| | | **T** | - | 165 | - | 140 | - | 110 | - | - | 95 | - | - |
| | | **D** | - | 165 | - | 140 | - | 110 | - | - | 95 | - | - |
| | | **V** | - | 155 | - | 130 | - | 100 | - | - | 85 | - | - |
| | | **R** | - | 175 | - | 150 | - | 120 | - | - | 105 | - | - |
| | Roughing | **S** | - | 130 | - | 115 | - | 90 | - | - | 80 | - | - |
| | | **C, W** | - | 130 | - | 115 | - | 90 | - | - | 80 | - | - |
| | | **T** | - | 120 | - | 105 | - | 80 | - | - | 70 | - | - |
| | | **D** | - | 120 | - | 105 | - | 80 | - | - | 70 | - | - |
| | | **R** | - | 130 | - | 115 | - | 90 | - | - | 80 | - | - |
| K$_{II}$<br>(Al)<br>HB<br>100 | Finishing | **S** | - | - | - | - | - | 800 | - | - | - | 680 | - |
| | | **C, W** | - | - | - | - | - | 800 | - | - | - | 680 | - |
| | | **T** | - | - | - | - | - | 800 | - | - | - | 680 | - |
| | | **D** | - | - | - | - | - | 750 | - | - | - | 600 | - |
| | | **V** | - | - | - | - | - | 700 | - | - | - | 550 | - |
| | | **R** | - | - | - | - | - | 800 | - | - | - | 680 | - |
| | Semi roughing | **S** | - | - | - | - | - | 600 | - | - | - | 480 | - |
| | | **C, W** | - | - | - | - | - | 600 | - | - | - | 480 | - |
| | | **T** | - | - | - | - | - | 600 | - | - | - | 480 | - |
| | | **D** | - | - | - | - | - | 550 | - | - | - | 450 | - |
| | | **V** | - | - | - | - | - | 500 | - | - | - | 400 | - |
| | | **R** | - | - | - | - | - | 600 | - | - | - | 480 | - |
| | Roughing | **S** | - | - | - | - | - | 400 | - | - | - | 350 | - |
| | | **C, W** | - | - | - | - | - | 400 | - | - | - | 350 | - |
| | | **T** | - | - | - | - | - | 400 | - | - | - | 350 | - |
| | | **D** | - | - | - | - | - | 350 | - | - | - | 320 | - |
| | | **V** | - | - | - | - | - | 300 | - | - | - | 280 | - |
| | | **R** | - | - | - | - | - | 400 | - | - | - | 350 | - |

| M | Turning | | Tool Materials/cutting speed, $v_{c15}$ (m•min$^{-1}$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Grade<br>Type | 320P | 210K | 525P | 530P | 535P | 816 | S10 | S20 | 836 | HF7 | HF10 |
| K$_{II}$<br>(Cu)<br><br>HB<br>90 | Finishing | **S** | - | - | - | - | - | 450 | - | - | - | 360 | - |
| | | **C, W** | - | - | - | - | - | 450 | - | - | - | 360 | - |
| | | **T** | - | - | - | - | - | 450 | - | - | - | 360 | - |
| | | **D** | - | - | - | - | - | 400 | - | - | - | 320 | - |
| | | **V** | - | - | - | - | - | 350 | - | - | - | 300 | - |
| | | **R** | - | - | - | - | - | 450 | - | - | - | 360 | - |
| | Semi roughing | **S** | - | - | - | - | - | 350 | - | - | - | 300 | - |
| | | **C, W** | - | - | - | - | - | 350 | - | - | - | 300 | - |
| | | **T** | - | - | - | - | - | 350 | - | - | - | 300 | - |
| | | **D** | - | - | - | - | - | 320 | - | - | - | 250 | - |
| | | **V** | - | - | - | - | - | 300 | - | - | - | 200 | - |
| | | **R** | - | - | - | - | - | 350 | - | - | - | 300 | - |
| | Roughing | **S** | - | - | - | - | - | 300 | - | - | - | 270 | - |
| | | **C, W** | - | - | - | - | - | 300 | - | - | - | 270 | - |
| | | **T** | - | - | - | - | - | 300 | - | - | - | 270 | - |
| | | **D** | - | - | - | - | - | 280 | - | - | - | 250 | - |
| | | **V** | - | - | - | - | - | 250 | - | - | - | 230 | - |
| | | **R** | - | - | - | - | - | 300 | - | - | - | 270 | - |

