An information and knowledge framework to support multiple viewpoints in the design for manufacture of injection moulded products

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

Metadata Record: https://dspace.lboro.ac.uk/2134/12921

Publisher: © George Gunendran

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
Please note that fines are charged on ALL overdue items.
AN INFORMATION AND KNOWLEDGE FRAMEWORK TO SUPPORT MULTIPLE VIEWPOINTS IN THE DESIGN FOR MANUFACTURE OF INJECTION MOULDED PRODUCTS

By Anantharajah George Gunendran

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

April 2004

© by George Gunendran (2004)
DEDICATION

To my Parents,

to Marine,

and to Gunalini.
ACKNOWLEDGEMENTS

This work has been carried out at Loughborough University and financially supported by the Wolfson School of Mechanical and Manufacturing Engineering.

Many people have directly or indirectly supported this research and I would like to thank all of them, in particular my friends in rooms TW2.13 and TW 2.14 for sharing their lives and friendship with me during these years of my work.

Naturally, there are also some special thanks to:

Dr. Robert Young for your professional guidance, encouragement, understanding and friendship. I have really enjoyed working with you.

My wife Marine and child Gunalini for your infinite patience, force, optimism, trust and love. Your presence in my life has enabled me to achieve what I have.

My mother, brother, and my wife’s family, thank you for your constant attention, help and contact. You managed to shorten significantly the physical distance between us.

The Sri Lankan community in Loughborough, with whom my family and I have shared so many enjoyable times, for your support, friendship and good food.
Abstract

Integration of product life cycle activities in design and manufacture has been pursued and advanced for over 20 years. A commonly accepted computational approach to integrating product design and manufacture software is to define a neutral representation of product related information for the software applications and capture them in product models. Product models provide a source and repository for all product related information during the product development activities. However, in a team-based environment, the information of product representation needs to be viewed from multiple perspectives. This is because each team member is likely to be interested in different aspects of the information. This leads to the need for multiple viewpoint information representations of the product to be integrated with each other to support product development activities. While much work has been done into the concept of product modelling, there is a need to extend this approach to support information integration between multiple viewpoint product representations by defining techniques to capture the knowledge of relationships of such multiple viewpoint information representations.

The research reported in this thesis identifies a novel method for integrating multiple viewpoint representations of products. An ontology is defined with two separate but related layers to capture multiple viewpoint product representations and the knowledge of relationships between such multiple viewpoints separately. This ontology contains: product model as information layer; knowledge layer to capture integration knowledge; and knowledge links to facilitate the communication of both layers. The work uses injection moulded product design and manufacture as an example to explore and demonstrate the research idea.

An experimental two-layered ontology has been implemented using an object-oriented database and Visual C++ programming language. Experiments have been performed to demonstrate how the two-layered ontology can support the integration of multiple viewpoint information. This research contributes to the understanding of the definition of information and knowledge models and methods of integration to support multiple viewpoint information of products.
### 3. TWO-LAYERED ONTOLOGY FOR FLEXIBLE INFORMATION INTEGRATION BETWEEN MULTIPLE VIEWPOINTS

3.1 Introduction ......................................................... 42

3.2 Background for Information Integration Between Multiple Viewpoints Issues ........... 42

3.3 Flexible Information Integration Between Multiple Feature Sets .......................... 46

3.4 Two-layered Single Ontology for Flexible Information Integration .......................... 51

### 4. PRODUCT MODEL TO SUPPORT MULTI VIEWPOINT INFORMATION INTEGRATION

4.1 Introduction ......................................................... 53

4.2 Product Model Approach ............................................ 53

4.3 Classification of Product Related Information ................................................. 54

4.3.1 Purpose ........................................................... 55

4.3.2 Phase ............................................................. 56

4.3.3 Architecture .................................................... 57

4.3.4 Views ............................................................. 58

4.3.5 Characteristics .................................................. 59

4.4 Methods to Capture Classified Group of Information in Database ......................... 59

4.4.1 Single Instance Approach ........................................ 59

4.4.2 Multiple Instance Approach ..................................... 60

4.4.3 Hybrid Approach ................................................ 61

4.5 Relationships Between the Instances of Product Related Information .................... 61

4.5.1 Relationships Between All Groups of Information ........................................... 62

4.5.2 Relationship Via Common Instance ........................................... 62

4.6 Capturing Classified Groups of Information and Their Relationships .................... 63

4.6.1 Purpose Information and Product Development Activities ............................... 63

4.6.2 Phase Information and Product Development Activities ................................... 65

4.6.3 Architecture Information and Product Development Activities ....................... 66

4.6.4 Characteristics Information and Product Development Activities .................... 67

4.6.5 Views Information and Product Development Activities .................................. 68

4.7 Capturing Relationships Between Multiple Individual Products ......................... 69

4.8 Top-Level Product Model Structure .................................... 70

4.9 Information of Multiple Views and their Integrations ....................................... 70

4.9.1 Relating Multiple Viewpoint Classes Directly .............................................. 72

4.9.2 Relating Multiple Viewpoints Via Common Class ......................................... 72

4.9.3 Functional View as the Core View ................................................. 73
| 7.1 | Introduction | 113 |
| 7.2 | Design and implementation of the experimental system | 113 |
| 7.2.1 | Scope and Experiments | 113 |
| 7.2.2 | The Experimental System Environment and Description | 114 |
| 7.2.3 | Visualisation of the Results/ Contents of Database | 115 |
| 7.3 | Product Model Structure to Capture Product Related Information | 116 |
| 7.3.1 | General Description | 116 |
| 7.3.2 | Populating Product and Its Related Information Instances to the Product Model | 117 |
| 7.3.3 | Visualising the Product and Its Related Information Instances | 119 |
| 7.4 | Experiment to Show that the Product Model Structure is Able to Capture Multiple Viewpoint Information | 121 |
| 7.4.1 | General Description | 121 |
| 7.4.2 | Populating Multiple Views Information of a Product In the Product Model | 121 |
| 7.4.3 | Visualising Multiple Views Information of a Product | 124 |
| 7.5 | Knowledge Layer Structure to Capture the Integration Knowledge | 125 |
| 7.5.1 | General Description | 125 |
| 7.5.2 | Transformation Knowledge and the Structure of the Knowledge Model | 126 |
| 7.5.2.1 | The Population of Transformation Knowledge | 127 |
| 7.5.2.2 | Visualising Populated Transformation Knowledge | 129 |
| 7.5.3 | View Specific Knowledge and the Structure of the Knowledge Model | 131 |
| 7.5.3.1 | Populating View Specific Knowledge | 132 |
| 7.5.3.2 | Visualising View Specific Knowledge | 133 |
| 7.6 | Knowledge Layer to Support Information Integration Between Views | 135 |
| 7.6.1 | General Description | 135 |
| 7.6.2 | Information Transformation Between Views | 136 |
| 7.6.3 | Validation of Transformed Information | 140 |
| 7.6.4 | Information Integration and Knowledge in the Knowledge Layer | 143 |
| 7.7 | Maintaining Consistency in All Views Information With The Support of Information and Knowledge Layers | 147 |
| 7.7.1 | General Description | 147 |
| 7.7.2 | Experiment to Test The Consistency Maintenance in Multiple Viewpoint Information After the Modification in a Populated View | 148 |
| 7.7.3 | Experiment to Test The Consistency Maintenance in Multiple Viewpoint Information After The Modification in Transformed View Information | 152 |
| 7.8 | Integration Between Views of Multiple Products | 154 |
| 7.8.1 | General Description | 154 |
| 7.8.2 | Population of Multiple Related Product Instances and Their Information | 158 |
7.8.2.1 Populating Multiple Product Instances .............................................. 158
7.8.3 Information Integration Between Mouldability View of Plastic Product and
Functional and Machining Views of Mould Products .............................. 162
  7.8.3.1 Information Transformation Between Views of Two Related Products 163
  7.8.3.2 Information Transformation between Functional and Other Viewpoints 163
7.9 Summary .................................................................................................. 165

8. DISCUSSION, CONCLUSIONS AND FUTURE WORK ............................. 166
  8.1 Introduction .............................................................................................. 166
  8.2 Discussion ............................................................................................... 166
  8.3 Conclusions ............................................................................................. 169
  8.4 Recommendation for future work ............................................................ 171

8. REFERENCES .............................................................................................. 172

Appendix A Tools .......................................................................................... 189
  A.1 Introduction ............................................................................................. 189
  A.2 Unified Modelling Language (UML) ...................................................... 189
    A.2.1 Object .................................................................................................. 189
    A.2.2 State, Behaviour, and Identity .......................................................... 190
    A.2.3 Class .................................................................................................... 190
      A.2.3.1 Representing Behaviour and Structure in a Class ....................... 191
      A.2.3.2 Displaying Attributes and Operations of Classes ...................... 191
    A.2.4 Association Classes ........................................................................... 192
    A.2.5 Inheritance Classes ........................................................................... 192
    A.2.6 The Relationships between Classes .................................................. 193
      A.2.6.1 Association relationships ................................................................. 193
      A.2.6.2 Aggregation relationships ................................................................. 194
    A.2.7 Naming the Relationships ................................................................. 194
    A.2.8 Role Names ........................................................................................ 195
    A.2.9 Multiplicity Indicators ....................................................................... 195
    A.2.10 Inheritance Trees .............................................................................. 196
    A.2.11 Single Inheritance Versus Multiple Inheritances ......................... 196
    A.2.12 Inheritance Versus Aggregation Relationships ............................... 196
  A.3 ObjectStore Database ............................................................................. 197
Appendix B  STRUCTURES OF THE INFORMATION AND THE KNOWLEDGE MODELS  198
B.1  Introduction .............................................. 198
B.2  Structure of the Information Model .......................... 198
  B.2.1  Top Level Classes of Product Model .......................... 198
  B.2.2  Characteristics Branch Classes .............................. 199
  B.2.3  Views Branch of Classes .................................. 200
  B.2.4  Class Structure to Capture Feature Representation .......... 200
    B.2.4.1  Class Structure to Capture Functional Features ........... 201
    B.2.4.2  Class Structure to Capture Manufacturing Features ......... 202
B.3  Structure of the Knowledge Model ............................... 205
  B.3.1  Top Level Classes of Integration Knowledge Model ............. 205
  B.3.2  Class Structure to Capture Transformation Knowledge .......... 205
    B.3.2.1  Class Structure to Capture Function to Auxiliary Process .... 206
    B.3.2.2  Class Structure to Capture Function to Main Process ......... 207
  B.3.3  Class Structure to Capture View Specific Knowledge ............ 208
Appendix C  CLASS STRUCTURES FOR FEATURE SETS AND THE EXAMPLES OF POPULATED INTEGRATION KNOWLEDGE
C.1  Introduction .............................................. 209
C.2  Class Structures of Feature Sets ................................ 209
  C.2.1  Class Structure for Mouldability Features ................. 209
  C.2.2  Class Structure for Shaping Functional Features ............ 211
  C.2.3  Class Structure for Machining Features .................... 212
  C.2.4  Class Structure for Assembly Features ....................... 215
C.3  Examples of Populated Integration Knowledge ....................... 216
  C.3.1  Examples of Transformation Knowledge .......................... 216
  C.3.2  Examples of View Specific Knowledge .......................... 220
Appendix D  EXPERIMENTAL SYSTEM INTERFACES AND PROCEDURES OF EXPERIMENTS
D.1  Creating New Product Model .................................. 221
D.2  Populating Product Instances .................................... 221
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.3</td>
<td>Populating Product Related Information Instances</td>
<td>222</td>
</tr>
<tr>
<td>D.4</td>
<td>Visualising Populated Product Related Information</td>
<td>223</td>
</tr>
<tr>
<td>D.5</td>
<td>Population of the Instances of Multiple Views Information Representation</td>
<td>225</td>
</tr>
<tr>
<td>D.6</td>
<td>Visualising Multiple Viewpoint Information</td>
<td>227</td>
</tr>
<tr>
<td>D.7</td>
<td>Creating New Integration Knowledge Model</td>
<td>227</td>
</tr>
<tr>
<td>D.8</td>
<td>Populating Integration Knowledge</td>
<td>228</td>
</tr>
<tr>
<td>D.9</td>
<td>Visualisation of the Integration Knowledge</td>
<td>230</td>
</tr>
<tr>
<td>D.10</td>
<td>Knowledge Layer to Support Flexible Information Integration between Views</td>
<td>233</td>
</tr>
<tr>
<td>D.10.1</td>
<td>Transformation of information between views</td>
<td>233</td>
</tr>
<tr>
<td>D.10.2</td>
<td>Validation of transformed information</td>
<td>237</td>
</tr>
</tbody>
</table>
Chapter 1

1. INTRODUCTION

1.1 Context of the Research

In product design and manufacture, there is a continuing demand for improvements in quality, cost effectiveness and lead-time. Concurrent engineering is a widely accepted philosophy to meet these demands. It allows experts, involved in different phases of product design and manufacturing, to work together in teams, sharing information to support the decision-making processes of design. These characteristics of concurrent engineering require the involvement of a huge amount of information and computation. In addition, the support of computers and software systems is necessary to handle large amount of information and computations to gain the full benefit of the concurrent engineering approach (Vergeest and Horvath, 99; De Martino et al., 98).

The advent of high computational power computers has contributed to the increased use of software applications such as CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), and analysis and simulation packages. However, to work in a concurrent fashion these software tools should be able to cooperate with each other and share information in order to provide more efficient and less error prone product development and manufacture activities.

At the present stage, the level of information sharing between different software systems is very limited and requires human intervention. Information sharing problems are further complicated by inter-operability problems when multiple software system solutions are required, which is typical of the situation in concurrent engineering. The work of the STEP standards community has made progress in information sharing, starting with the product geometry interchange and now providing further standards for product life cycle activities (Teeuw et al., 96; Krause et al., 93). However, the variation and types of information involved in the support of product life cycle activities means that there is a significant amount of research needed. One approach to supporting such information sharing is by defining common information infrastructures, which may be shared by many different software applications throughout the whole product life cycle. This approach can be termed information modelling and be used as a fundamental concept for the research reported in this thesis. Within this concept, there is a need to explore the approaches that can offer flexible integration between product life cycle activities. In the
design stage in particular, this typically involves many members of the design team, and taking different perspectives or views of the problem.

In addition to information sharing, a further point which can bring more confusion in the area of information modelling is the broad range of definitions used for the terms data, information, and knowledge. In this thesis the following definitions from Harding (Harding, 96) are used: Data relates to words or numbers, the meaning of which may vary, depending on context; Information is data which is structured so that it has a particular meaning; Knowledge is information with added detail relating to how it may be used.

The work reported in this thesis contributes to the area of information modelling, and focuses on the methodology for integrating information between multiple viewpoints to support product design and manufacture.

1.2 Research Environment

To address the research issues, the design of injection moulded plastic products and their method of manufacturing has been selected because the plastic product and the mould to produce the plastic product are highly inter-related and hence need to be considered in multiple viewpoints to satisfy the requirements of each viewpoint.

Figure 1.1 Rectangular shaped plastic container

Figure 1.1 shows a simple rectangular shaped plastic container and figure 1.2 illustrates some of the injection mould parts, which are necessary to produce the plastic container by injection moulding. These interrelationships need to be maintained in a multiple viewpoint environment. Within this research, design for function, design for manufacture, and design for assembly views have been chosen to explore the research idea.
The Rational Rose UML (Unified Modelling Language) tool was used to support the exploration of the research idea and the development of information and knowledge structures. In the experimental implementation of this research, the ObjectStore database and Visual C++ programming environment were employed. While the ObjectStore database was used to realise the information and knowledge structures, Visual C++ was selected to realise the functionality of the experimental system that is developed. These were chosen because of their capabilities in integrated working within the object-oriented approach, in addition to being the most accessible/available tools.

Information modelling has significant potential for integrating multiple viewpoint activities and software tools, and a large amount of work has been performed in the areas of modelling product and manufacturing information and knowledge (Young et al., 04). In addition, considerable understanding has been developed to deal with multiple viewpoint considerations in the injection-moulding environment (Canciglieri and Young, 2003). Further value can be added by creating flexible integration among multiple viewpoint information. One way of obtaining this flexible integration is by capturing integration knowledge of relationships between multiple viewpoints. The author's research is based on the hypothesis that "flexible information integration can be supported
across multiple views by capturing the information of multiple views and the knowledge of relationships among such multiple viewpoints in separate but related models. To validate this hypothesis the following aim and objectives have been defined.

1.3 Aim and Objectives

The aim of this research is to contribute to the understanding of the definition of information and knowledge models and the integration techniques to integrate multiple viewpoint information flexibly. In order to reach the above-mentioned aim, the following objectives have been identified.

I. To understand the concepts of information integration in concurrent engineering environment

II. To understand the work that has contributed to the area of information modelling and the integration of multiple viewpoints information.

III. To understand the information representations and requirements of multiple viewpoints.

IV. To identify a flexible environment by which multiple views of information can be maintained.

V. To build an experimental system to test flexible information integration between multiple viewpoints.

VI. To evaluate the results achieved by the experimental system.

1.4 Thesis Structure

This chapter sets the context of the work. The thesis has been organised in eight chapters and the overall structure and contents are presented in figure 1.3. Chapter 2 presents a survey of relevant areas of the work reported in this thesis. It starts with information modelling and moves on to the view specific information representations and methods of information sharing. Finally, it deals with the knowledge involvement of information sharing, ontologies and the frameworks of information and knowledge.

Chapter 3 highlights the contribution of the work in the context of the problem area that the research has addressed. This chapter also defines the contents of chapters 4, 5, and 6,
which provide a description and understanding of double-layered ontology for information integration between multiple views.

Figure 1.3 Structure of the thesis
Chapter 4 provides the structure for the information layer, which is a product model structure to capture all product related information from conceptual to disposal stages of the product's life cycle.

Chapter 5 provides the structure for the knowledge layer, which is defined to capture integration knowledge of multiple viewpoints.

Chapter 6 describes the knowledge links that help to communicate both information and knowledge layers to each other.

Chapter 7 provides the description of the experiments conducted to explore the double layer ontology and to demonstrate the extent to which that can provide support to the information integration between multiple viewpoints.

Finally, chapter 8 presents the conclusions reached in this research, as well as recommendations for further development in the area of information integration between multiple viewpoints.
Chapter 2

2. LITERATURE REVIEW

2.1 Introduction

This chapter presents the results of the literature survey performed by the author about the main theme of this thesis, an information and knowledge framework to support multiple viewpoints in the design for manufacture of injection moulded products. It also provides background information for further discussions in the rest of the chapters in this thesis. Section 2.2 describes an overview of information support systems in a concurrent engineering environment, followed by information sharing in product development activities, described in section 2.3. Section 2.4 reviews issues related to the main aspects of information modelling and management and is followed by section 2.5, which reviews integration methodologies to support multi viewpoint considerations.

The review realised in this chapter form a basis for the critical discussion in chapter 3, which highlights the contribution of the work in the context of the integration of multiple viewpoint information.

2.2 Information Support Systems

Concurrent Engineering (CE) is a manufacturing philosophy that has emerged in the mid eighties in response to growing pressures to reduce cost and lead times while improving the product quality (Hanneghan et al. 00). The essence of concurrent engineering is an integrated and collaborative process, where people in different disciplines cooperate to design products and develop related processes through coordination, communication, and control (Davis and Trapp, 91; Nagy et al., 92; Chen et al., 94). Two approaches are usually addressed for CE implementation: team based, and computational based (Jo et al., 93).

From the computational implementation point of view, research efforts have been made to provide tools to facilitate concurrent process and product development. One of the approaches is the integration of CAD tools to support functionality such as product design (Shen and Barthes, 97; Tomiyama et al., 96), process planning (Anderson and Chang, 90), manufacturability assessment (Chen et al., 95), and cost
estimation (Lee, 92). The second approach focuses on the management and integration of product and process data. It includes developing engineering databases (Wood, 92; Urban et al., 94), developing information models that contain design and manufacturing data (Davis and Trapp, 91; Cлектус, 92; Wong and Sriram, 93; Chen et al., 94) and developing tools for engineering data management (Stover, 93; Miller, 93). The other works are related to the development of tools or functionality that can support the communication, cooperation, and coordination of multifunctional teams (Sriram et al., 91; Cutkosky et al., 93; Saad and Maher, 96; Chen, 99).

The work reported in this thesis follows the second approach, which focuses on the integration of product related information. Specifically, the concept of information modelling has been used as a fundamental concept. In this concept, the information is separated from applications and captured in information models, to offer greater flexibility to integrate the applications, as shown in figure 2.1. In addition, information models are used as a repository and source to share the information related to the life cycle of product and further detailed in section 2.4.

![Figure 2.1 The general information system concept (Young et al., 98)](image)

**2.3 Information Sharing Between Product Development Activities**

Several researchers have identified the necessity of the integration of functional tools in order to get the full benefit of concurrent engineering, (Chen, 99). Concurrent design involves interactions within and between diverse cross-functional teams of individuals who may be scattered over a wide geographic range. Within and between such teams collaboration, co-ordination, and co-decision making are critical to a successful design. The main computational technique for promoting collaboration and
co-decision making between team members is by information sharing (Miao and Haake, 99). However, initially computers were supported very few product design and manufacturing activities. The historical overview of the computer support in the product development activities is given in the following section.

2.3.1 Historical Overview of CAD Modelling Approach

Before 1960, very few computer applications for design existed. Between 1960 and 1970 most computer applications were limited to calculation programs because, there was no interaction between the user and the program (Sutherland, 63).

After 1970, the development of visual display terminals offered the possibility of interaction between the program and the user. However, computer hardware has very expensive and restricted the use of computers in product development activities such as CAD (Salomons, 95).

From the early 1980s rapid developments in the field of information technology in combination with cheaper and more powerful personal computers led to the introduction of CAD on a wider scale (Wagner, 90).

The computer geometrical representation of the design object has also been developed gradually over time. Initially CAD supported drafting activities with 2D geometrical models. These CAD systems provided the user with straight lines, arcs, ellipses etc. The first 3D models were wire frame models (Masuda, 90). However, wire frames could only provide very limited information with no surface or solid data. Surface models have then developed which could be used to model complex shapes. The solid modelling approach has after developed which could provide an unambiguous representation of the geometry (Wilson, 88).

Most of the today's CAD systems use 3D solid modelling approach. These solid models are often derived from 2D sketches by extrusion or sweep operations (Salomons, 95). However, these 3D solid models are able to represent only the geometrical information. The product development activities require not only the
geometrical information but also a broad range of product information. This has lead to the development of product model based approach.

In addition, 3D solid models are unable to represent the semantic information of product. Therefore, feature based product representation approach has been introduced (Bronsvoort and Jansen, 93). However, most of these features are domain specific and unable to understand by different domains. Specially, the designed product information needs to be understood by the manufacturing activities. The following section briefly explains the efforts made to integrate Computer-Aided design and manufacturing activities.

2.3.2 CAD/ CAM Integration

In the beginning of the nineties, researchers had concentrated on specific areas of design and manufacturing such as computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided engineering (CAE), computer-aided process planning (CAPP), material requirement planning (MRP), and design for manufacturing and assembly (DFMA). These tools gave us islands of automation, usually separated by gulf of inefficiencies. To reap the full benefit of the concurrent engineering, several researchers have attempted to integrate these individual functional tools so that they can work concurrently in the overall product life cycle (Li and Chow, 94).

However, CAD/ CAM/ CAE tools are developed for individual development tasks, they are not able to provide information, knowledge, and functions for concurrent product and process development. In order to achieve the goals of concurrent engineering, there is a need to develop a computer-based environment that can support computer aided design and process development in a concurrent basis (Chen 99). Nevertheless, the decisions made in the design stage of the product can have much impact on the complete life cycle of the product. A designer has to take into account requirements of other disciplines involved in the life of the product, for example, manufacturing, assembly, service and disassembly. Ideally, in the concurrent engineering environment, several analyses should be performed during the design stage to check whether such requirements are met. For example, in the context
of manufacturing, it should be analysed whether a product can be manufactured with available resources. This concept is called D(esign) F(or) X, where the X can be any life cycle phase of products (Holland and Bronsvoort, 00; Harding and Popplewell, 96).

The design for X concept promotes the development of CAD tools for multiple functional product phase environments. This leads to the usage of multiple CAD tools for the design phase of a product. Data exchange between these multiple CAD systems is necessary for the ideal concurrent environment and which is performed in the mid nineties via IGES standard and DXF format. These data exchange methodology are technically less advanced and very error intensive (Vergeest and Horvath, 99). However, nowadays STEP community proposes several data exchange and integration methodologies to integrate multiple computer aided systems, and briefly described in section 2.4.5 of this chapter.

Furthermore, the fundamental issue of CAD model sharing is known to be the mismatch between the required semantics of the objects involved and the semantics they actually possess (Vergeest and Horvath, 99). The reason for this mismatch is the lack of description of the semantics of the CAD models. On the other hand, if the semantics is not considered then the problem of CAD communication cannot be solved properly. Another approach followed is to integrate CAD/ CAM activities by features, which have the semantic information, and overcome the limitations of data exchange to a certain extent.

2.3.3 Feature Technology

A feature is an information element representing a region of interest within a product. Features are described the product by an aggregation of properties. The description contains relevant properties including their values and their relations. Furthermore, it is defined in the scope of a specific view of the product description with respect to the classes of properties and to the phases of the product lifecycle (Brunetti and Golob, 00).

In addition, features can be viewed as information sets that refer to the aspects of form or other attributes of a part, such that these sets can be used in reasoning about design,
performance in manufacturing of the part or assemblies. Therefore, the feature technology is expected to be able to provide a better approach to integrate design and manufacturing activities following design, such as engineering analysis, process planning, machining, and fixture (Salomons et al., 93; Chen et al., 96).

Further, concurrent engineering requires the information representation of the product throughout its lifecycle phases. Solid modelling only deals with product geometry but concurrent engineering activities require non-geometric information in addition to the geometric information. Feature modelling deals with the geometric and non-geometric (semantic) information (Bronsvoort and Jansen, 93; Lim et al., 95; Otto and Kimura, 98; Kraker et al., 99). Figure 2.2 illustrates some of the examples of features (Martino and Giannini, 98).

![Figure 2.2 Example of features (Martino and Giannini, 98).](image)

In addition, feature technology can be divided into two major categories; feature recognition, and design by feature (Net et al., 96; Martino and Giannini, 98; Martino et al., 98b). The feature technology approaches and their limitations are presented in the following sub sections.
2.3.3.1 Design by Feature

This is a top-down approach: the object is designed directly using features. The functional knowledge related to a specific context and strictly associated to the features used, can be inserted in the model at design stage. The most commercial CAD systems presently in the market, have adopted the design by feature technology. The major disadvantage of this approach is that the features are context dependent; when considering a different context it is necessary to redesign the object from scratch, using a different and appropriate set of features (Martino and Giannini, 98; Martino et al., 98b).

2.3.3.2 Feature Recognition

This is a bottom-up approach. All the features are extracted from the geometry. It makes possible to consider different applications by changing the recognition rules. The operation rules and sequences can be determined by consulting a pre-defined database, where the operation sequences of each type of feature are described. Serious drawback of this method is the recognition of interacting features. Figure 2.3 represents an example for the architecture of feature recogniser (Martino and Giannini, 98; Martino et al., 98b).

![Feature Recogniser Architecture](image)
2.3.3.3 Limitations of Feature Technology Approach

Most recent CAD systems contain feature based modelling that enriches product data representations with semantic information and supports design with high level modelling entities. However, these systems are still far from satisfying industry’s need to support an automatic transformation of information from design to production for concurrent engineering environment, due to the context dependency of features. Nevertheless, Martino and Giannini (Martino and Giannini, 98) have proposed an approach to overcome these limitations. They combined both design by feature and feature recognition approaches, in order to provide simultaneously multiple context information of products. In addition, several other advanced methodologies have been proposed to overcome the limitations such as freeform features (Ye et al., 01; Berg et al., 02), and cellular model (Bidarra et al., 98).

Further, the features that are offered in present CAD systems are usually predefined within the system, allowing the end-user only to change the parameters of the features (parametric design). In these parametrically modifiable features only the feature geometry can be user defined, and not the topology and other non-geometry related characteristics of the feature. However, for a lot of applications, it is required to be able to define users’ own application dependent features. These features should be application specific both geometrically and topologically, including the non-shape related aspects (Salomons, 95). However, product models required by the actors of the product lifecycle are specific domain feature based. Design by features permits to design the product directly by handling functional features generally pre-defined and parameterised. Those functional features are generally geometrical features like form-features specially built for a particular domain. The main disadvantage comes from the fact that the model cannot support simultaneously all the lifecycle points of view. The product must be re-defined for each domain (Paris and Brissaud, 00).

Therefore, researchers have explored the use of feature technology with only a limited success. Typically, features have been used to represent either one design or manufacturing viewpoint. To be successful, to integrate software systems must link a range of different views of products such as geometry, functionality, and
manufacturing views. This is one of the critical drawbacks with traditional feature based approach (Allanda and Anand, 95; Young and Bell, 93).

Feature technology, whether through feature recognition or design by feature, is concerned with relating two different representations of the same entity. The most frequently addressed issue in this area of research is how to link a geometric view of a product to a manufacturing view of a product, where each of two views has its own underlying data representation.

While feature technology linking two views has had some limited success, with today’s CAE systems often being ‘feature based’, it is very important to note that design and manufacture is a multi viewpoint problem. A product will have a number of design views as well as a number of manufacturing views. Therefore, multiple sets of features are required for the representation of a product to support product development activities. Hence, a CAE system based on multiple sets of features could support the whole product lifecycle activities because each viewpoint requires specific set of features to represent a product.

Each viewpoint needs to be supported and relationships between viewpoints must be captured if appropriate information support is to be made available. Information modelling has been widely recognised and studied by the international research community to support the interaction and sharing of product related information. The following section of this chapter details the information modelling approach. Other information and decision support systems include approaches that use algorithms, agent based systems and knowledge based systems.

2.4 Information Modelling and Management

The information modelling approach has the ability to offer the information for applications in a readily usable manner. The structure and classification of information types is important, if information is to be readily found and used. Classification makes it possible to identify where to look for information (Young et al., 00; Ullman, 02).
Ullman (Ullman, 02) identified and classified ten main groups of information for mechanical engineering design support systems, as shown in figure 2.4.

Figure 2.4 Product related information classified in 10 main groups (Ullman, 02)

This is perhaps the general way of classification without special regard for computational software systems applications. However, Young et al. (Young et al., 00) have classified product related information for computational applications into three main categories: architecture, characteristics, and views. The characteristics group of information contains information about the product geometry, dimensions, tolerances and material description. The architecture group contains the information about assemblies, sub assemblies and components. The views class offers the facility of viewing and using product information from different perspectives which is critical to success, if team based product development is to be successfully supported (Young et al., 00).

This classification is reasonable, however it doesn’t consider the early stages of the product information, such as conceptual stage, and functional information. In the engineering design, the end goal is the creation of an artefact, product, system, or process that performs a function or functions to fulfil customer need(s). Conceptualising, defining, or understanding an artefact, product or system in terms of
function is a fundamental aspect of engineering design (Hirtz et al., 02; Zhang et al., 03).

Hence, the functional information has been identified as the fundamental information for the product development activities (Wolter and Chandrasekaran, 91). This functional information can be decomposed into four kind of information: purpose, function, behaviour, and structure (Rosenman and Gero, 99; Rosenman and Wang, 99).

*Purpose:* describes the needs of clients and intention of designers. It is pursued the aim of the design activity. Generally, it is more conceptual.

*Function:* describes what the product does. Intended functions are those, which are formulated as necessary to fulfil the purpose. Unintended functions may occur, and it is important to note these too.

*Behaviour:* illustrates the working principles of the object and logical actions or influence on other objects. The required behaviours are those, which are necessary to be exhibited in order to produce the required or intended functions.

*Structure:* is what constitutes an artefact or defines its constitution.

The relationships between purpose, function, behaviour and structure can be summarised as: PUROSE enabled by FUNCTION achieved by BEHAVIOUR exhibited by STRUCTURE (Rosenman and Gero, 99; Rosenman and Wang, 99; Tor et al., 02).

The classified product information needs to be captured centrally in a structured manner in order to support interchange, and sharing among product development applications. Conventional CAD systems have file-based storage structures, which undermine the sharing and integration of information (Kim and Han, 03). The information modelling is one of the acceptable methods to capture product related information with their relationships in databases to share between multiple computational applications. Kung et al. (Kung et al., 99) have identified that the usage of database systems to integrate the information in a product development life cycle, can support higher degree of information sharing among the applications.
A model is a representation of the characteristics of a system. The system could be physical, conceptual or analytical. Commonly, the model for representing something physical is a prototype, for something conceptual is a scheme or a method for something analytical is an equation (Prasad, 96).

According to ISO (International Standard Organisation, 97), a model is a representation or description of an entity or a system, describing only the aspects considered relevant in the context of its purpose.

The purpose of modelling is to predict the behaviour of the system. As information is one of the most critical aspects in the product lifecycle, its modelling is essential to allow the designers to work in a concurrent way. Capturing information related to the whole lifecycle of product in an information model is called product modelling. Product models and their kinds are described in the following section. The research reported in this thesis contributes the definition of product model structure to support multiple viewpoint interaction and is detailed in chapter 4.

2.4.2 Product Models

A product model is a representation of a product in computer, and should contain adequate information about the product to satisfy the product information needs of all the applications within the CAE system. The product model is a source and repository of information for many applications, and such model allows information to be shared between many users and the software components of the CAE system (Shaw et al., 89, MaKay 1993, Harding and Popplewell, 96).

In addition, McKay et al. (McKay et al., 96) have described the product data model, which is a structure of product data. The framework acts as the backbone of product model that provides support to the selected engineering applications. It provides a structure, which holds the data, used by engineering applications and it defines the relationships between such data.

The structure of the product model depends on the nature of the product, and on the tools used to model the information. The product model contents are dependent on a particular product. It is impractical attempt to build a generic product model, because the models are bound to contain specific data (Krause et al., 93; McKay et al., 96).
Krause et al. (Krause et al., 93) states that product modelling is a broad topic closely related to many other issues in manufacturing engineering and concerns the complete life cycle, as shown in figure 2.5.

The information processing for product modelling is very complex in the engineering practice. Hence, the term product model can then be interpreted as the logical accumulation of all relevant information concerning for a given product during the life cycle. They store information in the form of digital data (Krause et al., 93).

Product models can be further structured into sub models, so called partial-models, and these partial-models contain information of specific tasks of product development activities. This is illustrated in Figure 2.6 (Krause et al., 93).
Krause et al. (Krause et al., 93) summarised the development of product modelling and the proposed categories of product models, as presented in figure 2.6. There are five types of product models:

1. Structure-oriented product model
2. Geometry-oriented product model
3. Feature-oriented product model
4. Knowledge-based product model
5. Integrated product models

In the structure-oriented product model, specific product data and formats can be stored. This kind of model has many limitations for product representation, such as lack of representation of product shape. However, product shape is important and provides a basis for further enhancement by other modelling techniques (Chin et al., 02).
Geometry-oriented product model is an extension of structure-oriented model with product shape representation. It satisfies the requirements of the computer based representation of the shape of the specific product, but unable to describe non-geometric product information (Chin et al., 02).

Feature-oriented product model has the ability to represent geometry as well as non-geometry product information. A knowledge-based model is an advanced model adopting AI (artificial intelligence) techniques. This model supports the information reasoning, by referring to former designs, human expertise and past experience about a class of products stored in the internal model during the product modelling process (Chin et al., 02).
The integrated product model, or global product model, is a functional combination of all the product models described above, including structure-, geometry-, feature-, and knowledge-based models (Chin et al., 02).

However, the introduction of knowledge in the product-modelling domain as well as integrated multiple kinds of product models are denote great progress, as the degree of automated reasoning is still an important research topic.

2.4.3 Manufacturing and Other Models

A manufacturing model is used to describe available manufacturing processes, resources and strategies. Its purpose is to provide a consistent source of manufacturing information for applications. This model has the potential to contain information, which is valuable to many different parts of the enterprise as a whole as well as to individual project team members. Thus, it may be accessed by many different types of applications, purposes ranging from the formulation of improved business strategies to real time production control. (MOSES, 92; Molina et al., 94; Harding and Popplewell, 96; Lee and Young, 98; Giachetti, 99). The use of manufacturing model in the research reported in this thesis is very limited, but the knowledge of capabilities and limitations of manufacturing methods have been extracted from manufacturing model to structure the integration knowledge model, which is detailed in chapter 5.

Other than product and manufacturing models, there are several information models proposed and used in the CAE environment. For example, the product range model, which captures the history of ranges of products. The aim of this model is to provide support to design decision-making. A range of alternative solution can be defined to offer direct support to the functional needs of the design (Costa, 00).

2.4.4 Multiple Models for a CAE Environment

Several information models have been proposed to be used in combination to support a CAE environment. Following are some examples of the use of multiple information models in combination to support a CAE environment. Some advantages of the use of
multiple models approach have been adopted for the research reported in this thesis and detailed in chapter 3.