**4) Feed rate and cutting depth in turning**

| Material | Feed rate, $f_r$ (mm•rev$^{-1}$) | | | | Cutting depth, $a_p$ (mm) | | | |
|---|---|---|---|---|---|---|---|---|
| | Fine turning | Finishing | Semi roughing | Roughing | Fine turning | Finishing | Semi roughing | Roughing |
| P$_{I, II}$ | 0.05–0.1 | 0.1-0.2 | 0.2-0.4 | 0.4-0.8 | 0.2–1.0 | 0.8-2.0 | 1.5-4.0 | 4.0-10.0 |
| M$_I$ | 0.05–0.1 | 0.1-0.2 | 0.2-0.4 | 0.4-0.8 | 0.2–1.0 | 0.8-2.0 | 1.5-4.0 | 4.0-10.0 |
| M$_{II}$ | - | 0.1-0.2 | 0.2-0.3 | 0.3-0.4 | - | 0.05-1.5 | 1.5-2.5 | 2.5-3.5 |
| M$_{III}$ | - | 0.08-0.2 | 0.2-0.3 | - | - | 0.8-1.5` | 1.5-2.5 | - |
| K$_I$ | 0.05–0.1 | 0.1-0.2 | 0.2-0.4 | 0.4-0.8 | 0.2–1.0 | 0.8-2.0 | 1.5-4.0 | 4.0-10.0 |
| K$_{II}$ | - | 0.1-0.2 | 0.2-0.4 | 0.4-0.8 | - | 0.8-2.0 | 1.5-4.0 | 4.0-10.0 |

### 5) Durability correction factor (P, M, K Types)

| $T_{\min}$ | $K_{VT}$ | $T_{\min}$ | $K_{VT}$ |
|---|---|---|---|
| 10 | 1.10 | 30 | 0.84 |
| 15 | 1.00 | 45 | 0.76 |
| 20 | 0.93 | 60 | 0.71 |

### 6) Work-piece hardness correction factor

- P Type

| $HB$ | $K_{VHB}$ | $HB$ | $K_{VHB}$ |
|---|---|---|---|
| 120 | 1.18 | 220 | 0.90 |
| 140 | 1.12 | 240 | 0.86 |
| 160 | 1.05 | 260 | 0.82 |
| 180 | 1.00 | 280 | 0.80 |
| 200 | 0.95 | 300 | 0.77 |

- M Type

| $HB$ | $K_{VHB}$ | $HB$ | $K_{VHB}$ |
|---|---|---|---|
| <150 | 1.40 | 270-300 | 0.72 |
| 150-180 | 1.18 | 300-330 | 0.68 |
| 180-210 | 1.00 | 330-360 | 0.66 |
| 210-240 | 0.87 | 360-390 | 0.62 |
| 240-270 | 0.79 | | |

- K Type

| $HB$ | $K_{VHB}$ | $HB$ | $K_{VHB}$ |
|---|---|---|---|
| Grey/nodular/malleable cast iron | | Heat resistant/special cast iron | |
| 160-200 | 1.26 | 200-300 | 0.50 |
| 200-240 | 1.00 | 300-360 | 0.40 |
| 240-280 | 0.80 | 360-450 | 0.30 |
| 280-330 | 0.60 | | |

### (7) Optimized cutting speed

$$v_C = v_{C15} \cdot k_{VT} \cdot k_{VHB} \tag{A.1}$$

Where,

| | |
|---|---|
| $v_C$ | Optimized cutting speed |
| $v_{C15}$ | Basic cutting speed |
| $k_{VHB}$ | Hardness correction factor |
| $k_{VT}$ | Durability correction factor |

## A.9 Structure of TurningToolOODB Java class

```
public class TurningToolOODB extends ToolHolderDB {

        /* properties */
        private String tool_ID;
        private int tool_Qty;
        private int setUpID;
        private String tool_material;
        private float tool_Index_Time;
        private float tool_Cost;
        private String tool_Type;
        private String tool_Direction;
        private String face_Direction;
        private String hold_direction_type;
        private String faceSet;
        private float max_production_time;
        private float used_time;
        private int insert_edge_len;
        private float speed;
        private float feedrate;

        /* methods */
        public void setTool_ID(String tID) {...}
        public void setTool_Qty(int n) {...}
        public void setTool_material(String m) {...}
        public void setTool_Index_Time(float t) {...}
        public void setTool_Cost(float c) {...}

                        ⋮

        public String getFace_Direction() {...}
        public String getHold_direction_type() {...}
        public String getFaceSet() {...}
        public float getMax_production_time() {...}
        public float getUsed_time() {...}
}
```