MOSES (Model Oriented Simultaneous Engineering Systems) architecture has been designed to satisfy design for function and design for manufacture. It is based on the use of two information models, a product model and a manufacturing model, which can be accessed by an open set of application programs via an integration environment. Figure 2.8 illustrates this architecture. Information models are constructed with Object Oriented Databases. The open set of application programs may contain and used by a concurrent engineering team member during the course of product design. Such applications may include CAD and FEA (Finite Element Analysis) packages, mathematical modellers, expert systems and any other application program, any or all of which may integrate, add, modify, or delete values within a product model database. Interactions between the information models and applications are enabled through the integration environment. Any modifications to the product model are monitored by a specialist co-ordinator application called the engineering moderator (Young, 96; Harding and Popplewell, 96; Canciglieri and Young, 97).

![Figure 2.8 MOSES research concept (Harding and Popplewell, 96)](image)
Main elements of MOSES CAE architecture are;
(1) A product model
(2) A manufacturing model
(3) Strategist applications
(4) Integration environment
(5) Engineering Moderator

Engineering moderator has been included specifically to promote communication and negotiation by the active exchange of information and knowledge between team members with different areas of expertise (MOSES, 94; Harding and Popplewell, 96)

In line with the MOSES project approach, Tonshoff and Zwick (Tonshoff and Zwick, 98) have proposed the integrated usage of product and process model for CAE environment. The integrated product and process model offers the possibility to map product specifications to existing production systems. In reversal, product developers may be enabling to recognise production system limitations in the early stage of product development. Figure 2.9 shows the integrated product and process models approach.

Figure 2.9 Integrated product and process model (IPPM) (Tonshoff and Zwick, 98)
Marpoulos (Marpoulos, 95) proposes an aggregate process planning architecture that uses a product database, aggregate process models and a factory model. The aggregate process model includes information about the layout, machines, quality and suppliers.

The main difference between the approaches proposed by the research group in the Loughborough University (MOSES and MIM projects) and the rest is that the manufacturing model comprises the resources, processes and strategies, while other approaches propose to have separate models for the resources, and process information.

However, most conventional CAE systems do not support structured databases. In many cases, CAD data are stored in file-based storage structures and organised in a proprietary format. This lack of well-defined approach to structuring and accessing data makes communications between CAE tools especially difficult (Urban et al., 95). In order to overcome such difficulties, engineering data standards have been defined.

2.4.5 Standards for Product Data Interchange

In the area of Product Data Interchange (PDI), a number of standards have been proposed such as IGES, SET, DXF, VDAFS, ISIF, and EDIF. For over ten years, IGES (Initial Graphics Exchange Specification) has been used to exchange topologic and geometric data between CAD/CAM systems. In the Europe two alternatives for IGES have been developed. The German motor industry developed VDAFS (Verband der Automobil-ndustrie – Flachenschmittaltelle), since IGES could not satisfy their need. In France Aerospatiale developed SET (Standard d’Exchange et de Transfer), which is frequently used by the French industry. SET is more compact than IGES and can be used to archive data and to integrate applications. Standards such as IGES, VDA-FS, & SET, however, present some disadvantages, which have led to the development of STEP (Standard for Exchange of Product model data). For example, IGES the following problems are mentioned (Bloor and Owen, 91; Teeuw et al., 96)

1. Many applications merely implement part of the IGES standard. As a result, problems arise because it sometimes is not clear; what is implemented, and what is not.
(2) IGES only supports the exchange of graphical data (drawings), rather than several different representations of a product.

(3) The specification of IGES is in some points ambiguous, and leading to individual interpretations in different implementations.

These problems are solved by the STEP. The first problem is solved by the concept of Application Protocols, which is only implement a part of the STEP. The second problem is solved by definition, since the STEP is based on product models. The third problem is solved as STEP standardises both at the physical level (the STEP physical file format) and at a logical level (the language EXPRESS) (Teeuw et al., 96).

The STEP, ISO standard 10303, defines a neutral data format for the representation and exchange of product data. The objective of this standard is to complete a system independent representation of all product-related data during the product life cycle (Krause et al., 93; Ashworth et al., 96; Luo, 00; Oh et al., 01). The technical committee ISO TC184/SC4 is working on the development of a standard structure for a product model (NIST, 97). The methods that are commonly used in the generation of the STEP information models are IDEF0 and the EXPRESS language.

The requirements of the STEP are (Vernadat, 94):

- Support a neutral definition of product and process information.
- Support exchange of product information with the minimum of human interpretation.
- Interrelate a broad range of product information to support applications found throughout the product lifecycle.

STEP is organised into a number of separate standards called Parts (McMahon and Browne, 93; ISO 10303-1, 94). The parts are organised into seven groups as follows (Peng and Trappey, 98; Zhang et al., 98):

1. Description methods
2. Implementation methods
3. Conformance testing methodologies and framework
STEP product modelling is based on integrated resources. Integrated resources, which consist of generic resources and application resources, define a generic information model for product data. As an element of STEP, an application protocol includes the definitions of scope, context, and information requirements of an application. It provides the capability of interpreting the integrated resources to meet the product information requirements of specific applications. The application activity model (AAM), application reference model (ARM), and application-interpreted model (AIM) are the three important resulting models documented in an informative annex to the application protocol. A mapping from the information requirements to the AIM is also provided within an application protocol. An example of an application protocol for the automotive mechanical design process is AP214 (ISO 10303-214, 97); another for configuration control is AP203 (ISO 10303-203, 95). In order to support the development of integrated product models, STEP put forward a formal information modelling language named EXPRESS, which itself is a part of STEP standard (ISO 10303-11, 94). The product data in integrated resources and application protocols are described by EXPRESS to ensure consistency and avoid ambiguity. The graphical representation of EXPRESS is EXPRESS-G, which is provided to aid in understanding the definitions (Chin et al., 02). However, EXPRESS language lacks the I/O capability for computational programming (Yeh and You, 99).

Fowler (Fowler, 95) identified that the need for more broad range of data frameworks and information sharing, but at the present stage STEP community work has focused mainly on data exchange between existing CAE systems.

The ISO standard STEP provides a unique, comprehensive technology to solve most of the industrial problems in data exchange (Jandeleit and Strohmeier, 00). However, it still does not provide all the resources needed to achieve a high-level, semantic data communication between engineering processes.
2.4.6 Product Data Management

Product Data Management (PDM) concerns the management of all data related to the product development environment, for example, a Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), or Computer Integrated Manufacturing (CIM) environment. These environments are characterised by huge volumes of data, complex data structures, concurrent use of data etc. All these trends emphasise the importance of electronic data documentation, archiving, and access, which are the subjects of PDM (Teeuw et al., 96).

Obviously, PDM is closely related to PDI (Product Data Interchange). The overlap between these two research areas is rather large. PDI standards or techniques can be used as a means to meet the general objectives of PDM. For example, the exchange of product data between design environments and logistic (product planning) environment is a problem. Emerging PDI standards and techniques may provide the solution for these problems (Teeuw et al., 96).

In addition, PDM can be seen as an integrating tool concerning many different areas, which ensures that the right information is available to the right person at the right time and in the right form throughout the enterprise (Liu and Xu, 01).

Most of the commercially available PDM systems have the capabilities to offer the following five basic user functions (Chen and Tsao, 98; Chen and Jan, 00; Liu and Xu, 01):

1. Data vault and document management, which provides for storage and retrieval of product information.

2. Workflow and process management, which controls procedures for handling product data and provides a mechanism to drive a business with information.

3. Product structure management, which handles bills of material, product configurations and associated versions and design varieties.

4. Parts management, which provides information on standards components and facilitates re-use of designs.
5. Program management, which provides work breakdown structures and allows coordination between processes, resource scheduling, and project tracking.

In general, PDM offers following advantages to the organisations (Liu and Xu, 01):

1. Interdisciplinary collaboration
2. Reduced product development cycle time
3. Reduced complexity of accessing the information of a company
4. Improved project management
5. Improved lifecycle design
6. Supply chain collaboration

However, present stage PDM having several limitations such as inability to support early stage of the product development and the formal product representation (Szykman et al., 01; Sharma and Gao, 02; Gao et al., 03). In addition to these, present PDM systems cannot respond sensitively to the frequent changes of the design parameters (Chung and Lee, 02).

Current PDM systems get information from CAD/CAM/CAE systems, which generally have file-based storage systems, via exchange mechanisms. IGES (Initial Graphics Exchange), STEP and direct translation between systems have been used as the exchange mechanism. However, information of a product could be distributed in several files, and it is difficult to assess and restore for changes. Kim and Han (Kim and Han, 03) have suggested that the use of neutral database for capturing information can be a solution for this problem.

Historically, software tool vendors have considered proprietary data representations- a significant source for inter operability problems- as part of their competitive advantage. Working to eliminate the barriers of interoperability is often viewed by a software vendor as something that will make it easier for customers to purchase and use a competitor’s product rather than those sold by the company. Nevertheless, some of the PDM systems utilise product database structure as a means of organising key product documents and CAD part files. However, the problems of providing a comprehensive, single source of information, which can support the multiple needs of
a range of design team members and their various software tools have yet to be resolved.

2.5 Integration Methodologies to Support Multi Viewpoint Approach

Currently available tools in the CAE environment typically support point solutions and are not geared to team based support (Gunendran and Young, 02) for example, design for assembly (Boothroyd and Dewhurst, 90), design for machining (Dong and Vijayan, 97; Lu and Moldi, 97), and design for mouldability (Cutkosky et al., 89). Future CAE systems must provide support multiple perspectives of the product, so it is increasingly necessary to balance and integrate multiple views (or perspectives) of the product (Canciglieri and Young, 03). This view is supported by the work of Rosenman and Gero (Rosenman and Gero, 96), they have explored a multiple view approach in architectural design.

2.5.1 Necessity and Requirements of Multi Viewpoint Consideration

Young (Young, 99) has identified that a product will have a number of functional viewpoints and each viewpoint could therefore be said to have its own inputs and outputs. Even a single product can be consider in several viewpoints, for example, figure 2.10 highlights some of the viewpoints, which affect the design and
manufacture of an ice cream cup. If information systems are to be effective, they must be able to support a substantial range of viewpoints and the interaction between them (Young et al., 98; Young, 99; Young et al., 00).

Tang and Frazer (Tang and Frazer, 01) have identified the problem of sharing design information, exploring design solutions from different perspectives and coordinating the decisions made by different members of a design team. The main reason is that the interaction of a team of designers has not been explicitly modelled in the computational terms so far.

Lim et al. (Lim et al., 95) have identified the potential of viewing the same artefact from different views and propose a neutral and domain specific models with mapping capabilities. Inline with this research, several other researchers contribute to the understanding of multiple viewpoints and integration by several methodologies, such as multi-view product model (Tichkiewitch, 96; Tichkiewitch, 97), and cellular model for feature conversion (Bidarra et al., 97). Kugathasan and McMahon (Kugathasan and McMahon, 01) have proposed single central model and multiple view specific models with mapping mechanisms for the integration of multiple viewpoints in the production environment of automobile body parts. In addition, Mawussi (Mawussi, 95) has proposed a model integrating the representation: of the parts, of the raw parts, and forming tools. Harani (Harani, 97) has also based her experiments on the multi-view and the multi-technological case study: an electric motor.

The above-mentioned examples of the researches represent a very limited contribution compared to what is needed for the exploitation and implementation of multi viewpoint CAE systems for the better support of concurrent engineering philosophy by flexible integration of multiple viewpoint information (Bernard and Perry, 03). The following subsections detail the requirements of a multiple viewpoint CAE system.

2.5.1.1 General Requirements

Multiple viewpoint CAE systems must be able to be used at the same time by all the contributors, who have to intervene at any time during the product life cycle, with their own language, based on their viewpoints (Canciglieri and Young, 03). These viewpoint languages could be features; relevant to the specific viewpoint (Tichkiewitch, 96; Tichkiewitch, 97; Bidarra et al., 97). This leads to the argument
that a product needs to be modelled with multiple viewpoint features to be represented in a multiple viewpoint CAE system.

2.5.1.2 Consistency Maintenance in Multi View Information

A product model can represent the product in multiple viewpoints, if it is modelled with multiple viewpoint features. In this modelling system, if a user makes a change in one viewpoint representation, other views need to adjust according to that change, and vice versa. Maintaining consistency in multiple viewpoint representations is a fundamental challenge, and not easily met at this time (Kraker et al., 99; Hofmann and Arinyo, 00; Noort et al., 02). In addition, if a change in some viewpoint representation results in any unsatisfied requirements in other viewpoints, the system needs to handle the contradictions and satisfy those requirements again (Noort et al., 02).

One-way of maintaining consistency of product information is by the capture of all information in a neutral model (Net et al., 96; Hoffmann and Arinyo, 00). This approach is called a 'central model approach'. This requires each software application to derive its view of the information from the neutral format (Spoon and Hardwick, 97).

However, in an integrated product development framework, all the actors of the product lifecycle must cooperate throughout the various product development activities. Each actor brings their competence and skills of their own domain and uses data, knowledge and tools in their own environment. For the efficient work of each actor there is a need for the structure of a product model to be split into different views (Tichkiewitch, 96; Tichkiewitch, 97; Paris and Brissaud, 00). Further, Eastman and Jeng (Eastman and Jeng, 99) have stated that there is a need for both a central and multiple view specific models as depicted in figure 2.11, to effectively supports product lifecycle activities. This approach is called a 'distributed model approach'.
Chapter 2

Figure 2.11 Distributed models approach (Eastman and Jeng, 99)

The distributed model approach is more appropriate than the central model approach for multiple viewpoint product development activities, because multiple formats of product information need to be captured in a product model. The multiple formats of information have complex relationships between them. These complex relationships can be captured in the distributed model approach by defining relationships between the distributed models. If the central model approach is followed, the complex relationship knowledge needs to be captured in the central model in addition to the neutral format of product information. Therefore, the central model approach becomes more complex than distributed model approach in order to support multiple viewpoint product development activities.

2.5.1.3 Knowledge for the Integration of Multiple Viewpoints

Knowledge about the relationships of viewpoints is necessary to transform information between viewpoints (Canciglieri and Young, 03). In addition, Gunendran and Young (Gunendran and Young, 02) have identified that the integration knowledge, which support the information integration between multiple viewpoints, is necessary for the information transformation between viewpoints as well as the validation of transformed information with viewpoint considerations.
Stuurstraat and Tolman (Stuurstraat and Tolman, 99) have identified the requirements of knowledge for an optimum total solution to cope with conflicting requirements of multiple viewpoints. This is inline with the argument of Borg (Borg, 96), who has identified the influence of a single viewpoint decision as a cumulative consequence on the rest of the viewpoints' propagation effect. In addition, Borg (Borg, 96) has proposed a framework for the systemisation of viewpoint knowledge to support product life cycle activities. Zhang and Xue (Zhang and Xue, 02) have highlighted the necessity of knowledge in product lifecycle considerations and propose a distributed data and knowledge base.

These arguments are supported the approach of the research reported in this thesis. The literature review is further expanded to the areas of Knowledge based CAD systems, and knowledge based product data modelling to be aware of existing approaches.

Three approaches have been employed in the integration of a CAD system with knowledge-based application programs (Chen YM et al., 96):

(1) The application programs are completely uncoupled from the design environment and an external file keeps the part model in the knowledge-based environment.

(2) Here also the same approach as (1) except that instead of keeping an external file, a high level model or a graph is maintained by building auxiliary data structure with pointers back to the geometric details of the design environment.

(3) High-level part information, and geometric details are incorporated in a database that can directly support high-level reasoning.

Hashemian and Gu (Hashemian and Gu, 95) have argued that a constraint-based system can be used as mechanism to model the knowledge of product life in the form of constraints. Constraints may vary in type and nature; therefore, any constraint-based approach to design should provide for a variety of constraint types from different aspects of the life cycle of a product (Hashemian and Gu, 95). Constraints may be in the form of rules, or procedural functions and impose certain relationships between connected variables (Lovett et al., 99).
Chapter 2

Choi et al. (Choi et al., 00) have identified that an object-oriented database can be used as a rule base, because it can store and manipulate rule objects efficiently. Further, the rule contains parts as event, condition, and action. The parts of a rule are also treated as objects in the rule base (Choi et al., 00).

Therefore, adding knowledge, irrespective of the method followed, to the conventional CAE systems is a powerful way of providing them with more capabilities (Lovett et al., 99; Chapman and Pinfold, 99).

2.5.2 Multiple Viewpoints Integration Methodologies

The necessity and requirements for the integration of multiple viewpoints have been identified and highlighted in previous section. This section describes the methodologies, which can be followed to construct multiple viewpoint CAE systems.

2.5.2.1 Mapping, Translation and Transformation Mechanisms

Mapping

The generally accepted approach is to capture the information of multiple viewpoints in a CAE system with viewpoint languages. An example for viewpoint language is features, which can represent information of different views by different viewpoint features, such as machining feature, and mouldable feature. Hence, the integration between multiple viewpoints can be obtained by integrate the viewpoint languages. One way of creating integration between the multiple viewpoint representations is by mapping. The Oxford dictionary defines the word mapping as a correspondence by which each element of a given set has associated with it one or more elements of a second set. Methods in feature mapping, which mainly deals with multiple domains, are: information is either added, abstracted, refined or combined in different ways to achieve relevant views in particular domains (Shah, 98; Subrahmanyam, 02).

Mapping can be performed with the help of either a mapping schema language (Spooner and Hardwick, 97; Chin et al., 02), or mapping methodologies (Lim et al., 95; Eastman and Jeng, 99; Kugathasan and McMahon, 01). To date, several researchers have been contributed to the understanding of mapping between multiple
viewpoint representations, but very limited success have been achieved in the understanding of the knowledge between view specific representations. However, most of the mapping methodologies facilitate the mapping of geometrical information between pair of domains. Therefore, flexible information integration between multiple viewpoints by mapping has been reached very limited success.

Translation and Transformation Mechanisms

According to the Oxford dictionary, translation is an expression or rendering of something in another medium, form, or mode of expression. Considerable contributions have been made by the research community for the understanding of the potential of translation mechanisms, or translators in the integration of multiple viewpoints. Urban et al. (Urban et al., 99) have stated that it is difficult to move information/data from one view to another without performing some form of translation. This is inline with the argument of Lee and Young (Lee and Young, 98) and Canciglieri and Young (Canciglieri and Young, 03) have proposed translation mechanisms in order to provide links between multiple viewpoints for an injection moulding environment. These translation mechanisms hold knowledge about the relationship between pairs of views, and therefore can act as a means to translating the information from one viewpoint to the appropriate form needed by another viewpoint (Canciglieri, 99).

However, the translation mechanisms proposed by Canciglieri (Canciglieri, 99) has limitations such as, applicable between pairs of viewpoints, and only in single direction. Further, for N number of different viewpoints, this approach has the potential for creating N(N-1) translation mechanisms as depicted in figure 2.12a. Nevertheless, Teeuw et al. (Teeuw et al., 96) and Urban (Urban et al., 96) have argues that the usage of neutral or shared viewpoint can reduce the number of translation mechanisms as depicted in figure 2.12b (Teeuw et al., 96; Urban et al., 99). A central viewpoint approach has been adopted in the research reported in this thesis.
Most of the literature reviewed in this section terms information navigation as translation. However, from the author’s point of view, a multiple view CAE system require not only the information translation between multiple views, but also the addition of the view specific semantics. This process may require a complete change in the format of the information to navigate from one viewpoint to another. This complete format changing process is termed a transformation of information. The Oxford dictionary defines transformation as a complete change in form, shape, or appearance.

2.5.2.2 Ontologies

Another approach to integrate multiple viewpoint information is by using ontologies. Artificial Intelligence laboratories have initially used ontology, but in recent years many disciplines have developed standardised ontologies for their domains. These ontologies are used to share and explain information in their fields. An ontology defines a common vocabulary for users who need to share information in a domain. It includes definitions of basic concepts in the domain and relations among them. The following are some of the reasons to develop the ontology (Musen, 92): -

- To share common understanding of the structure of information
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyse domain knowledge
Sharing a common understanding of the structure of information among people or software applications is one of the more common goals in developing ontologies (Musen, 92).

There are a number of different definitions for the meaning of ontology; Examples of these are:

1) An ontology is an explicit specification of a conceptualisation (Gruber, 93).

2) An ontology is an explicit, formal specification of a shared conceptualisation (Studer et al., 98)

3) Ontology is a formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions)) (Noy and McGuinness, 01). According to Noy and McGuinness, developing an ontology includes:
   • Defining classes in the ontology,
   • Arranging the classes in a taxonomic (subclass-superclass) hierarchy,
   • Defining slots and describing allowed values for these slots,
   • Filling in the values for slots for instances.

4) According to Labrou (Labrou, 02) an ontology contains three parts: (1) conceptualisation – a hierarchy of concepts and their relationships, which accounts for how one breakdowns the given domain; (2) vocabulary – the terms chosen for objects, relationships and properties; and (3) axiomatisation – constraints on the possible values for attributes and conditions that are applicable across attributes.

5) Tamma and Capon (Tamma and Capon, 02) defined an extensive definition for ontology based on Gruber (Gruber, 93). An ontology comprises concepts, relations, functions, instances and axioms. A concept represents a set or a class
of entities or things within a view. A concept is described in terms of attributes (properties). Attributes can be intrinsic or extrinsic. Intrinsic attributes are those that inherently characterise an object and, they do not depend on any other object. Extrinsic attributes are usually assigned by some external object, and thus are not inherent. Relations and functions describe the interaction between concepts or attributes of a concept. Instances are the individuals represented by the concepts, and finally axioms set constraints on concepts or instances (Tamma and Capon, 02).

The work reported in this thesis uses the definition defined by Tamma and Capon (Tamma and Capon, 02) because it suits to represent a multiple viewpoints information integration environment with an ontology. Information representations of multiple viewpoints can be captured with the concepts of an ontology. The relations and functions of an ontology facilitates to describe the interactions of multiple viewpoints. The axioms are help to represent the knowledge of relationships between multiple viewpoint considerations. Therefore, Ontologies are useful for the task of integrating information of multiple viewpoints.

For the integration tasks, ontologies can be used to describe the semantics of the information sources and to make the contents explicit. In general, there are three different approaches followed to employ the ontology in integration, they are: single, multiple, and hybrid approaches. Figure 2.13 gives an overview of these approaches (Wache et al., 01).

![Figure 2.13 Three different approaches for ontology based information integration (Wache et al., 01)](image-url)
In the single ontology approach, all information sources are related to a global ontology, which provides a shared vocabulary for the specification of the semantics (Arens et al., 96). In multiple ontology approach, the semantics of each information source is described by its own ontology. The hybrid approach is a combination of both single, and multiple ontology approaches (Wache et al., 01). However, single ontology is capable to represent information and knowledge of a multiple viewpoint integration environment. Therefore, the integration with single ontology approach has been adopted in the research reported in this thesis.

2.5.2.3 Agents

The agent-based approach also contributes to the understanding of information integration of multiple viewpoints (Wooldridge and Jennings, 95). There are no real agreements in the definition of agent. However, it can be defined as a piece of software that communicates with other agents through an agent communication language (Genesereth and Ketchpel, 94). Although this is an circular definition, it captures the essential element of agency: agents need not know about the internal and the implementation of other agents (they only need to know and understand the domain) and they communicate with other agents in a high-level language that allows them to make statements about their internal state and the domain they interact in (Labrou, 02). However, the communications or translation of information between distinct domains still causes a problem due to the diversity of types of information involved in a multiple viewpoint environment.

In addition to the above-mentioned methodologies, several other approaches such as multiple models (Net et al., 96, Paris and Brissaud, 00), and translation languages (Eastman and Jeng, 99) have been researched to the usage of multiple viewpoints integration. Further, the combinations of above-mentioned methodologies are also being researched by the international research community for the integration of multiple viewpoint information.
2.6 Summary

This chapter has provided a survey into the three main areas: information sharing, information modelling, and multiple viewpoint integration. These three areas are directly involving with the work reported in this thesis.

Information sharing between CAE systems has been identified as a critical issue, and in general, information models are providing support to the product life cycle activities, by being a repository for information. Although, the concept of information modelling conceptually supports the integration of CAE systems, the limitation to support the integration of multiple viewpoints have been identified. The main issues identified for multiple viewpoint integration are as follows:

1. Definition of information structure, which can support multiple viewpoint representation of products.
2. Integration between such multiple viewpoint representations.
3. Identifying and structuring knowledge that can support multiple viewpoints integration.
4. Maintaining consistency between multiple viewpoints representations of a product.

The research reported in this thesis has focused to address the above-mentioned issues. They are critically reviewed in chapter 3 where the contribution of the work of this research is highlighted.
3. TWO-LAYERED ONTOLOGY FOR FLEXIBLE INFORMATION INTEGRATION BETWEEN MULTIPLE VIEWPOINTS

3.1 Introduction

This chapter briefly explains the background of the research and presents the research concept of the work reported in this thesis. It also provides an outline to the subsequent chapters.

3.2 Background for Information Integration Between Multiple Viewpoints

Issues

As highlighted in the previous chapters, the work reported in this thesis has been performed within the concept of information modelling to achieve interoperability between multiple software systems that support product design and manufacturing activities. It follows the view that the product related information/knowledge from conceptual to disposal phases should be captured in common databases. In addition, these databases need to be designed with a common structure, which is understandable and handled by all software vendors, to support a wide range of product design and manufacturing applications, as shown in the Figure 3.1. An application is a program designed to perform a specific function or group of functions directly for the user, or in some cases, for another application (Eills et al., 95; Young et al. 03). Separating information and knowledge from applications and capturing this in databases, makes the system easily adaptable to the user specific demands without the need to program the entire system again (Jandeleit M and Strohmeier, 00).

Figure 3.1 Common information structure (Young et al., 98)
Each application of product design and manufacture considers different views of the product (Yongwu et al. 99; Stuurstraat and Tolman, 99). For example, a design and manufacturing environment for injection moulded plastic products requires multiple views which consider consumer product design, mould design, mould manufacture, plastic product packing, mould maintenance and disposal views. Figure 3.2 illustrates some of these viewpoint considerations. An information model that captures all product-related information, called a product model, should be available to support activities for all these viewpoint applications (Krause et al. 93).

Figure 3.2 Multiple viewpoints in injection moulding (Canciglieri and Young, 03)

Actors, a combination of software and humans, perform product development activities, where each actor in the product development environment must have his/her own view of the product to efficiently participate and co-design (Rosenman and Gero, 99; Serge et al., 96). The efficient work of a viewpoint actor requires a specific format of information, which is understandable according to the viewpoint considerations (McKay et al., 96). Therefore, each viewpoint information representation requires different information elements to represent the information of a product.

Features, normally used to integrate design and manufacture activities (Brunetti and Golob, 00), can be used to represent view-specific information of a product since
features are building blocks of product representations and hold sets of information of a viewpoint of the product (Lim et al. 95; Tichkiewitch, 97). Multiple sets of features have been defined by standard organisations and several researchers to represent products in different viewpoints. Therefore, multi viewpoint representation of a product in a product model requires multiple sets of view specific features (Martino and Giannini, 98; Holland and Bronsvoort, 00; Noot et al. 02). The background of feature technology is reviewed in chapter 2.3.2. In addition, the feature sets mouldability, shaping functional, and machining that were used in the research reported in this thesis are given in appendix C.

Information integration between these multiple view specific feature sets is necessary for an integrated framework. Lee (Lee, 96) identified that the transformation between view specific feature sets is necessary for information integration. A transformation mechanism, which holds knowledge of related views, can be used to transform information between a pair of view specific feature sets (Canciglieri and Young, 03). Figure 3.3 shows an integrated information environment with a multi viewpoint product model that has transformation mechanisms to integrate multiple view specific feature sets.

Figure 3.3 Multi viewpoint product model in an integrated information environment

(Canciglieri and Young, 03)

For example, producing a plastic tumbler by injection moulding can be considered from several integrated viewpoints such as mouldability, assembly, functionality, and
manufacturing views and represented in the product model with their view specific feature sets. Figure 3.4 reveals the diagrammatic illustration of the relationship between the shape information of mouldability view features of the plastic tumbler to the functional view features of the mould core and mould cavity. The design for function view of the mould-core and the mould-cavity require internal and external surfaces of plastic tumbler respectively. However, this relationship is not simply based on geometry information. A range of knowledge such as material, parting line position, and direction of de-moulding are required for final definition of the core and cavity geometry. Hence the transformation mechanisms that support the transformation of information between multiple viewpoints should contain the knowledge of relationships of the views involved.

Figure 3.4 Example of shape relationship between mouldable and functional views

Even though the transformation mechanisms support the transformation of information, the transformed information needs to be validated in relation to the new viewpoint. So another important part of information integration between viewpoints is validation of the transformed information. For example, figure 3.5 shows the information integration between the functional and machining view feature sets of a mould cavity, where the machining view is one of the possible manufacturing views. Cooling channels of functional views are proposed parallel to the cavity profile to
obtain uniform cooling, but these cooling channels of the functional view can be considered as drilled holes in the machining view, and are easy to produce perpendicular to the originating surfaces. According to the requirements of machining view, cooling channels need to be modified. Hence, the information integration not only requires the transformation knowledge but also the view specific knowledge to verify the acceptability.

Figure 3.5 Example of view specific knowledge involvement in information integration

The extension of Canciglieri’s (Canciglieri and Young, 03) approach with validation procedures offers pair wise transformation between views. However the large number of views involved in product design and manufacturing system is complex and likely to lead to ineffective information navigation between views. The following section introduces an approach to obtain flexible information integration.

3.3 Flexible Information Integration Between Multiple Feature Sets

A major aspect of the work reported in this thesis defines a methodology to obtain flexible information integration between multiple view specific feature sets, which is an unresolved problem in the context of information integration between multiple viewpoints of design and manufacture. Flexible information integration requires information navigation between multiple view specific feature sets, bi-directionally and the ability to update all the features of related views, when a view feature undergoes any modifications (Canciglieri and Young, 03).
Figure 3.6 shows an example of the complexity in information integration between features of the mouldable view of a plastic tumbler and the functional, machining and assembly views of an injection mould. To satisfy the view specific considerations of the machining view, modification to the cooling channel of the cavity insert is required. Once the modification has been performed it should be propagated to all related views. In this example functional and assembly views have to be informed about the modification of the cooling channel in the machining view. Therefore, the information integration requires not only information transformation between a pair of views, but also the ability to update and maintain consistency in the information of all related views.

Using Canciglieri’s (Canciglieri and Young, 03) pair wise transformation approach to obtain flexible information integration between multiple views requires transformations between all pair of views of all related products, as shown in figure 3.7 (a). Also complexity begins when a new viewpoint consideration is added that requires several numbers of transformation mechanisms between the new view and the rest of the views to obtain flexible information integration. When the number of views increases, integration between multiple viewpoints becomes more complex and
difficult task. Hence, pair wise transformation between views is inadequate to obtain flexible information integration between multiple view specific feature sets.

The idea pursued in this research is to overcome the inflexibility of the pair wise transformation approach by introducing a core view with relationships to all other views and performing transformation between the core view and the other views. This approach is shown in Figure 3.7 (b) and further explained in the subsequent chapters. Any modification performed to a particular view can be propagated to all related views via the core view. Further, the addition of a new view requires only one transformation knowledge, that is between the core view and the added new view. Hence, this approach is capable of performing flexible transformation of information between views and easily accommodates new view considerations.

![Figure 3.7 Two different approaches for multiple views integration](image)

The combination of transformation knowledge to perform transformation of information between views and view specific knowledge to validate transformed information can be defined as integration knowledge.

Integration knowledge to support the transformation of information between multiple views should be capable of performing the transformation of information for any kind of geometry and the method of manufacturing of the product (Canciglieri and Young, 03). Any kind of geometry of the product means the shape of the product, for example the shape of the plastic container whether it is rectangular or cylindrical. In addition,
the method of manufacturing means the way of producing, for example injection moulding, casting, machining, and welding. Hence, a generic and extensive approach has been taken in this research in order to reach the above-mentioned requirements highlighted by Canciglieri and Young (Canciglieri and Young, 03). Work reported in this thesis has been explored to separate product related information and the integration knowledge in contrast to the approach of Canciglieri and Young (Canciglieri and Young, 03). In their approach, part of the transformation knowledge captured in the structures of the product model that leads to the definition of multiple structures for product model for different views and products. Integration knowledge can be perceived as being captured in a separate but related layer of information. Figure 3.8 shows a framework of separated information and knowledge layers to support flexible information integration between multiple viewpoints.

Figure 3.8 Information & knowledge layers for flexible integration between views

The two layers illustrated in figure 3.8 can be perceived as two separate, but related databases. The relationships between them are: view information related to their corresponding view specific knowledge; and the core view related to the transformation knowledge.

A knowledge layer is defined to capture multiple groups of integration knowledge, which can be applicable for the integration of multiple viewpoints of a range of products in different product development environments. To perform information integration between particular viewpoints a product requires a specific group of
integration knowledge that is applicable for the involving viewpoints of the product. Hence, a specific group of integration knowledge needs to be identified from the knowledge layer for a particular viewpoint integration. To identify the required group of integration knowledge needs the knowledge of relationships between integration activities and their required specific groups of integration knowledge. Knowledge links have been proposed to capture this knowledge of relationships and the repositories of multiple groups of integration knowledge in the knowledge layer. Hence, the knowledge links are offering a bridge between the information and knowledge layers, as shown in figure 3.9.

Figure 3.9 Framework of information & knowledge layers with knowledge links

Integration of information within the framework of information and knowledge layers and knowledge links is in line with the ontologies approach of integrating different formats of information (Stuckenschmidt et al., 00). Ontology comprises concepts, relations, functions, instances and axioms (Gruber, 93). A brief description of the components of an ontology and how it is suitable to represent a multiple viewpoint information integration environment is given in section 2.5.2.2. In general, there are three different approaches followed to employ ontology in integration, they are single, multiple and hybrid approaches (Wache et al., 01). These three kinds of ontology integration are described in chapter 2.5.2.2.
Chapter 3

Integration based on a single ontology is an appropriate approach (Wache et al., 01) for product design and manufacturing activities, since all product-related information and knowledge could be captured in single information and knowledge databases respectively. In addition, depending on the nature of the changes in one information source, it can imply changes in the global ontology and then mappings to the other information sources. Further, mapping between multiple ontologies is a well-known problem in the knowledge engineering context (Preece et al., 99). Hence, the integration based on a single ontology with separate information and knowledge layers as well as knowledge links provides the procedure to resolve integration of multiple views.

3.4 Two-layered Single Ontology for Flexible Information Integration

![Figure 3.10 Elements of ontology for flexible information integration between multiple views](image)

Flexible information integration between multiple viewpoints can be obtained by capturing integration knowledge and view specific feature sets of information in two distinct but related layers of an ontology as shown in the Figure 3.10. Further,
required applications can be coupled to this double-layered ontology to perform product design and manufacture activities.

The information layer contains concepts, relationships, functions and instances,
- to capture all product related information from conceptual to disposal phases with multiple viewpoint feature sets in separate repositories
- to support flexible information integration between multiple viewpoints

A detailed explanation of the information layer is presented in chapter 4.

Axioms, which contain the relationship knowledge between multiple viewpoints and view specific knowledge, are defined and captured in the knowledge layer. A detailed explanation of the knowledge layer with its contents and the utilisation of information and knowledge layers for information integration between viewpoints with examples are detailed in the chapter 5.

Knowledge links aid the identification and retrieval of the required groups of integration knowledge from the knowledge layer to support multiple viewpoint information integration. A detailed explanation of the requirements of knowledge links in the information integration of multiple viewpoints is presented in chapter 6.
4. PRODUCT MODEL TO SUPPORT MULTI VIEWPOINT INFORMATION INTEGRATION

4.1 Introduction

This chapter describes the information layer and its contents. The information layer is a database defined to capture all product related information and its relationships from conceptual to disposal phase, and is called a product model. The initial part of this chapter briefly explains the product model approach and its general structure. The second part of the chapter explains the product related information and its classification. The third part deals with relationships between classified information and the product model structure to capture these relationships. The final part of the chapter describes the multiple viewpoint feature representation of the product and the complex relationships between the feature sets. In addition to this it also introduces the class structure to maintain consistency between multiple viewpoints information.

4.2 Product Model Approach

The product model concept and its background were analysed in chapter 2.4.2 of this thesis, where it was defined as "a computer readable representation of all product related data" (Young and Bell, 95). This product representation is captured in a database to support product design and manufacture activities through the product life cycle by providing access to and being a repository of such information.

![Figure 4.1 General product model structure](image_url)

The general product model structure, which was defined in the MIM project, is shown in figure 4.1. The limitation of this product model structure is that it supports the
information only from a particular viewpoint, but different software programs need different formats of information to perform their activities (Dorador, 01). One aspect of the author’s works reported in this thesis is the definition of a novel product model structure, which provides a basis for multi viewpoint information integration. Product related information is classified in five main groups and captured in this product model structure. The following section of this chapter explains the product related information and its classification.