## A.10 Calculation of production time and cost

**1) Production time factor, $T_p$**

$$T_p = T_m + T_i \times N_t + T_{su} \times N_{su} \tag{A.2}$$

Where,

| | |
|---|---|
| $T_m$ | Machining time $\left(L/\left(f_r * N\right)\right)$ |
| $L$ | Feature length |
| $N$ | Cutting speed in rpm |
| $f_r$ | Maximum permissible feed in mm/rev |
| $T_i$ | Time to perform an index |
| $N_t$ | Number of tools to machine a part |
| $T_{su}$ | Time to perform a set-up |
| $N_{su}$ | Number of tools that must be added to the machine's magazine |

**2) Production cost factor, $C_t$**

$$C_t = C_i \times N_t + C_{su} \times N_{su} + \sum_{i=1}^{n} C_t^{(i)} \tag{A.3}$$

Where,

| | |
|---|---|
| $C_i$ | Cost of performing an index for a tool, $(P * T_i)$ |
| $C_{su}$ | Cost of performing a set-up for a tool, $(P * T_{su})$ |
| $N_{su}$ | Number of tools that must be added to the machine's magazine |
| $C_t^{(i)}$ | Cost of a tool $i$ |
| $n$ | Number of tools required to perform operations |
| $P$ | Pay rate |

**3) Rank of tool alternatives, *Score***

$$Score = \alpha \times C_t + \beta \times T_p \tag{A.4}$$

Where,

| Weighting Condition | $\alpha$ | $\beta$ |
|---:|:---:|:---:|
| Normal condition | 0.5 | 0.5 |
| Cost weighted condition | 1.0 | 0.0 |
| Production time weighted condition | 0.0 | 1.0 |

## A.11 Knowledge tables and equation for calculating the maximum power

### 1) Cutting Resistant and Feed Influence exponent

| Material group | $P_{S1}$ [MPa] | | $1-Z$ |
|---|---|---|---|
| $P_I$ | 1760 | | 0.76 |
| $P_{II}$ | 1770 | | 0.75 |
| $M_I$ | 2530 | | 0.75 |
| $M_{II}$ | Ni, Co alloys | 2895 | 0.76 |
| | Ti alloys | 1860 | 0.79 |
| $M_{III}$ | 2060 | | 0.86 |
| $K_I$ | Grey cast iron | 1120 | 0.78 |
| | Malleable cast iron | 1190 | 0.77 |
| | Nodular cast iron | 1430 | 0.76 |
| $K_{II}$ | Cu alloys | 710 | 0.76 |
| | Al alloys | 508 | 0.78 |
| | Mg alloy | 250 | 0.78 |

### 2) Coefficient, $k_{kr}$

| Approach angle | $k_{kr}$ |
|---|---|
| 90 | 1.00 |
| 80 | 1.015 |
| 70 | 1.02 |
| 60 | 1.04 |
| 55 | 1.06 |
| 50 | 1.08 |
| 45 | 1.10 |

### 3) Required motor power, $P$

$$P = \frac{a_p \cdot f_r^{1-Z} \cdot P_{S1} \cdot k_{kr} \cdot v_C}{42000} \left[kW\right] \qquad \text{(A.5)}$$