4.3 Classification of Product Related Information

Product design and manufacture activities use a large amount of inter-related information. A product model is defined to capture this inter-related product information from conceptual to disposal stages. The management of product related information in a product model helps to offer the right information in the right format to the right application to perform product design and manufacturing activities effectively. However, the management of product related information is becoming a difficult task due to multiple information formats and their inter relationships (Chen and Jan, 00).

One approach to effectively capturing and managing complex related information in a database is classifying it according to its usage. Ulman (Ulman, 02) proposes the classification of product related information in ten types. Details of his classifications are explained in section 2.4.1. The product model defined in the MIM project contains only three main groups of product related information as views, characteristics and product architecture as shown in figure 4.1 (Dorador, 01). However, information related to the purposes of the product and phases of the product life cycle are also necessary for product development activities and the integration of multiple viewpoint information. Hence, the product model structure is defined in the research reported in this thesis with five main groups of product related information: purpose, phase, architecture, characteristics and views, in order to support the integration of multiple viewpoint information. A detailed exploration of each of these classified groups of information is given below.
4.3.1 Purpose

In engineering design, the end goal is the creation of a component or product to fulfil its requirements and purposes. Conceptualising, defining or understanding a product or component, in terms of functions, is a fundamental aspect of engineering design (Ulman, 97, Otto and Wood, 01). A detailed explanation of the functional representation of a product is given in section 2.4.1. This type of representation provides an abstraction to conceptualise and to evolve designs and also to apply to many stages of product development activities: product architecture, concept generation, embodiment and detail design stages, are examples (Hirtz et al., 02). Therefore, the purpose and functional information of a product need to be captured in a product model in order to support product design and manufacture activities. In this research, the purpose group of information has been classified as product related information and it includes information about purpose, intended function and unintended function. Hence, the product model structure has been defined with the purpose group of information in order to capture the purpose and functional information.

![Diagram](Image)

Figure 4.2 Example of some initial thoughts in plastic container design

As mentioned in chapter 1, the manufacture of a rectangular shaped plastic container by injection moulding is taken as a sample product to explain the research idea reported in this thesis. The early design stage of the rectangular plastic container begins with the requirements and purposes in the form of thoughts as illustrated in figure 4.2. In product designing activities these product requirements are further developed to the definition of functions and then concepts to accomplish those
functions. Hence the initial stage of design information has influence on the later stages of product development and manufacturing activities even though it lacks of geometric information (Tang et al., 01). Therefore the initial design stage information needs to be captured in the product model to support the later stages of product development activities. However, the initial stage of the product development information is different from the later stage of well-developed geometrical information. Hence the initial stage product related information could be classified as a separate kind of information and called the purpose of the product.

4.3.2 Phase

Product design and manufacture activities start from the product requirements and are further developed to a functional and conceptual level, as mentioned above. Once the concept is finalised it would be further developed into the embodiment stage, where the information of product geometry begins to be conceived. Embodiment design information is further developed in the detail design stage. In detail design several viewpoint considerations refine the product information according to the viewpoint requirements (Pugh, 90; Ulrich and Eppinger, 00). Figure 4.3 shows, as an example, information added and modified in each stages of the plastic container design.

Figure 4.3 Different design phases of the plastic container

The product related information is modified and added little by little in each stage of the product development activities. Therefore, the product related information has to
be classified according to phases and be captured in the database to support better maintenance and access of the information.

4.3.3 Architecture

Developing product architecture is a key role in the product design and development phases. It encompasses from the functional requirements to alternative product layouts. In addition, the information about related components and assembly of a product are necessary for the decision making in manufacturing and assembly related activities (Stone et al., 00). For example, figure 4.4 shows the components of an injection mould to produce plastic products, where the definition of sprue bush dimensions depends on the dimensions of the cavity insert and cavity plate as well as the fit functions between them.

![Diagram of an injection mould architecture](image)

Figure 4.4 Example of the architecture of an injection mould

Thus the information about the product components and their relationships are necessary to make design and manufacturing decision of each component. Information about involving components of a product is a type of product related information and is termed as the architecture of the product. Hence the product architecture information needs to be captured in the product model.
4.3.4 Views

Every product needs to be considered in multiple viewpoints to check the performance and acceptability of the viewpoint considerations. In each viewpoint, the product is represented in different formats according to the viewpoint consideration (Canciglieri, 03). These different formats of product related information need to be captured separately in the product model to support product development activities (Dorador, 01). For example, figure 4.5 shows different viewpoint representations of the plastic container. The functional view considers the functional requirements of the plastic container, such as store things, partially transparent sidewalls, rectangular in shape and approximate size. The same container has to be considered in the packing view, including considerations like stacking containers one on the other. To satisfy the packing consideration, the container geometry needs to be modified with a taper angle as shown in figure 4.5 of the packing view. Also this product has to be considered in mouldability view to satisfy the mouldability considerations, such as curved corners, and draft angles.

![Diagram of plastic container views](image)

Figure 4.5 Example of multiple views of a plastic container

Multiple views of information of a product represented in different formats are captured separately in the product model. Hence the product related information needs to be classified according to the viewpoint consideration and termed as views of the product. View information is an important group of product related information for the research reported in this thesis, because the integration of multiple viewpoint information occurs within this group of information.
4.3.5 Characteristics

Some kinds of product related information have to be considered in all product development activities (Dorador, 01). For example, the material of rectangular container has an influence in the decision making in all the stages of product development activities, such as the method of manufacturing, the design for manufacture, and the design for assembly. In this example material is a property of the product. Hence, product related information that represents the property of the product has to be captured in the product model separately and termed as part of the characteristics of the product.

Therefore, the information related to a product needs to be classified as functional, phase, architecture, view and characteristics groups to capture and be maintained in the database. The following section explains possible methods to capture classified product related information in the product model.

4.4 Methods to Capture Classified Group of Information in Database

A product model is a database defined to capture information of several products. Information of each product is classified in five main groups. The problem is how to capture and maintain multiple groups of information, which are related to different products, without muddled up each other in a single database. Three possible approaches are given below to overcome this problem.

4.4.1 Single Instance Approach

![Figure 4.6 Database structure to represent product information by single instance](image)
One extreme approach is using a single but complex instance to represent a product in the database. In this approach, the single product instance contains all the product related information, such as purpose, phase, architecture, views and characteristics, as shown in figure 4.6. The advantage of this approach is that a single instance contains all the information related to the product and supports all related activities. However, this approach becomes complex when capturing the relationships of classified groups of product related information. In addition to this, different groups of information are represented in different formats, so capturing different formats of information in a single instance is a difficult task.

4.4.2 Multiple Instance Approach

The opposite extreme is a multiple instances approach, where the information is captured in many separate instances for a single product. Figure 4.7 shows the multiple instances approach, where purpose, phase, architecture, characteristics and views groups of information are captured in separate instances. This approach allows different formats of information to be captured in separate instances.

![Database structure to capture product information by multiple instances](image)

Figure 4.7 Database structure to capture product information by multiple instances
Thus the multiple instances approach has been selected to address the issues of information integration of multi viewpoint design and manufacture activities. However, in this approach, complex relationships between different groups of product-related information need to be captured since the product and its related information are captured in separate instances. Methods to capture the relationships between multiple instances of product related information in the product model are explored in the section 4.5 of this chapter.

4.4.3 Hybrid Approach

Another approach to capturing product related information in a database is the combination of both the above-mentioned approaches. This approach has been applied to the product model structure developed in the MIM project. The product and its architecture information are captured with a single instance approach while the characteristics and views information are captured by a multiple instances approach. Capturing product information and its architecture by a single instance approach has limitations in supporting multiple viewpoint design and manufacturing activities, since a viewpoint considers the product as an individual product while other viewpoints may consider the same product as an assembly or a component.

For example, if an injection mould is considered as a product, then the cavity insert is a sub assembly. If the cavity insert is considered as a product, then it is an individual product. On the other hand, when the feeding system is considered as a product, the cavity insert is a component. So the cavity insert needs to be captured in the product model as an individual product, sub assembly and a component. The single instance approach has the limitation of capturing all these considerations together in a database. Hence, the multiple instance approach has been followed in the research reported in this thesis.

4.5 Relationships Between the Instances of Product Related Information

In the multiple instance approach, different groups of product related information are captured in different instances. However, product related information has complex relationships among the groups. These complex relationships have to be captured in
the product model. Two possible methods of relating multiple instances of product and its related instances are explained below.

### 4.5.1 Relationships Between All Groups of Information

One way of obtaining relationships between related groups of instances is by creating relationships between all classified groups of information as shown in figure 4.8.

![Figure 4.8 Classified groups of information as inter-related to each other](image)

This approach leads to a complex database structure since the product related information is highly inter-related between all classified groups of information.

### 4.5.2 Relationship Via Common Instance

![Figure 4.9 Classified groups of information related via common instance](image)
Another way of relating different groups of instances is by defining a common instance and then creating relationships with the common instance and all other necessary groups of instances as shown in figure 4.9. This approach has been followed in the research reported in this thesis to capture product related information in the product model and is appropriate to address information integration of multiple viewpoint issues.

Hence a common instance needs to be defined for each single product to relate other related classified groups together. This common instance should contain the information about the product name, product code, product ID, etc. so as to enable easy access from the database.

The following section explains how the approach of combining multiple instances with a common instance can be used to address the issues of capturing highly interrelated product information in a database to support multiple viewpoint information integration.

4.6 Capturing Classified Groups of Information and Their Relationships

Classified groups of product related information could be captured in a product model with the help of multiple instances. This section explains each of the classified groups of information and their relationships and how they influence the decision making of the product development activities.

4.6.1 Purpose Information and Product Development Activities

The purpose group of information contains information about the requirements and functions of the product. Every product design and manufacture activity should be verified with this group of information to fulfil the product's intended requirements.

For example, one of the requirements of the rectangular plastic container, which is made out of transparent material, is partial transparency through the sidewall. This requirement needs to be considered in all design and manufacture activities of the plastic container and the mould. Especially in the detail design view of the container
and the manufacturing view of the mould, since these are highly influenced by the requirement, as shown in figure 4.10. The requirement can be accomplished by defining the sidewall with a rough surface finish in the detail design stage of the plastic container. In the mould manufacturing stage, the smooth surface finish of the pocket is easy to produce, but it would affect the plastic intended requirement of the container. Hence the pocket needs to be manufactured with a rough surface finish to produce a plastic container with partially transparent sidewalls.

Figure 4.10 Intended product purposes and their influences in product design and manufacture activities

The product model should be able to support the verification of the purpose instances at different stages of the product development activities. To do this verification, instances of purpose need a relationship between the other instances of the product related information.

To obtain this relationship, the purpose instance should have a relationship with the common instance as mentioned in section 4.5.2. This relationship can be obtained by defining an aggregation relationship between the classes of product and the purpose as shown in figure 4.11. Detailed explanation of the different types of relationships between classes in a database structure is given in appendix A.
4.6.2 Phase Information and Product Development Activities

The phase group contains information about different stages of product development activities. To support product development activities, product information need to be captured in a product model throughout conceptual to disposal stages. Different stages of product lifecycle information are similar but slightly varying. For example, detail design stage and finished product information may vary depend on the accuracy of the manufacturing system. Further, the operational stage information of a product may vary from its initial stage information due to wear and tear. The problem is how to capture and maintain similar but slightly varying multiple sets of information in a database. To overcome this limitation, phase instances have been introduced to support the maintenance of the information of products in a database. Therefore, the phase instances should have relationships with all product related information instances. This relationship can be obtained by defining aggregation relationships between phase class and the product class as shown in the figure 4.12.

Figure 4.11 Purpose and product classes are related with aggregation relationship.

Figure 4.12 Phase and product classes are related with aggregation relationship
4.6.3 Architecture Information and Product Development Activities

The architecture group contains information about related components, sub-assembly, and the assembly of the product. This information is necessary for the product development activities such as design for assembly, and design for machining. For example, the architecture of the cavity half of a mould contains information about its components, such as cavity insert, sprue bush and cavity plate. This information is necessary for deciding the assembly and the method of manufacture of these components. Figure 4.13 shows the interaction between the architecture information and the method of manufacturing the sprue bush.

Hence, the architecture instances should have relationships with all other product-related information instances. This relationship can be obtained by defining an aggregation relationship between the architecture class and the product class as shown in the figure 4.14.

Figure 4.13 Relationship between architecture of mould half and the method of manufacturing of the sprue bush
Chapter 4

Figure 4.14 Architecture and product classes are related with aggregation relationship

4.6.4 Characteristics Information and Product Development Activities

The characteristics group contains common shareable product information of the product such as, material, and geometry. All product development activities are influenced by this common shareable information. For example, the properties of the plastic container material influence the size of the gate and the sprue dimensions, as shown in the figure 4.15.

Hence the characteristics instances should have relationships with all other product-related information instances. This relationship can be obtained by defining an aggregation relationship between the characteristics class and the product class, as shown in the figure 4.16.
4.6.5 Views Information and Product Development Activities

The views group contains multiple formats of product representations such as functional, manufacturing, assembly, and disposal views to be understood by different viewpoint considerations. These view instances influence all product related design and manufacturing activities in producing the product which satisfies all viewpoint requirements. Hence, the relationship between views and other product related instances is necessary. This relationship can be obtained by defining an aggregation relationship between views and the product classes, as shown in figure 4.17.

In addition to this relationship, views instances have to be inter-related to each other to capture inter relationships between viewpoint considerations. How the product model supports the integration of viewpoint information is explained in section 4.9 of this chapter.
4.7 Capturing Relationships Between Multiple Individual Products

Section 4.6 addressed the issue of capturing different groups of information of a single product in the product model, but in reality individual products have relationships among themselves that influence other product design and manufacturing activities. For example, a plastic container and the mould cavity insert to produce the plastic container have relationships in order to produce the geometric shape of the plastic product by defining cavity geometry. Similarly the mould cavity and the mould plate have relationships. Examples of these relationships are shown in figure 4.18.

A product model should be able to capture these relationships between individual products. To create this relationship, a product instance needs a relationship with another product instance. This relationship can be captured in the structure of the product model by creating the “product has product” relationship as shown in figure 4.19.

![figure 4.18 Relationships between individual products](image)

![figure 4.19 Product has product relationship](image)
4.8 Top-Level Product Model Structure

Product related information could be classified into five main groups as purpose, phase, architecture, characteristics, and views. These classified groups of information are related via a common instance with aggregation relationships to support multiple viewpoint information integration. In addition to these relationships, individual product instances are also related by "product has product" relationships. The product model structure, shown in figure 4.20, captures all of these relationships. This is the top level of the product model structure. Detail of each of the branches of the product model is presented in appendix B.

![Diagram of product model structure]

Figure 4.20 Top-level classes of product model

4.9 Information of Multiple Views and their Integrations

The multiple viewpoint representation of a product can be captured by multiple instances of the views group information. These viewpoint instances and product instance are related by an aggregation relationship between product and view classes of the product model structure. However, the interaction between multiple viewpoint instances, which is necessary to support multiple viewpoint information integration, is not explicitly captured in the product model structure, as shown in figure 4.20,
although in reality, multiple viewpoint information instances are highly interrelated to each other. For example, the functional view of plastic container contains functional requirements such as geometric shape and partial transparent through sidewall. The mouldability view of this plastic container represents the product in a mouldable manner and also needs to satisfy the geometrical shape requirements of the functional view of the container. The functional view of the cavity insert should capture the shape information, which is captured in the mouldable view, as a functional requirement. The machining view of the cavity insert represents the requirements of the functional view of the cavity insert in a machineable manner. Relationships between these multiple viewpoint representations of the plastic container and the cavity insert are elaborated in figure 4.21. Methodologies to capture the relationships between multiple viewpoint representations are explained in the following section.

In the above-mentioned example, the relationships of multiple viewpoints look like linear relationships between pairs of successive viewpoints, but in reality multiple viewpoints are inter-related in a complex manner. A detailed explanation of the complex relationships between multiple viewpoint representations is presented in section 4.10 of this chapter.

Figure 4.21 Multiple viewpoint representation of plastic product and the mould
4.9.1 Relating Multiple Viewpoint Classes Directly

One way of interacting multiple viewpoint representations of the product is by creating aggregation relationships between different viewpoint classes, as shown in figure 4.22 (Cancigileri, 03). In this approach, each of the viewpoint classes is inherited from the parent view class and is inter related by aggregation relationships. This approach becomes more complex when the number of viewpoint considerations increases.

![Diagram of aggregation relationships between each viewpoint classes](image)

Figure 4.22 Aggregation relationships between each viewpoint classes

4.9.2 Relating Multiple Viewpoints Via Common Class

Another approach has been proposed in the research reported in this thesis to overcome the limitation of the above-mentioned approach. In this approach, product and views classes are related by "has" relationships, as explained in section 4.6.5. Hence, every instance of different viewpoint representations of a single product is related via the product instance, as shown in figure 4.23. Therefore, separate relationships between each of the viewpoint representations are not necessary to access and integrate different viewpoint instances.
4.9.3 Functional View as the Core View

In the above-mentioned approach, instances of multiple viewpoint representations are interrelated to each other via product instance. Even though the viewpoint instances are related to one another, the functional view contains requirements and functions of the product. These functions and requirements of a product need to be verified with all other viewpoint representations, because some of the viewpoint considerations may affect or ignore the product’s intended requirements or functions.

For example, the functional view of the plastic container contains information about the intended requirements of the product, such as geometric shape, and partial transparency through sidewalls. Hence, the geometrical shape information can be transformed and modified according to the mouldability view considerations, while other requirements such as partial transparency of sidewall may be ignored since it is not a mouldability issue. Geometrical shape information can be further navigated to the functional and machining views of the mould cavity insert. If all the functional view instances of the plastic container were not verified with the machining view of the cavity insert, partially transparent through sidewalls, which is one of the intended functions of plastic container, would be ignored. Figure 4.24 shows the cross verification of the functional view with other views. Thus, it can be seen that all the transformed views should be verified with the functional view to avoid misinterpretation or ignoring of the intended requirements. Hence, the functional view should be considered as a core view and has to be verified with all other views.
Even though the functional view has been considered as a core view, it is not necessary to have special relationships between the core view and other views, because multiple views of information are individually captured in separate instances and related via the product instance. However, the transformation knowledge, which supports information integration between multiple viewpoint representations, should be able to access the core view instances and transform to other viewpoint instances as shown in figure 4.25. A detailed explanation of transformation knowledge and the transformation of information between views are explained in chapter 5 of this thesis.
4.10 Viewpoint Information Representation

The earlier sections of this chapter detailed how multiple viewpoint integrations are performed in addition to the classification and capture product related information with their relationships. The following section explains the method of representing a product in different viewpoints. Each viewpoint information representation requires a different format of information that can be understandable by that viewpoint. As mentioned in the previous chapter features can be used to represent the product in different viewpoints. In addition, a single viewpoint information representation needs multiple features.
Figure 4.26 Machining feature representation

For example, the cavity insert can be represented in the machining view by machining features as shown in figure 4.26, where pocket and hole are machining features. This cavity insert machining view instance should contain a pocket and four hole-instances of machining features. So a single view instance has to be related to different feature instances. To capture this relationship in the product model, the view class needs to be related to a feature representation class by an aggregation relationship, as shown in figure 4.27. Different viewpoints and their feature representations can be captured in the product model with the help of children classes of views and feature representations respectively. The following section explains multiple viewpoint feature sets and the database structure to capture feature representations.