Where,

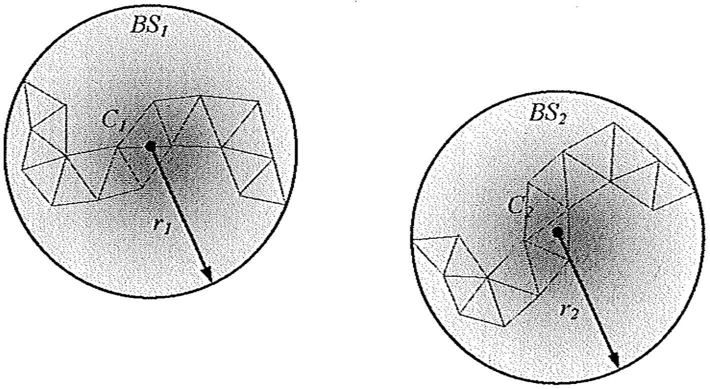
| | |
|---|---|
| $v_C$ | Optimized cutting speed [m/min] (Refer to Equation (A.1)) |
| $f_r$ | Feed rate [mm/rev] |
| $a_p$ | Cutting depth [mm] |
| $1 - Z$ | Feed influence exponent for different materials machined |
| $P_{S1}$ | Specific cutting resistance (cutting force for feed f=1mm/rev at *kr*=90) |
| $k_{kr}$ | Coefficient representing the approach angle *kr* influence |

## A.12 Structure of MachineOODB Java class

```
public class MachineOODB {

        /* properties */
        private int lm_ID;
        private String lm_Model;
        private int lm_swing_over_bed;
        private int lm_machining_length;
        private int lm_min_spindle_speed;
        private int lm_max_spindle_speed;
        private float lm_motor_power;
        private int lm_tail_stock_stroke;
        private int lm_turret_station;
        private float lm_machine_cost;
        private int lm_EIDL;
        private int lm_MTBF;
        private float lm_MTTR;
        private float lm_maint_cost_yr;
        private float lm_pay_rate;

        /* methods */
        public void setLm_swing_over_bed(int d) {…}
        public void setLm_machining_length(int d) {…}
        public void setLm_min_spindle_speed(int s) {…}
        public void setLm_max_spindle_speed(int s) {…}
        public void setLm_motor_power(float p) {…}

                        ⋮

        public float getLm_machine_cost() {…}
        public int getLm_EIDL() {…}
        public int getLm_MTBF() {…}
        public float getLm_MTTR() {…}
        public float getLm_maint_cost_yr() {…}
        public float getLm_pay_rate() {…}
}
```

## A.13 Collision detection methods used in this research

### 1) Bounding sphere (BS)



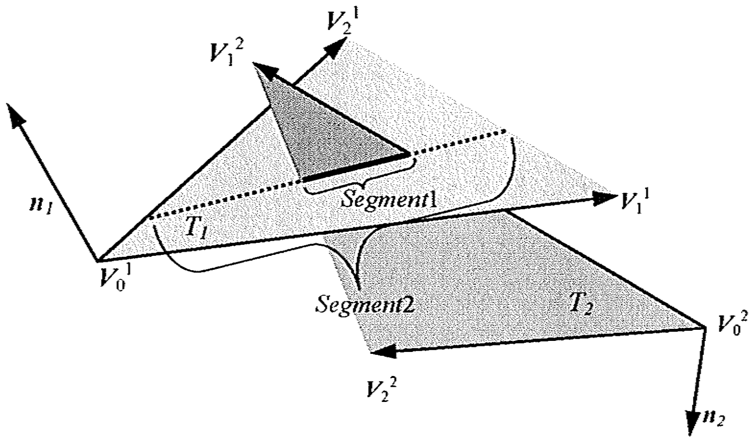
$$(C_1 - C_2)(C_1 - C_2) > (r_1 + r_2)^2 \qquad \text{(A.6)}$$