![Diagram of Views relationship](image)
4.10.1 Feature Sets and Their Possible Options

Representing a product in multiple viewpoints requires different sets of viewpoint features and each set of viewpoint features should be able to represent all possible options of the viewpoint information. For example, the machining view of a product needs to be represented by possible machineable shapes such as holes, pockets, cylindrical shape, flat surfaces, slots, and chamfers. Figure 4.28 shows some of the machineable shapes. The set of machining features should be able to represent all of these machineable shapes and be able to be captured in the product model.

To capture a set of machining features in a database requires separate classes for each of the machining features, as shown in figure 4.29. Classes of machining features are extracted from STEP defined machining features (ISO 10303-224, 2000). The detailed class structure to capture possible machineable features is given in appendix C. Thus a machining view of a product can be represented by the collection of required machining feature instances and captured in the product model.
Similarly the functional view of the mould can be represented with the help of mould functional features such as shaping, feeding, venting, and ejecting (Ress, 95; Menges et al., 00). Some of the mould functions are shown in figure 4.30.

There are several methods to accomplish each of the functions of the mould. For example, the venting function of the mould can be performed by parting line venting, pin venting, etc. Figure 4.31 shows some of the possible venting methods.
Figure 4.31 Example of venting methods

Figure 4.32 Example of class structure to capture mould functions
Functions of the mould and all possible ways of accomplishing its functions have to be captured in the product model. Figure 4.32 shows an example of the database structure to capture possible ways that accomplish the functions of the mould. Similarly, product views can be captured in the database with view specific feature sets. The detailed explanation of each of the view specific feature sets that are used in the research reported in this thesis can be found in appendix C.

4.10.2 Feature Representations and Their Relative Positions

As explained above, a product view can be captured in a database with the help of view specific feature instances. However, the feature sets themselves are not enough to define a view of the product, it needs information about the relative positions of the feature instances. For example, a cavity insert can be represented in the machining view by machining features, as shown in figure 4.33a, but the machining view activities require more information than machining features such as relative positions of machining features as shown in figure 4.33b.

![Feature representation](image)

**Figure 4.33 Machining view of a cavity insert represented with the help of machining features and their relative positions**

To capture the relative position information of features in the product model requires a separate class called relative position relationship. In addition to this, two different view specific feature instances need to be related via the relative position information instance. In the above-mentioned example, hole 1 and hole 2 instances can be related.
via the relative position instance. For example, this relationship between two holes could be captured in the product model, as holes 1 and 2 are 5cm apart. So the relationships between feature representation and relative position classes require double aggregation relationships, as shown in figure 4.34, to relate two instances of feature representations.

Figure 4.34 Feature representation and relative position classes are related by two has relationships

4.10.3 Complex Relationships of Features Among Multiple Viewpoints

The above section addresses the issue of capturing the viewpoint information in the product model. Multiple viewpoint information is captured with the relationships that have to support the information integration activities. During the information integration activities viewpoint features undergo either modification or transformation. If a feature undergoes modification according to the viewpoint consideration, that should be propagated to all related viewpoints features.

Figure 4.35 Cavity insert and sprue bush assembly
For example, the cavity insert and sprue bush are assembled by push fit as shown in figure 4.35. The tolerances of the sprue bush and the cavity insert hole are decided in the assembly view, which depends on the fit function of the functional view. Also the method of manufacturing of the hole and the bush are decided in the manufacturing view, which depends on the tolerances of the assembly view. If the fit function is modified, that could affect the tolerances of the assembly view and the method of manufacturing of manufacturing view. So the modifications have to be propagated to all related view features. The problem is how to identify the related and dependent features to propagate the modifications. Capturing dependent feature information along with each instance of features is one possible method, which is proposed in the research reported in this thesis, i.e., to track dependent features.

Figure 4.36 Feature representation and depended feature classes are related by has relationship

Figure 4.36 shows a part of the product model structure to capture the information of dependent features along with each feature instance. As shown in figure 4.36, the dependent feature class has two children classes, “Transform to” and “Transform from”, to capture the dependent feature information. As mentioned in the above example, assembly functions are transformed as tolerances to the assembly view. So the dependent relationship between functional view and assembly view features can be captured using the class structure, as shown in figure 4.36. The fit function instance can capture the relationship of the tolerance feature instance by using the “Transform to” instance. In a similar way, the tolerance feature instance can capture the relationship of the fit function instance by using the “Transform from” instance. If information of dependent features is captured along with the instances of each feature, the modification of a feature can be propagated to all its related feature instances.
4.11 Summary

In this chapter, the product model structure, derived in order to support multiple viewpoint information integration, has been described. In addition, the following issues have been addressed:

1) Product related information could be classified in five main groups, as purpose, phase, architecture, characteristics, and views, to support better maintenance of the multiple viewpoints product model.

2) These five main groups of information instances can be related to each other by a common instance.

3) Multiple viewpoints information instances are also interrelated by the same common instance.

4) View specific features can be used to represent the product in multiple viewpoints.

5) The relative position relationship class helps to capture the relationship between view specific features.

6) The dependent feature class helps to capture and track dependent feature information.

The top-level classes of the product model structure, which satisfies the above issues, is shown in figure 4.37. The detailed product model structure that is proposed and used in the research reported in this thesis is given in appendix B.

Figure 4.37 Top-level product model structure
Chapter 5

5. INTEGRATION KNOWLEDGE MODEL TO SUPPORT MULTI VIEWPOINT INFORMATION INTEGRATION

5.1 Introduction

This chapter describes the main novel part of the work reported in this thesis that is the knowledge layer and its contents. The knowledge layer is defined to capture integration knowledge to support information integration between views. The initial part of this chapter explains the requirements of information integration between views. The second part deals with the integration knowledge and the class structure to capture the integration knowledge. The final part explains, with examples, how the knowledge layer supports the information integration between multiple viewpoints.

5.2 Requirements of Knowledge Layer for Information Integration

The necessity of information integration between multiple viewpoints is highlighted in chapters 1 and 2 of this thesis. The information integration between multiple views requires transformation and view specific knowledge, as detailed in chapter 3.

Figure 5.1 Example of multiple viewpoint representations and transformation knowledge of plastic product and injection mould
Figure 5.2 Different data structures for different products and views (Canciglieri OJ, 03)
An example of multiple viewpoint information and the transformation knowledge between views for an injection moulded plastic product and the mould to produce that plastic product is shown in figure 5.1. The information representation of multiple viewpoints can be captured in the information layer, as detailed in chapter 4. The integration knowledge needs to be captured in the ontology to support information integration between multiple viewpoints.

Integration knowledge is separated from information and captured in a separate knowledge base in order to define a single general structure to capture information of all kinds of products. This separation of information and knowledge has been reasoned in chapter 3. Further, this approach overcomes a major limitation of Cancigileri’s (Cancigileri, 99) approach where part of transformation knowledge is tied with the structure of the product model and leads to the definition of multiple product model structures for different products and views. For example, figures 5.2 a and b show structures of mouldability views of prismatic and rotational plastic products. These two structures look similar, but the classes and attributes are different. Also figure 5.2 c shows a prismatic product machining view structure, which is different from the mouldability view structure of the prismatic product as shown in figure 5.2b.

<table>
<thead>
<tr>
<th>No. of PP</th>
<th>Product Type</th>
<th>Mould Type</th>
<th>Wall Type</th>
<th>Position WRT PP</th>
<th>Inclination WRT PP</th>
<th>Required Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Primary</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
</tr>
<tr>
<td>Option 2</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Transition</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
</tr>
<tr>
<td>Option 3</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Transition</td>
<td>Below-Non-Touching</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Option 4</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Primary</td>
<td>Below-Non-Touching</td>
<td>Parallel</td>
</tr>
<tr>
<td>Option 5</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Primary</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
</tr>
<tr>
<td>Option 6</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Transition</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
</tr>
<tr>
<td>Option 7</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Transition</td>
<td>Below-Non-Touching</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Option 8</td>
<td>1</td>
<td>PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Primary</td>
<td>Below-Non-Touching</td>
<td>Parallel</td>
</tr>
</tbody>
</table>

Figure 5.3 Tabulated transformation knowledge between mouldability and shaping views

A possible method for capturing transformation knowledge in a knowledge base is tabulating options of different conditions. For example, transformation knowledge between the mouldable view of the plastic product and the functional view of the mould can be tabulated as shown in figure 5.3. Each row of the table captures a possible occurrence of transformation between views. These rows of options can be
added to the knowledge base by populating conditions. The following sections of this chapter explain the structure of the knowledge base and how integration knowledge can be captured with this structure to support information integration between multiple viewpoints.

5.3 Structure of Knowledge Base to Support Information Integration

Integration knowledge needs to be captured in the knowledge base to support information integration between viewpoints. Integration knowledge contains transformation and view specific knowledge, as described in chapter 3. Transformation knowledge helps the transformation of information between views, while view specific knowledge helps to validate the transformed information with view specific considerations. Integration of information between feeding sub functional, assembly sub functional, assembly and machining views of cavity insert and sprue bush are taken as examples to explain the influence of transformation and view specific knowledge in the information integration between multiple viewpoints.

The information transformation between the sub functional and the assembly views of the cavity insert and sprue bush is performed initially. This information transformation is performed with the help of the hole and shaft assembly function to assembly process transformation knowledge, which are extracted from British Standard Selected ISO Fits- Hole basis (BS 4500, 1969). In this particular example, the assembly functional view proposes push fit and the feeding functional view proposes the sprue bush external diameter as 21mm. These sub functional views information are transformed to the assembly view with the transformation knowledge tabulated in the table shown in figure 5.4, where the tolerance range for push fit of hole and shaft diameter of 18 to 30mm is 0.002 to 0.015. Figure 5.4 diagrammatically illustrates the sub functional to assembly views information transformation of cavity insert sprue bush.
Figure 5.4 Information transformation between function and assembly views of cavity insert and sprue bush

Similarly, the assembly view of the sprue bush can be transformed to the machining view with the help of the geometry to machining transformation knowledge as shown in figure 5.5.
Chapter 5

Figure 5.5 Information transformation between assembly and machining views of sprue bush

In the machining view, the machining process will be decided dependent on the shape and the tolerances of machining features. In the above-mentioned example, it is not possible to produce the outer diameter even by high quality conventional machining processes with the given tolerance range, which is shown in figure 5.6.
How to machine with this tight tolerance?

Figure 5.6 Checking with machining view specific constrains

This limitation of the machining view will be overcome either by informing the functional view and getting machineable tolerances, or by proposing a different method of manufacturing. So the information integration between multiple viewpoints requires transformation knowledge and view specific knowledge. Hence the knowledge base structure to capture integration knowledge should have two main branches transformation, and view specific knowledge, as shown in figure 5.7. These transformation and view specific knowledge are explained in detail in the following sections of this chapter.

Figure 5.7 Two main branches of integration knowledge
5.4 Knowledge Base Structure to Capture Transformation Knowledge

Transformation knowledge helps to transform information from one viewpoint to another in the multiple viewpoints environment. Even though the transformation knowledge supports transformation between multiple viewpoints, the actual transformation is performed between a central view and other views. As explained in chapter 4.9.3, the functional view can be considered as a central view and the information transformation can be performed between the functional view and the other process views.

However, product development activities begin with purposes and requirements of the products. In the next stage of product development, concepts are developed to achieve the requirements and purposes of the products. Concepts are developed in order to accomplish functions of products with certain behaviour (Ulrich and Eppinger, 00). Further, the relationship of functions and behaviour is detailed in chapter 2.4.1. Hence, the information related to a product is undergoes transformation in three main stages as follows: requirements to concepts, concepts to functions and functions to processes. These transformations require different sets of transformation knowledge. Therefore, the transformation knowledge can be classified into three main branches: requirements to concepts, concepts to function and function to process.

The requirement to concept and concept to function transformation knowledge can be derived from the database of requirements, concepts and functions. The function to process transformation knowledge can be derived from the process model, which contains information about all processes (Tonshoff and Zwich, 98). The main branches of transformation knowledge and its related databases are shown in figure 5.8.
5.4.1 Function to Process Transformation Knowledge

Functional information can be transformed to process information with the help of function to process transformation knowledge. So the transformed processes are performed to accomplish functional requirements. The processes can be divided into two main categories: main and auxiliary processes. The differences between these two processes are explained in the next section of this chapter.

5.4.1.1 Main and Auxiliary Processes

The processes that perform the important intended functions are called the main processes. The processes that help to perform the main processes are called the auxiliary processes. The functional requirements can be achieved by performing the main processes, but to perform the main process might need the support of the auxiliary processes. The auxiliary processes also influence the performance of functions, even though they are not directly involved in the functional requirements. The shaping function of the plastic product and the cavity insert are taken as examples.
to explain the difference between the main and auxiliary processes. The shape of the plastic product is generated by injection moulding, by injecting molten plastic into the mould, but other processes such as cooling, venting, and aligning are also influencing the shape of the plastic product, as illustrated in figure 5.9. In this example, these supporting processes could be called auxiliary processes. Similarly the machining process, which is the main process, can offer the shape of the cavity insert. However, the shape of cavity insert is indirectly influenced by machining supporting processes such as feeding, cooling, and aligning, as listed in figure 5.9. These supporting processes could be called auxiliary processes.

Figure 5.9 Main and auxiliary processes of injection moulding and machining

Figure 5.10 Two main branches of function to process transformation knowledge
The function to process transformation knowledge has to consider both the main and auxiliary processes. Hence the structure to capture function to process transformation knowledge should contain main and auxiliary processes as sub classes of function to process transformation knowledge class as shown in figure 5.10. These main and auxiliary processes are further explained in the following sections of this chapter.

5.4.1.2 Function to Main Process Transformation Knowledge

The function to main process transformation knowledge transforms functional requirements to main processes. The shaping and assembly main processes are taken as examples to demonstrate this. Any of the shaping processes can be classified as either a forming, adding or removal processes. Machining and other chip removal processes are some examples of removal process. Welding could be considered as an adding process, and casting, forging and injection moulding can be classified as forming processes. In a similar manner, the assembly process can be classified as an aligning, mating or fixing processes.

To perform these main processes some auxiliary processes are necessary. This relationship can be captured in the class structure by creating aggregation relationship between classes of auxiliary and main processes. This relationship and the classification of main processes are represented in the class structure as shown in figure 5.11.

![Class structure to represent the classification of main processes and the relationship of auxiliary and main processes](image-url)
Chapter 5

The detailed class structure used to capture function to process transformation knowledge is explained below using examples of injection moulding forming process and fixing assembly process.

![Class structure diagram](image)

**Figure 5.12 Class structure to capture mouldability to forming & Hole and shaft assembly to fixing transformation knowledge**

Figure 5.12 shows the detailed class structure to capture the main function to process transformation knowledge specially for shaping by injection moulding and fixing of hole and shaft shapes. In this structure, the mouldability function to forming process transformation knowledge is captured with the class "Mouldability to Forming". The attributes of this class help to capture and tabulate different possible occurrences and conditions. For example, the attributes of the "Mouldability to Forming" transformation knowledge class, such as number of parting planes, position of mouldability feature with respect to parting plane, and mouldable wall types, are shown in figure 5.13.

![Mouldability to forming class and its attributes](image)

**Figure 5.13 Mouldability to forming class and its attributes**
Figure 5.14 shows an example of the populated mouldability function to forming process transformation knowledge using the mouldability to forming transformation knowledge class. This transformation knowledge helps to decide which sides of the mouldability feature information need to be extracted from the mouldability feature to construct the mould cavity and core.

<table>
<thead>
<tr>
<th>No. of PP</th>
<th>Product Type</th>
<th>Mould Type</th>
<th>Wall Type</th>
<th>Position WRT PP</th>
<th>Inclination WRT PP</th>
<th>Required Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Primary</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
<td>Outside</td>
</tr>
<tr>
<td>Option 2</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Transition</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
<td>Outside</td>
</tr>
<tr>
<td>Option 3</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Transition</td>
<td>Below-NonTouching</td>
<td>Not Applicable</td>
<td>Outside</td>
</tr>
<tr>
<td>Option 4</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY</td>
<td>Primary</td>
<td>Below-NonTouching</td>
<td>Parallel</td>
<td>Outside</td>
</tr>
<tr>
<td>Option 5</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Primary</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
<td>InSide</td>
</tr>
<tr>
<td>Option 6</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Transition</td>
<td>Below-Touching</td>
<td>Perpendicular</td>
<td>InSide</td>
</tr>
<tr>
<td>Option 7</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Transition</td>
<td>Below-NonTouching</td>
<td>Not Applicable</td>
<td>InSide</td>
</tr>
<tr>
<td>Option 8</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE</td>
<td>Primary</td>
<td>Below-NonTouching</td>
<td>Parallel</td>
<td>InSide</td>
</tr>
</tbody>
</table>

Figure 5.14 Example of mouldability function to forming process transformation knowledge

Similarly, hole and shaft assembly function to assembly process transformation knowledge can be captured with the class "Hole and Shaft Assembly", that is a sub class of the fixing function to process transformation knowledge. This "Hole and Shaft Assembly" class is shown in figure 5.15 with its attributes. Also figure 5.16 shows an example of populated transformation knowledge of the assembly function to assembly process that can be applicable for cylindrical shape assembly.

![Hole and Shaft Assembly class and its attributes](image)

Figure 5.15 Hole and shaft assembly class and its attributes

Hence, the transformation knowledge of the function to main process can be captured in the knowledge base with the appropriate class structure. In a similar manner, auxiliary process transformation knowledge can be captured in the knowledge base. The following section of this chapter explains the class structure to capture function to auxiliary process transformation knowledge.
### Chapter 5

#### Diameter Transition Fits

<table>
<thead>
<tr>
<th>Diameter</th>
<th>Transition Fits</th>
<th>Push Fit</th>
<th>Tight Assembly Fit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hole (H7)</td>
<td>Shaft (k6)</td>
</tr>
<tr>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0.010</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0.012</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0.015</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>0</td>
<td>0.018</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>0</td>
<td>0.021</td>
</tr>
<tr>
<td>30</td>
<td>50</td>
<td>0</td>
<td>0.025</td>
</tr>
<tr>
<td>50</td>
<td>80</td>
<td>0</td>
<td>0.030</td>
</tr>
</tbody>
</table>

Figure 5.16 Example of assembly function to process transformation knowledge

### 5.4.1.3 Function to Auxiliary Process Transformation Knowledge

As explained earlier, the main processes can be performed with the help of auxiliary processes. This auxiliary process transformation knowledge has to be captured in the knowledge base. Figure 5.17 shows the detailed class structure to capture the transformation knowledge of the auxiliary processes, such as feeding, cooling, and venting. This class structure is expanded up to the level to capture the feeding and cooling auxiliary processes of the injection moulding.