Where,

$C_1$, $C_2$ Centre point of two bounding spheres $BS_1$ and $BS_2$

$r_1$, $r_2$ Radius of two bounding spheres $BS_1$ and $BS_2$

### 2) Triangle-to-triangle interference check



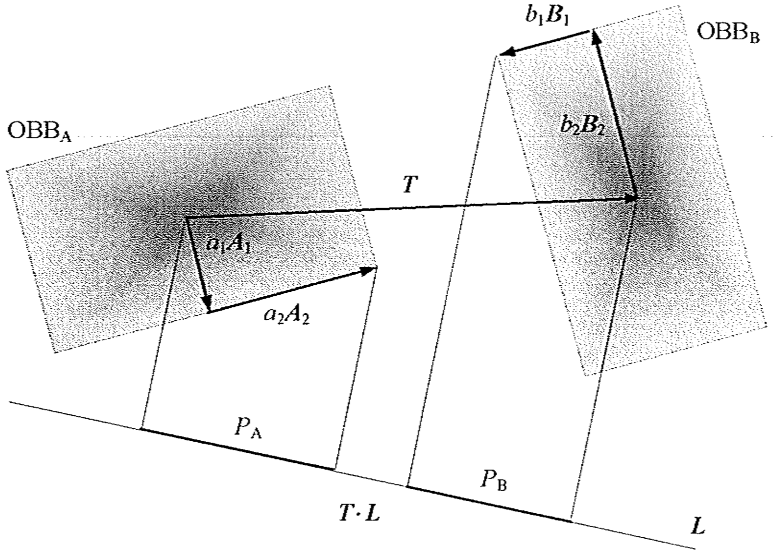
**Algorithm A.12.1**: Triangle-to-triangle interference check

```
Step 1  Compute plan equations $\Rightarrow \pi_i : \boldsymbol{n}_i \cdot X + d_i = 0\ (i = 1 \text{ and } 2)$
        where, $\boldsymbol{n}_i = (V_1{}^i - V_0{}^i) \times (V_2{}^i - V_0{}^i)$ and $d_i = -\boldsymbol{n}_i \cdot V_0{}^i$
Step 2  Compute signed distances of $T_1$ vertices to plane of $T_2 \Rightarrow$
        $d_{V_i{}^1} = \boldsymbol{n}_2 \cdot V_i{}^1 + d_2,\ i = 0, 1, 2$
Step 3  Reject if all $d_{V_i{}^1} > 0$ or $d_{V_i{}^1} < 0,\ i = 0, 1, 2$
Step 4  Repeat for vertices of $T_2$ against plane of $T_1$
Step 5  Find intersection points $\Rightarrow$ i.e., $P_{V_{0\sim1}{}^1} = (d_{V_0{}^1} V_1{}^2 - d_{V_1{}^1} V_0{}^2) / (d_{V_0{}^1} - d_{V_1{}^1})$
Step 6  Do interference check with two segments
    Step 6.1  Let $d_1 = \| P_{V_{0\sim1}{}^1} - P_{V_{0\sim2}{}^1} \|,\ d_2 = \| P_{V_{0\sim1}{}^2} - P_{V_{0\sim2}{}^2} \|$
    Step 6.2  Let $r = \max\{ \| P_{V_{0\sim1}{}^1} - P_{V_{0\sim1}{}^2} \|, \| P_{V_{0\sim1}{}^1} - P_{V_{0\sim2}{}^2} \|, \| P_{V_{0\sim2}{}^1} - P_{V_{0\sim1}{}^2} \|, \| P_{V_{0\sim2}{}^1} - P_{V_{0\sim2}{}^2} \| \}$
    Step 6.3  $r > d_1 + d_2$
```

### 3) Hierarchical Oriented bounding box (OBB)-Tree

**Algorithm A.12.2**: Construction of hierarchical OBB-tree

```
Step 1  Computation of OBB orientation (three-axes)
    Step 1.1  Let the $i^{th}$ triangle of the convex hull have vertices $\boldsymbol{p}^i$, $\boldsymbol{q}^i$ and $\boldsymbol{r}^i$
    Step 1.2  Let the number of triangles in the convex hull be $n$
    Step 1.3  Determine the area of $i^{th}$ triangle $\Rightarrow A^i = |(\boldsymbol{p}^i - \boldsymbol{q}^i) \times (\boldsymbol{p}^i - \boldsymbol{r}^i)| / 2$
    Step 1.4  Determine the surface area of the convex hull $\Rightarrow A_H = \sum_i A^i$
    Step 1.5  Determine the centroid of the $i^{th}$ triangle $\Rightarrow \boldsymbol{m}^i = (\boldsymbol{p}^i + \boldsymbol{q}^i + \boldsymbol{r}^i)/3$
    Step 1.6  Determine the centroid of the entire convex hull $\Rightarrow \boldsymbol{m}_H = \sum_i A^i \boldsymbol{m}^i / A_H$
    Step 1.7  Determine the elements of the covariance matrix $\Rightarrow$
              $C_{n\times n} = \sum_{k=1}^{n} \frac{A^k}{12A_H} [9m_i^k m_j^k + p_i^k p_j^k + q_i^k q_j^k + r_i^k r_j^k] - m_i m_j$
    Step 1.8  Determine the three eigenvectors of $C_{n\times n}$
    Step 1.9  Do normalization
Step 2  Computation of OBB dimension
    Step 2.1  Find the maximum and minimum extents of the original triangle
              set along each axis, and size the OBB
```



$$\exists L : |T \cdot L| > P_A + P_B \quad \text{or} \quad \exists L \in \{A_1, A_2, B_1, B_2\} : |T \cdot L| > P_A + P_B \tag{A.7}$$

Where,

| | |
|---|---|
| $A_1, A_2, B_1, B_2$ | Axes (unit vectors) of OBB$_A$ and OBB$_B$ |
| $a_1, a_2, b_1, b_2$ | Radii of OBB$_A$ and OBB$_B$ |
| $L$ | Unit vector |

$P_A = |a_1 A_1 L| + |a_2 A_2 L|$

$P_B = |b_1 B_1 L| + |b_2 B_2 L|$

## A.14 Structure of AssyToolOODB Java class

```
public class AssyToolOODB implements VectorAnalysis {

        /* properties */
        int toolID;                                 //Tool ID
        String toolName;                            //Tool name
        int toolImageID;                            //Tool image ID
        int toolClass;                              //FAT or TAT
        float[] toolParam=new float[15];            //Tool parameters

        /* Methods */
        public void setNewTool (float[] p) {…}
        public void setToolClass (int c) {…}
        public void setToolImageID (int img) {…}
        public void setToolID (int tID) {…}
        public void setToolName (String tName) {…}
        public void toolVRMLGeometryData (ToolAccessibilityCheck ta, float
        cAlpha, float c_ef, float cU0, float[] fRot, float[] fPos, float color) {…}
        private float[] calculateNewRotationAngleToFastener (float[] rot, float[]
        fRot){ …}
        private float[] calculateNewToolEndPosition (float u, float x, float z)
        { …}
        private float[] calculateNewToolHeadPosition (float u, float x, float z)
        { …}
        private float[] calculateNewToolExtensionPosition (float u, float x, float z)
        { …}
        private float[] calculateNewToolHandlePosition (float alpha, float h, float
        le) {…}

                                    ⋮

        public String getVRMLCodedTool () {…}
        public float[] getTool () {…}
        public int getToolClass () {…}
        public int getToolImageID () {…}
}
```

## A.15 Structure of VRMLGeometryData Java class

```
public class VRMLGeometryData extends ColorData {

        int nTVRML;                          // Total number of parts and fasteners
        String nodeName;                     // VRML node name
        String name;                         // Component name
        int fileType;                        // 0:part and 1:fastener
        int basFastType;                     // Basic fastener type (i.e., bolt, etc.)
        int nCoord;                          // Number of coordinates in this
                                             // VRML model
        int nCoordIndex;                     // Number of coordinate indexes
        float[][] coord;                     // Coordinates used to define this
                                             // VRML model
        int[][] coordIndex;                  // Indexes used to define this VRML
                                             // model
        float fRate, weight, fastRemTime;    // Information including part failure
                                             // rate, weight and fast. removal time
        int nOBB;                            // Number of defined OBBs
        float[] xLim;                    ⎫
        float[] yLim;                    ⎬   // Defined OBBs
        float[] zLim;                    ⎭
        float[] sphereVal=new float[4];      // Defined BS

        VRMLGeometryData(){...}
        VRMLGeometryData(int nt, String arg) {...}
        public float[][] getPatchCoordinate (int pID){...}
        public float[][] getApproxiBox() {...}
        public String getVRMLCode () {...}
        private void setXYZForApproxiBox(float[] x,float[] y, float[] z){...}
        private void setApproxiSphere() {...}

                                ⋮

}
```

# Curriculum Vitae

**Name**: Chulho Chung<br>
**E-mail**: umchung9@cc.umanitoba.ca, ilsjch@gmail.com<br>
**Telephone**: 1-204-474-8572 (O), 1-204-261-7398 (H)<br>
**FAX**: 1-204-275-7507<br>
**Mailing Address**: Department of Mechanical and Manufacturing Engineering,
University of Manitoba, 15 Gillson St. Winnipeg, Manitoba
Canada R3T 5V6

## EDUCATION

**Doctor of Philosophy**, Mechanical and Manufacturing Engineering, The University of Manitoba, Canada, 09/2002-Present.

**Master of Science**, Mechanical Design Engineering, Chungnam National University, South Korea, 08/1996.

**Bachelor of Engineering**, Mechanical and Aeronautical Engineering, Hankuk Aviation University, South Korea, 02/1990.

## PUBLICATIONS

### Refereed Papers in Scientific Journals

**Chung, C.** and Peng, Q. (2006). Tool selection-embedded optimal assembly planning in a dynamic manufacturing environment. Computer-Aided Design (In Review).