![Figure 5.17 Example of the structure to capture the auxiliary processes](image)

97
Hence the transformation knowledge of the function to main and auxiliary process can be captured in the knowledge base to support the transformation of information between views. As explained in section 5.3, view specific knowledge is also necessary in addition to the transformation knowledge to support information integration between multiple viewpoints. The following section of this chapter explains the class structure to capture view specific knowledge.

5.5 Knowledge Base Structure to Capture View Specific Knowledge

The life cycle of a product has several phases including concept, design, manufacturing, operation and maintenance, and disposal. Each of these phases needs to be considered in several viewpoints. These viewpoints have constraints, and limitations to perform their viewpoint activities. Knowledge of the ability, constraints and limitations of each viewpoint can be classified as view specific knowledge. This view specific knowledge needs to be captured in the knowledge base to verify the specific requirements against the view specific limitations. Figure 5.18 shows the class structure to capture the view specific knowledge of different stages of the product life cycle.

![Figure 5.18 Structure to capture view specific knowledge of different stages of product life cycle](image)

In the research reported in this thesis, the manufacturing stage of view specific knowledge is taken as an example to explore the influence of view specific knowledge in the integration of multi viewpoint information. The following section explains the detailed class structure to capture the manufacturing view specific knowledge.
5.5.1 Manufacturing View Specific Knowledge

The view specific knowledge of the manufacturing stage captures the limitations and constraints of different methods of manufacturing such as machining, injection moulding, and welding. Figure 5.19 shows the class structure to capture the view specific knowledge especially for the limitation of different machining methods.

![Diagram showing the class structure to capture machining limitation, which is a view specific knowledge.](image)

The view specific knowledge of the manufacturing stage captures the limitations and constraints of different methods of manufacturing such as machining, injection moulding, and welding. Figure 5.19 shows the class structure to capture the view specific knowledge especially for the limitation of different machining methods.

![Graph showing the feasible and infeasible regions for turning, boring & reaming.](image)

Figure 5.20 Example of machining limitation view specific knowledge in Graph format
For example, the possible machineable tolerances of turning, boring and reaming methods are given in the graph of diameter (mm) versus tolerances (μm), which is shown in figure 5.20. If the tolerance of that machining feature is in the feasible region of the graph the above-mentioned machining processes can be used to machine a machining feature. On the other hand, if it is in the infeasible region, it is not possible to machine up to the required tolerances. This is knowledge of the limitation of the machining processes and needs to be captured as view specific knowledge in the knowledge base. This machining limitation view specific knowledge is in graph format but it can be represented as a polynomial function like "Machineable tolerance = Ax^2 + Bx + C", where A, B & C are coefficients, which are dependent on the machining process. This polynomial function can be captured in the knowledge base by a "Machining Limitation" class with attributes such as machining method, machineable tolerance, and coefficients, as shown in figure 5.21. Hence, the graph format of view specific knowledge can be captured in the integration knowledge base with an appropriate class structure.

The above example of view specific knowledge is represented in the format of graph, but it is only a type of knowledge representation. The knowledge can be classified into two main kinds: explicit and implicit. Further, the explicit knowledge can be represented in different format: tables, graphs, story telling, etc (Guerra, 02). However, capturing and managing different kinds of knowledge in a knowledge model falls out of the scope of the research reported in this thesis. Further, the methods and techniques of knowledge capturing and management is being researched by the international research community.
Hence, integration knowledge can be captured in the knowledge base as transformation and view specific knowledge. This integration knowledge assists the integration between multiple viewpoint information that is captured in the information layer. The next section explains, with examples, how the knowledge layer supports information integration between views.

5.6 Examples of Utilising Combined Information and Knowledge Layers for Information Integration Between Multiple Views

Information integration between functional, assembly, and machining views of the cavity insert and sprue bush are taken as examples to explain the influence of the knowledge layer in information integration.

5.6.1 Transformation of Cavity Insert & Sprue Bush Assembly Requirements to Assembly Functions

The cavity insert and sprue bush assembly requirement information, which is in the information layer, needs to be transformed into assembly functions with the help of the "usage to fit function" transformation knowledge, which is captured in the knowledge layer. Examples of usage to fit function transformation knowledge are tabulated in figure 5.22.
<table>
<thead>
<tr>
<th>Fit Function</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slack running fit</td>
<td>Used to give flexibility under load, easy assembly or a close fit at elevated working temperatures</td>
</tr>
<tr>
<td>Loose running fit</td>
<td>Used for gland seals, loose pulleys and very large bearings</td>
</tr>
<tr>
<td>Easy running fit</td>
<td>Used for widely separated bearings or several bearings in line</td>
</tr>
<tr>
<td>Normal running fit</td>
<td>Suitable for applications requiring a good quality fit that is easy to produce</td>
</tr>
<tr>
<td>Sliding and location fit</td>
<td>Not normally used for continuously running bearings unless load is slight. Suitable or precision sliding and location</td>
</tr>
<tr>
<td>Location fit</td>
<td>Suitable for many non-running assemblies</td>
</tr>
<tr>
<td>Push fit</td>
<td>Used for location fits when a slight interference, which eliminates movement of one part relative to other, is an advantage</td>
</tr>
<tr>
<td>Tight assembly fit</td>
<td>Used when the degree of clearance that can result from a push fit not acceptable</td>
</tr>
<tr>
<td>Press fit</td>
<td>Ferrous parts are not over strained during assembly or dismantling</td>
</tr>
<tr>
<td>Heavy press fit</td>
<td>Mainly used for permanent assemblies</td>
</tr>
</tbody>
</table>

Figure 5.22 Example of usage to fit function transformation knowledge

In this particular case, the usage of the insert and bush assembly locate the assembly parts, avoid leakage when the plastic is injected, etc. According to the usage to fit function transformation knowledge, these requirements can be obtained by introducing either push fit or tight assembly fit function. So the requirements can be transformed to function with the help of the knowledge layer. The next section of this chapter explains how the fit function information can be transformed to the assembly view information.

5.6.2 Transformation of Cavity Insert & Sprue Bush Assembly Fit Functions to Assembly Features

In a similar manner as explained above, the fit function can be transformed to the assembly view features with the help of the function to process transformation knowledge as shown in figure 5.23. In this particular case, a push fit of diameter 21 mm is transformed to required a tolerance range 0-21µm and 2-15µm for the hole and shaft respectively.
As explained earlier, transformation between views is not significant as validation is also required for feasible information integration between views. The next section explains how the assembly feature information can be transformed to the machining features and validated with machining view specific constraints.

### 5.6.3 Integration Between Assembly and Machining Views of Sprue Bush

In a similar manner to that explained in the above sections assembly features can be transformed to machining features. The transformed machining feature needs to be validated with machining view specific knowledge, which is captured in the knowledge layer.
In this particular case, the transformed machining feature of the sprue bush is an outer round with minimum and maximum tolerances of 2 and 15 respectively. The capability of machining methods needs to be checked against the tolerances to verify whether the outer round can be produced by the machining method or not. If the required tolerances are within the feasible region, the product can be machined according to the requirements and the information can be transformed to the machining view. On the other hand, if the required tolerances are in the infeasible region, this limitation needs to be reported to the user. Then the user has to propose an alternative solution to overcome the limitation.

So the combination of information and knowledge layers helps to perform information integration between views. However, these two layers alone are not significant to perform information integration, because information integration activities require particular groups of integration knowledge from the knowledge base. In the knowledge base several groups of integration knowledge are captured to support the integration of multiple viewpoints. The problem is how to identify the required group of knowledge to support information integration between specific viewpoints. To overcome this limitation, knowledge links are introduced to identify the required groups of knowledge instances from the knowledge base. The detailed explanation of knowledge links and related activities are explained in chapter 6 of this thesis.

5.7 Flexible Information Integration Between Multiple Views With the Help of a Knowledge Layer & Temporary Storage

The above three sections explain how the information integration can be performed with the help of the knowledge layer. In these three sections, information integration between views is explained in a successive manner, but flexible information integration requires concurrent integration between multiple views. If the information and knowledge structures are defined appropriately, as reported in this thesis, concurrent information integration can be performed. In the concurrent integration approach, transformed information is validated with view specific considerations and is captured in a temporary storage until all required integrations are completed satisfactorily. Then all the transformed information is populated in the product model.
If a transformed view's information is not accepted by its viewpoint considerations, that would be reported to the user and the user's instructions are to be followed to populate the product model. This concurrent integration has been proved in the experimental system and reported in chapter 7.

5.8 Summary of Integration Knowledge

The integration knowledge base class structure has to satisfy the following issues to support flexible information integration between views.

1) Transformation and view specific knowledge are the two main kinds of integration knowledge. The class structure of the knowledge base should have two branches to capture these two main kinds of knowledge.

2) Requirements to concepts, concepts to function, and function to process are the three main groups of transformation knowledge. The transformation knowledge class needs to have three sub classes to capture the transformation knowledge.

3) Function to process transformation knowledge can be derived from the process model, because the function to process transformation knowledge depends on the process characteristics.

4) Processes can be divided into two main groups: the main and auxiliary processes. So the function to process transformation knowledge has two main classifications: function to main process and function to auxiliary process. Hence the integration knowledge base class structure should have two branches of subclasses in the function to process transformation knowledge.

5) View specific knowledge contains the knowledge of ability, constraints and limitations of viewpoint activities. View specific knowledge can be categorised according to different stages of product life cycle. Product life can be divided into five main stages: concept, design, manufacturing, operation and maintenance, and disposal. So the knowledge base structure should have five sub classes in the view specific knowledge.

Top-level classes of integration knowledge structure are shown in figure 5.24. The detailed structure with attributes of each class is given in appendix B. The populated knowledge that is used in the research reported in this thesis is also tabulated in appendix C.
Figure 5.24 Top-level classes of Integration knowledge
6. KNOWLEDGE LINKS TO SUPPORT MULTI VIEWPOINT INFORMATION INTEGRATION

6.1 Introduction

This chapter describes the knowledge links and the integration activities. The knowledge and information layers communicate with each other with the help of knowledge links. Knowledge links contain knowledge about the repositories of information and knowledge groups in the databases. The initial part of this chapter highlights the requirements of the knowledge links. The final part describes how knowledge links support the information integration between views.

6.2 Requirements for Knowledge Links

Integration knowledge supports information integration between multiple viewpoints. The knowledge layer, which is detailed in chapter 5, is defined to capture the integration knowledge. Integration knowledge needs to be defined in several groups that can be applicable for different product development environment as well as multiple views. Hence, the integration knowledge is a collection of several groups of transformation and view specific knowledge. To perform integration between specific viewpoints of particular environment needs specific groups of transformation and verification knowledge that are applicable for the selected viewpoints of the particular environment.

For example, an injection moulded plastic product design and manufacture environment requires integration knowledge groups such as mouldability to shaping transformation knowledge, geometry shape to machining features transformation knowledge, machining limitation knowledge, and mouldability limitation knowledge. Figure 6.1 shows an example of integration knowledge groups captured in a knowledge base that are applicable for an injection-moulded design and manufacture environment.
The required groups of knowledge need to be identified and retrieved from the knowledge base to perform information integration between specific viewpoints. For example, the information transformation between the shaping functional view to the machining view requires geometry shape to machining feature transformation knowledge. Once the information is transformed into machining features it needs to be validated with machining limitation knowledge. Identifying required groups of knowledge from the knowledge base is a complex task of the integration of multiple viewpoints information, as shown in figure 6.2, because the integration knowledge base contains multiple groups of integration knowledge. Therefore, the knowledge of the locations of each group of the integration knowledge in the knowledge model needs to be captured to support the integration activities. This limitation can be overcome by introducing knowledge links. The detailed explanation of the usage of the knowledge link is given in the following sections of this chapter.
Figure 6.2 Limitation of information integration between views, even though information and knowledge layers are present

6.3 Knowledge Links that Relate Databases

Knowledge links, which have knowledge of the locations of the integration knowledge groups in the knowledge base, have been used to support the access of the required knowledge from the knowledge base. Knowledge links were introduced initially by Costa (Costa, 00) and defined as “it holds information and knowledge about specific pieces of the product data model and provide the means to retrieve
required information”. In his approach, knowledge links are used to link two information databases but the research reported in this thesis uses knowledge links to access the required knowledge from a knowledge base. The next section describes the knowledge of knowledge links that support the integration between multiple viewpoint information integration.

6.4 Knowledge of Knowledge Links to Support Information Integration between Multiple Viewpoints

Knowledge links need to have knowledge to find and access the required groups of integration knowledge from the knowledge base to support the information integration activities between specific views.

![Diagram of integration activity and relationship knowledge in information and knowledge layers framework](image)

Figure 6.3 Example of integration activity and relationship knowledge in information and knowledge layers framework

For example, to perform information integration between the shaping functional and the machining views requires shaping function to machining feature integration knowledge. In this particular example, knowledge links need to have knowledge
about the relationships between the integration activity of the shaping function and machining views and its required integration knowledge. In this example, the integration knowledge is the transformation knowledge between the shaping function to machining feature and machining view specific knowledge. In addition to the relationship knowledge of the integration activity and integration knowledge, the knowledge link needs to have the knowledge about the locations of the required integration knowledge. Figure 6.3 shows the knowledge links that relate the integration activity and required integration knowledge. Hence, the knowledge link contains knowledge of the relationships of integration activities, and its required integration knowledge, in addition to the locations of the integration knowledge groups in the knowledge layer.

Now the question arises as to where to keep the knowledge of the knowledge links. This knowledge can be kept in either the database or the integration applications. In the research reported in this thesis information and integration knowledge are captured in two separated but related databases. If the relationship knowledge were captured in either of these databases, the contents of these databases would be swamped with relationship knowledge. For this reason, knowledge of knowledge links is defined to be captured in the integration application, i.e. programming code.

It has been identified and highlighted in the research reported in this thesis that capturing of knowledge in an application limits the flexibility of the information integration in a multiple viewpoint system. Nevertheless, the proposed knowledge links contain knowledge of the relationships of integration activities and its required integration knowledge in addition to the locations of integration knowledge groups in the knowledge model. However, the main participation of the knowledge link in an integration activity is to offer knowledge about the location of integration knowledge in the integration knowledge model. Further, the populated integration knowledge in the knowledge model can be altered easily, since it is captured in a database. Hence, the flexibility of a multiple viewpoint information integration framework will not be limited by capturing knowledge of the knowledge links in the application.
6.5 Summary

Information integration between multiple views can be performed with supporting integration knowledge that is captured in the knowledge layer. To perform the integration, the required groups of knowledge need to be accessed from the knowledge base. Knowledge links provide support in order to identify and retrieve the required groups of knowledge from the knowledge layer. Knowledge links should have the knowledge about the relationships between integration activities and the required knowledge groups with their repositories in the knowledge base. It has been proposed that this knowledge of knowledge links is captured in the integration application, i.e. programming code.
7. THE EXPERIMENTAL SYSTEM IMPLEMENTATION

7.1 Introduction

This chapter describes the experiments conducted to explore the class structures of information and knowledge layers and also demonstrate the extent to which the framework provide support for multiple viewpoint information integration. Section 7.2 presents the scope, and experiments of the developed experimental software system and a description of general aspects that were considered for its design and implementation. Sections 7.3 to 7.8 present experiments conduct in the research.

7.2 Design and implementation of the experimental system

7.2.1 Scope and Experiments

The experiments conducted during the research have focused on the definition of an ontology that supports the integration of multiple viewpoints. This includes the definition of information/ knowledge layers as well as the use of knowledge links. In addition, these experiments show that the ontology proposed in this research is a valid basis for a flexible integration of multiple viewpoint information. The experiments conducted in this research are:

1) To explore the capability of the class structure defined for the product model to capture five main groups of product related information: purpose, phase, architecture, characteristics, and views.
2) To explore how the class structure of the information layer supports to the capture of the information of multiple views of products.
3) To explore the capability of the class structure defined for the knowledge layer to capture transformation and view specific knowledge.
4) To explore how the knowledge layer provides support to the information integration between multiple views of a single product.
5) To explore how both information and knowledge layers support consistency maintenance in all information views, if a view feature is modified.
6) To explore how information and knowledge layers support information integration between views of related products.
Experiments 1 & 2 are realised in sections 7.3 & 7.4 using the example of a cavity insert - sprue bush assembly. Experiment 3 is realised in section 7.5 using the integration knowledge of a design and manufacture environment of injection moulded plastic product. Experiments 4 and 5 are realised in sections 7.6 and 7.7 respectively using the example of a cavity insert- sprue bush assembly. Experiment 6 is realised in section 7.8 using the examples of a rectangular shape plastic container in combination with mould parts.

7.2.2 The Experimental System Environment and Description

The research ideas presented in chapters 3 to 6 of this thesis have been validated using the experimental system. Following the MIM project approach, two main elements have been perused in the experimental system implementation. The first element is the structures for the database, which have been realised using the ObjectStore database. The second element is software applications, which have been realised by the programming environment Visual C++.

The structures of the database have been represented using the UML notation. Objects involved in the integration of multiple viewpoints of design for manufacture of products have been identified with their relationships and structured using UML notation. The developed structures of the information and the knowledge layers were discussed in chapters 4 & 5 respectively. The full structure of information and knowledge models are given in appendix B. The UML notations that were used to represent the objectives and the kinds of relationships between objects are presented in Appendix A.

In the software application element, three main steps have been identified for the experimental system development, as follows:

1) The application interface for information layer. This concerns the population, retrieving, and viewing of information in the information layer.

2) The application interface for knowledge layer. This concerns the population, retrieving, and viewing of knowledge in the knowledge layer.

3) The interfaces for the applications of information integration, and the consistency maintenance between the multiple viewpoints of products.
These three steps have guided the experiments of the research. Figure 7.1 illustrates the general structure of the experimental system.

![Diagram of Experimental System Structure](image)

**Figure 7.1 The general structure for experimental system development**

The information layer applications have been designed to capture information related to an injection moulded product development environment. The major part of the knowledge layer applications have been designed to populate integration and view specific knowledge for the injection moulded product development environment similar to the information layer applications, but in an automatic way since, the integration of multiple viewpoints of injection moulded products require huge number of knowledge instances. However, applications of knowledge layer have the functionality to modify populated knowledge instances. Applications designed for integration tasks provide more flexibility in terms of using information and knowledge models for an injection moulded product development environment.