Peng, Q. and **Chung, C.** (2006). A visualised CAPP system for agile manufacturing, *Special Issue on Manufacturing under Changing Environment*, International Journal of Manufacturing Technology and Management (In Press).

Peng, Q., **Chung, C.**, Yu, C., and Luan, T. (2006). A networked virtual manufacturing systems for SMEs, International Journal of Computer Integrated Manufacturing (In Press).

**Chung, C.** and Peng, Q. (2006). Evolutionary sequence planning for selective-disassembly in de-manufacturing. International Journal of Computer Integrated Manufacturing, 19 (3): 278-286.

**Chung, C.** and Peng, Q. (2005). A novel approach to geometric feasibility analysis for fast assembly tool reasoning. International Journal of Advanced Manufacturing Technology, DOI: 10.1007/s00170-005-0173-z (Online First).

**Chung, C.** and Peng, Q. (2005). A hybrid approach to selective-disassembly sequence planning for de-manufacturing and its implementation on the Internet. *Special issue on intelligent disassembly in the de-manufacturing process*, International Journal of Advanced Manufacturing Technology, DOI: 10.1007/s00170-005-0038-5 (Online First).

**Chung, C.** and Peng, Q. (2005). An integrated approach to selective-disassembly sequence planning. International Journal of Robotics and Computer Integrated Manufacturing, 21(4-5): 475-485.

**Chung, C.** and Peng, Q. (2005). Selective disassembly sequence planning on the Internet. International Journal of Engineering Simulation, 6(1): 10-16, ISSN 1468-1137.

**Chung, C.** and Peng, Q. (2004). The selection of tools and machines on Web-based manufacturing environments. International Journal of Machine Tools and Manufacture, 44(2-3): 315-324.

**Chung, C.** and Kim, T. (1998). A study on the effective NC machining data generation for aircraft wing via 3-dimensional measured data. Journal of Korean Society for Aeronautical and Space Sciences, 26(7): 60-71.

## Refereed Conference Presentations and Proceedings

**Chung, C.** and Peng, Q. (2006). A dynamic tasks planning system for Web-based collaborative product development. 2006 CIRP International Design Seminar on Design and Innovation for Sustainable Society. Kananaskis, Alberta, Canada, July 16-19.

**Chung, C.** and Peng, Q. (2005). Fast assembly tool reasoning based on geometric accessibility analysis. ASME 2005 Design Engineering Technical Conferences and

Computers and Information in Engineering Conference. Long Beach, California, USA, September 24-28.

**Chung, C.** and Peng, Q. (2005). Tool selection-embedded optimal assembly planning. The 2nd CDEN International Conference on Design Education, Innovation, and Practice. Kananaskis, Alberta, Canada, July 18-20.

**Chung, C.** and Peng, Q. (2004). An integrated approach to selective-disassembly sequence planning. Proceedings of the International Conference on Flexible Automation & Intelligent Manufacturing, Toronto, Canada, FAIM 2004, July 12-14, pp. 262-269.

**Chung, C.** and Peng, Q. (2004). An algorithmic approach to quantitative maintainability analysis in product design. Proceedings of the International Conference on Computers and Industrial Engineering, Cheju, South Korea, C&IE 2004, March 25-27, pp. 1-6.

**Chung, C.** and Peng, Q. (2003). The development of a Web-centred VR-CAPP system. Proceedings of the International Conference on Agile Manufacturing, Beijing, China, pp.369-376, ICAM2003, ISBN 7-111-13395-1, China Machine Press.

**Chung, C.** and Byun, M. (1996). A study on the effective NC machining data generation with arc spline for free form surface. Proceedings of the Annual Symposium of Korean Society for Mechanical Engineers (KSME). pp.1166-1172.

## Research and Technical Reports

Hwang, C., **Chung, C.**, Ahn, S., and Park, H. (2000). A study on the efficient design method for ship-to-ship missile system (SS101, SS102, SS102A), Agency for Defence Development (ADD). MADC-416-000636. pps. 201.

Hwang, C., **Chung, C.**, Ahn, S., and Park, H. (2000). A study on the applications of the digital prototyping technology in developing ship-to-ship missile system. Agency for Defence Development (ADD). MADC-416-000607. pps. 54.

Hwang, C., **Chung, C.**, Ahn, S., and Park, H. (2000). A study on the efficient tolerance analysis for mechanical interfaces of ship-to-ship missile (SS102A). Agency for Defence Development (ADD). MADC-416-000032. pps. 155.

**Chung, C.**, Ahn, S., and Park, H. (1999). A study on the missile outer profile evaluation and reverse engineering algorithm via a 3D laser tracker (LTD-500). Agency for Defence Development (ADD). MSDC-416-990149. pps. 79.

Ha, Y., Lee, K., **Chung, C.**, and Yang, S. (1998). A study on the failure mode effect and criticality analysis (FMECA) for KTX-1. Agency for Defence Development (ADD). ASDC-501-981106. pps. 308.

Lee, K., **Chung, C.**, and Yang, S. (1998). A study on the reliability centred maintenance analysis (RCMA) for KTX-1 structural significant items. Agency for Defence Development (ADD). ASDC-501-981105. pps. 184.

Shin, Y., Ju, H., **Chung, C.**, and Yang, S. (1998). A study on the development of KTX-1 reliability, maintainability, and availability (RAM) prediction program. Agency for Defence Development (ADD). ASDC-501-980299. pps. 100.

**Chung, C.** (1997). An approach to KTX-1 logistics support analysis (LSAP). Agency for Defence Development (ADD). ASDC-415-971287. pps. 403.

**Chung, C.** and Yang, S. (1997). A study on the logistics support analysis for KTX-1 (Part 2). Agency for Defence Development (ADD). ASDC-401-970606. pps. 106.

**Chung, C.** (1997). A study on the logistics support analysis for KTX-1 (Part 1). Agency for Defence Development (ADD). ASDC-401-970280. pps. 101.

**Chung, C.** (1997). A study on the efficient generation and evaluation of aircraft waviness. Agency for Defence Development (ADD). ASDC-401-970261. pps. 55.

**Chung, C.** and Lee, K. (1997). A study on the effective NC data generation for KTX-1 parts with free form surfaces. Agency for Defence Development (ADD). ASDC-401-970023. pps. 236.

**Chung, C.** and Ju, H. (1997). A study on the KTX-1 mission reliability prediction (Part 3). Agency for Defence Development (ADD). ASDC-401-970022. pps. 116.

Kim, J. and **Chung, C.** (1995). A study on the KTX-1 maintainability prediction (Part 1). Agency for Defence Development (ADD). ASDC-501-950656. pps. 329.

Kim, J. and **Chung, C.** (1994). A study on the failure rate analysis for KTX-1 Hydraulic Equipments. Agency for Defence Development (ADD). ASDC-501-940935. pps. 78.

Lee, J., Cho, K., Kim, J., Jeon, J., Ju, H., **Chung, C.**, Yang, S., Ahn, J., and Chang, H. (1994). A study on the KTX-1 mission reliability prediction (Part 1). Agency for Defence Development (ADD). ASDC-401-940400. pps. 73.

## HONOURS AND AWARDS

- University of Manitoba Graduate Fellowships (UMGF), 2004-2006
- TransCanada Pipelines Graduate Fellowship in Engineering, 2005-2006
- Edward R. Toporeck Graduate Fellowship in Engineering, 2005-2006
- Conference Travel Award, University of Manitoba Graduates Studies, 2005-2006
- Edward R. Toporeck Graduate Fellowship in Engineering, 2004-2005
- Conference Travel Award, University of Manitoba Graduates Studies, 2004-2005
- UMGSA Travel Grant, University of Manitoba Graduate Student Association, 2004
- Defence Science and Technology Award, Agency for Defence Development (ADD), South Korea, 2000 (Title: **Chung, C.** *et al.*, A study on the application of the digital prototyping technology in developing ship-to-ship missile systems)
- National Defence-Development Award (KTX-1 program), National Department of Defence (DoD), South Korea, 1999
- Exploit Award (Ship-to-Ship Missile Development Program), Agency for Defence Development (ADD), South Korea, 1999
- Jungsuk Merit-based Scholarships, Hankuk Aviation University, South Korea, 1985-1989.