### 7.2.3 Visualisation of the Results/ Contents of Database

The results and the contents of databases have been visualised in two ways:

1) with the dialogs of the experimental system application developed in visual C++
2) with the ObjectStore inspector interface that allows the visualisation of the data stored in the ObjectStore database.
7.3 Product Model Structure to Capture Product Information

7.3.1 General Description

As highlighted in chapters 2 and 4, product related information has been classified into five main groups: phase, purpose, characteristics, architecture, and views. In addition to this, a product model structure was presented in chapter 4 to capture the five main groups of product information. The top-level class structure of the product model is given in figure 7.2. This section explains the experiment conducted to show the product model structure is able to capture five main groups of product related information.
The cavity insert-sprue bush assembly, which is shown in figure 7.3 has been taken as a sample product to show that the information layer is capable to capture all five groups of a product information. In addition, the same cavity insert-sprue bush assembly product has been used in the experiments described in sections 7.3 to 7.7 of this chapter.

7.3.2 Populating Product and Its Related Information Instances to the Product Model

To capture a product and its related information in the information layer, a product model needs to be created first. Product instances can be populated in the created product model. As mentioned earlier, in this experiment the cavity insert-sprue bush assembly has been used as a sample product instance. Figure 7.4 shows the Insert - Bush product and its related information instances into five groups: phase, architecture, purpose, characteristics, and views. The product and its related information instances have been captured in the product model with multiple instances. Section 4.4.2 explains the multiple instances approach, which requires multiple instances to represent a product and its related information in a product model.

Figure 7.4 Example of product information that has been populated in the product model
Chapter 7

The user interface shown in figure 7.5 has been used to populate product instances and their related view instances. The product population dialog has been designed to populate view instances with product instances in order to prevent run time errors of the experimental system. This dialog reminds the user to populate views instances while creating product instances.

![Populating Product Instance](image)

Figure 7.5 User interfaces to populate product instances

*Feeding sub function, assembly sub function, assembly and machining* views instances have been populated with the insert-bush product instance, as shown above. The rest of the product related information has been populated to the product model by the help of separate interfaces as shown in figure 7.6.

Material property has been taken as an example for product characteristics and populated in the product model by characteristics population interface. *Insert – Bush* purpose information is populated in the product model by purpose population dialog. Similarly architecture and phase information of *Insert – Bush* have been populated in the product model by architecture and phase interfaces respectively. Hence, the interfaces shown in figure 7.6 help to populate product related information in the product model.
This experiment shows that the structure defined for the product model is capable of capturing the product instance with their related information instances. However, this experiment doesn’t show how the product and its related information have been kept in the product model database. The following section explains the second part of the experiment, which has been conducted to visualise the populated information of the product model database.

7.3.3 Visualising the Product and Its Related Information Instances

The populated product and its related information instances can be visualised either by Visual C++ dialogs, or by ObjectStore inspector. The ObjectStore inspector approach has been selected since it can visualise instances with the relationships. Figure 7.7 shows populated Insert – Bush product instance and its five main groups of related information instances. In addition, the relationships between these instances are clearly shown in this ObjectStore inspector diagram. Hence, the Insert – Bush product instance and its five main groups of related information instances have been captured in the product model. Procedures to perform population and visualisation of
Chapter 7

the instances of products and their related information are given in appendixes D-1 to D-5.

Experiment described in this section shows that the product model structure defined for the research reported in this thesis is capable of capturing product instances with their five main groups of product related information. In addition, the second part of the experiment shows that the product instance being a common instance to relate its related main groups of information. However, this experiment doesn’t show how information of multiple views instances, which is the important part of the research, can be populated and visualised from a product model. The next section explains the experiment conducted to show that multiple views information could be captured and retrieved with the structure defined for the product model.
7.4 Experiment to Show that the Product Model Structure is Able to Capture Multiple Viewpoint Information

7.4.1 General Description

Chapter 4 presented a structure for a product model to support the integration of multiple viewpoints information. The product model structure has been defined with several sub view classes to capture the multiple viewpoints information. In addition, the Views class has related with a Representation Language class by an aggregation relationship, as shown in figure 7.8, to capture view specific representations of products. This section explains the experiment conducted to show that the product model structure is capable of capturing multiple viewpoints information.

![Figure 7.8 Class structure to capture multiple viewpoints information and its viewpoint representation languages](image)

7.4.2 Populating Multiple Views Information of a Product In the Product Model

As mentioned earlier, cavity insert- sprue bush assembly has been taken as a sample product for the experiment described in this section. The functional, assembly, and machining views of the product have been taken into account for the experimental system. The functional view of the cavity insert - sprue bush assembly has been further divided into sub functional views: feeding, assembly, shaping, and venting. The feeding sub functional, the assembly sub functional, the machining, and the assembly views have been selected as examples for the experiment described in this chapter to explore the integration of multiple viewpoints of a cavity insert-sprue bush assembly since they are highly inter related to each other.
The feeding functional view considerations such as size of the gate, shape of the product, number of gates, gate position, injection pressure, and mould material influence the definitions of the external dimensions of the sprue bush. These dimensions of the sprue bush influence the dimensions of the cavity insert hole in addition to the assembly functional view requirements. The sub functional views such as feeding, and assembly contain information about the functional requirements, but the assembly view activities of the cavity insert and sprue bush require assembly feature information. Similarly, machining view information of the cavity insert and the sprue bush is necessary for the machining view activities. Figure 7.9 shows an example of these feeding sub functional, assembly sub functional, assembly, and machining views information. A 21mm diameter sprue bush fitted to a cavity insert by push fit has been taken as an example of cavity insert – sprue bush assembly for this experiment. The tolerances of cavity insert – sprue bush have been decided with the help of assembly, and machining views considerations. These views information have been captured in the product model to perform the experiment described in this section. In addition, these views information have been used for the experiments described in sections 7.6, 7.7 and 7.8 of this chapter.

Multiple views information of a product can be populated to the product model by either user interfaces or transformation from the information of other populated views. In this experiment the information of sub functional views have been populated using user interfaces while the information of the assembly and the machining views have

![Figure 7.9 Multiple viewpoint representations of cavity insert- sprue bush](image-url)
been populated by transformation from the information of sub functional views. There are several possible methods to feed plastic to the injection mould cavity in the feeding functional point of view. However, the objective of the experimental system is to show the integration of multiple viewpoints of products. Hence, feeding via a fan gate and sprue bush has been taken as a feeding method since sprue bush having strong relationship with assembly and machining views. This strong relationship provides suitable environment to explore the integration between multiple viewpoints. The external dimensions of the sprue bush have been decided by considering several feeding functional view considerations.

Figure 7.10 Dialogs to populate sub functional view information
Therefore, the external diameter of the sprue bush is 21 mm, which is as an example for the feeding functional information, has been populated in the product model using user interfaces shown in figure 7.10. Similarly, the push fit information, which is the assembly functional view information, has been populated to the product model using user interface shown in figure 7.10.

Populated feeding and assembly sub functional views information of *cavity insert-sprue bush assembly* have been transformed to the assembly and the machining views with the help of integration knowledge and populated in the product model. The transformation from sub functional views to assembly and machining views is detailed in section 7.6 of this chapter. Hence, feeding sub functional, assembly sub functional, assembly and machining views information of the *cavity insert-sprue bush assembly* have been populated in the product model.

Hence, the experiment shows that multiple views information of a product can be captured in the product model. However, it doesn’t show how these captured multiple viewpoint information can be visualised. The next section of this chapter explains methods to visualise captured multiple viewpoint information.

### 7.4.3 Visualising Multiple Views Information of a Product

Populated sub functional view information can be visualise with the same dialogs that are used to populate the information. However, these dialogs are only able to show the information of a single viewpoint of a selected product. Hence, a dialog, as shown in figure 7.11, has been design to show the information of the feeding sub functional, the assembly sub functional, the assembly, and the machining views of populated products in a product model.

Hence, the experiment described in this section shows that the class structure defined for the product model is capable of capturing and retrieving multiple viewpoint information of products. Appendixes D-3 and D-4 provide detailed procedures for populating and visualising the information of multiple views. However, this experiment doesn’t show how the structure of the product model supports the information integration of multiple viewpoints. Integration knowledge is necessary for
the integration of multiple viewpoints and captured in separate knowledge model as explained in chapters 3 and 5. The capabilities of the integration knowledge model are tested in the experimental system implementation and described in the following sections of this chapter.

Figure 7.11 Visualising multiple views information of cavity insert sprue bush assembly

7.5 Knowledge Layer Structure to Capture the Integration Knowledge

7.5.1 General Description

The knowledge layer, which is an integration knowledge model, is defined to capture integration knowledge. Integration knowledge is composed of transformation and view specific knowledge. The top-level class structure of the integration knowledge model is given in figure 7.12.
The experiments described in this section are aimed at showing that the integration knowledge model structure is capable of capturing transformation and view specific knowledge. The tabulated relationships between fit function and hole-shaft tolerances, which have extracted from British Standard Selected ISO Fits- Hole basis (BS 4500, 1969), have been used as an example of the assembly functional view to assembly view transformation knowledge. This transformation knowledge has been used in order to explore the capability of the integration knowledge model to capture transformation knowledge. Similarly, the relationships of machining process and their tolerance quality (BS 4500, 1969) have been taken as an example of machineable limitation knowledge to explore the capability of the integration knowledge model to capture view specific knowledge. Detail explanations of these experiments are given in the following sub sections of this chapter.

7.5.2 Transformation Knowledge and the Structure of the Knowledge Model

Transformation knowledge provides support to transform information from the functional view to the other views, as explained in chapters 3 and 5. An example of relationships between the fit function and assembly tolerances are provided in the table shown in figure 7.13. These relationships knowledge have been used to transform information between assembly sub function and assembly views. The next sub section describes the experimental procedures to populate the transformation knowledge.
Chapter 7

Diameter Clearance Fits
Loose Running

<table>
<thead>
<tr>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0.060</td>
<td>-0.120</td>
<td>-0.060</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.075</td>
<td>-0.145</td>
<td>-0.070</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0.090</td>
<td>-0.170</td>
<td>-0.080</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>0.110</td>
<td>-0.205</td>
<td>-0.095</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>0.130</td>
<td>-0.240</td>
<td>-0.110</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>0.160</td>
<td>-0.280</td>
<td>-0.120</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>0.160</td>
<td>-0.290</td>
<td>-0.130</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>65</td>
<td>0.190</td>
<td>-0.330</td>
<td>-0.140</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>80</td>
<td>0.190</td>
<td>-0.340</td>
<td>-0.150</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>100</td>
<td>0.220</td>
<td>-0.390</td>
<td>-0.170</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.13 Example of the relationships between fit function assembly tolerances

7.5.2.1 The Population of Transformation Knowledge

Figure 7.14 Main dialog for knowledge population
A knowledge model needs to be created first in order to capture the integration knowledge. Once the knowledge model is created, the transformation knowledge can be populated in the integration knowledge model using the interface as shown in figure 7.14, which is the main interface to populate the integration knowledge. Clicking the button named “MANUAL” of the hole & shaft assembly fixing transformation knowledge of figure 7.14 pops up the interface to capture the assembly sub function to the assembly view transformation knowledge. The interface that has been used to populate the relationships between fit function and assembly tolerances for cylindrical shapes is shown in figure 7.15.

![Figure 7.15 Dialog to populate fit function to assembly view transformation knowledge](image)

The fit function to the assembly view transformation knowledge, shown in figure 7.13, is only a small part of relationship knowledge. This table contains the relationship knowledge for loose running fit for 0-100 mm nominal diameter holes and shafts. Nevertheless, the transformation knowledge needs to consider all possible
options for the transformation between assembly functions and assembly views. For example, all kinds of running fits, and all possible diameter ranges need to be considered. Each of these options can be captured in the integration knowledge model by populating separate instances of knowledge.

Similarly, the shaping function to the manufacturing process transformation knowledge, such as mouldability to shaping, shaping to geometry, and geometry to machining can be populated in the knowledge model. Examples of the transformation knowledge used in this research are documented in appendix C. Experiment described in this section is aimed to show that the transformation knowledge can be captured in the integration knowledge model. However, this experiment does not visualise the populated transformation knowledge in the integration knowledge model. Following sub section of this chapter explains the procedures to visualise the populated transformation knowledge.

### 7.5.2.2 Visualising Populated Transformation Knowledge

![Visualising Populated Transformation Knowledge](image)

*Figure 7.16 Interface and steps to visualise fit function to assembly view transformation knowledge*
Figures 7.16 and 7.17 show the dialogs that have been used to visualise the fit function to assembly view transformation knowledge. By clicking HoleAndShaftAss of viewing dropdown menu, pops up a dialog named Select Fit Type and Fit Function. This dialog facilitates to visualise relationships between assembly tolerances and diameters of selected fit function via another dialog. In addition, the same set of dialogs can be used to modify the populated fit function to assembly view transformation knowledge. Similarly, other populated transformation knowledge can be visualised using different visualising dialogs. The detail procedures of populating, visualising and modifying transformation knowledge are given in appendixes D-6 and D-7.

The experiment described in this section shows that the structure defined for the integration knowledge model is capable of capturing and retrieving transformation knowledge. Nevertheless, the integration knowledge contains two main groups as transformation and view specific knowledge. Hence, the integration knowledge model needs to have the capacity to capture view specific knowledge in addition to the transformation knowledge. An experiment has been conducted to show the capability of the structure of integration knowledge model to capture view specific knowledge. Next sub section of this chapter describes this experiment briefly.
7.5.3 View Specific Knowledge and the Structure of the Knowledge Model

View specific knowledge helps to validate the transformed information as explained in chapters 3 & 5. Figure 7.18 shows an example of the relationships between the machineable tolerance and the diameter of hole and shaft for turning, boring and reaming processes, in the form of graph. This graph of relationship has been extracted from relationships of machining process and their tolerance quality (BS 4500, 1969). The graph shows feasible and infeasible regions of machining processes. Hence, this relationship has been used as view specific knowledge for machineable features to validate the machining view information. Therefore, the machineable process limitation knowledge is taken as an example to show that the knowledge model is capable of capturing the view specific knowledge. The graph format of knowledge can be represented to the nearest polynomial as $AX^2+BX+C$ and captured in a database, as explained in chapter 5.5.1. Similarly, the relationship knowledge of tolerance and diameter for different machining processes can be represented in the form of polynomial with appropriate coefficient values. Hence, the limitation of different machining processes can be captured in the knowledge layer by capturing the values of the polynomials. Similarly, the relationships between machining processes and machineable features have been captured in the knowledge layer. Following section describes the experimental procedures to populate view specific knowledge in an integration knowledge model.
7.5.3.1 Populating View Specific Knowledge

Figure 7.19 provides the table of different machining processes and the related polynomial coefficients of machineable tolerances. In addition, the figure 7.20 shows the table of machining method and the related feature machining process. The relationships shown in these tables have been populated in a knowledge model, since they are examples of the view specific knowledge.

<table>
<thead>
<tr>
<th>Machining Method</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>BORING</td>
<td>0.0103</td>
</tr>
<tr>
<td>CENTER LATHE BORING</td>
<td>0.006</td>
</tr>
<tr>
<td>DRILLING</td>
<td>0.0188</td>
</tr>
<tr>
<td>FINE BORING</td>
<td>0.0009</td>
</tr>
<tr>
<td>FINE TURNING</td>
<td>0.0027</td>
</tr>
<tr>
<td>GRINDING</td>
<td>0.0019</td>
</tr>
<tr>
<td>MILLING</td>
<td>0.0136</td>
</tr>
<tr>
<td>PLANNING</td>
<td>0.0136</td>
</tr>
<tr>
<td>REAMING</td>
<td>0.006</td>
</tr>
<tr>
<td>ROUGH BORING</td>
<td>0.0188</td>
</tr>
<tr>
<td>ROUGH TURNING</td>
<td>0.0188</td>
</tr>
<tr>
<td>SLOTTING</td>
<td>0.0136</td>
</tr>
<tr>
<td>TURNING</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Figure 7.19 Example of machining methods and machining tolerance limitation polynomial coefficients

<table>
<thead>
<tr>
<th>Machining Method</th>
<th>Feature Machining Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>CENTER LATHE BORING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>DRILLING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>FINE BORING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>FINE TURNING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>GRINDING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>MILLING</td>
<td>RECTANGULAR POCKET MACHINING</td>
</tr>
<tr>
<td>PLANNING</td>
<td></td>
</tr>
<tr>
<td>REAMING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>ROUGH BORING</td>
<td>HOLE MACHINING</td>
</tr>
<tr>
<td>ROUGH TURNING</td>
<td>OUTER ROUND MACHINING</td>
</tr>
<tr>
<td>SLOTTING</td>
<td></td>
</tr>
<tr>
<td>TURNING</td>
<td>OUTER ROUND MACHINING</td>
</tr>
</tbody>
</table>

Figure 7.20 Example of machining method and the related feature machining process
The above shown examples of view specific knowledge has been populated in an integration knowledge model by clicking FEATURE MACHINING AUTO button of Knowledge Population dialog as shown in figure 7.21.

![Figure 7.21 Population of machining view specific knowledge](image)

This experiment shows that the view specific knowledge can be populated in the integration knowledge model, but doesn't show how this populated knowledge can be visualised. Following section briefs how to visualise the captured view specific knowledge.

### 7.5.3.2 Visualising View Specific Knowledge

The populated knowledge of relationships between the machining processes and the polynomial coefficients of machineable tolerances has been visualised using interface shown in figure 7.22. Similarly, the populated machining method and the related feature machining process can be visualised using different dialog. The experiment
described in this section shows that the structure of integration knowledge model is capable of capturing view specific knowledge.

Figure 7.22 Visualising machining process tolerance limitation knowledge

Therefore, experiments described in section 7.5 shows that the integration knowledge can be captured in the knowledge layer. In addition, the captured integration knowledge can be visualised and modified. Appendices D-8 and D-9 show the detail procedures of how to populate, visualise, and modify the integration knowledge in an integration knowledge model.
7.6 Knowledge Layer to Support Information Integration Between Views

7.6.1 General Description

A flexible integration between the information of multiple viewpoints can be achieved by separating integration knowledge and product information. This is a novel idea of the research and detailed in chapter 3. This section explains three different experiments conducted, as listed below, to show that the separated information and knowledge layers are support the integration of the information of multiple viewpoints.

1) Perform the transformation of information between two views
2) Perform transformation between two views and then validate the transformed information
3) Perform the transformation between two views and identifying the required integration knowledge. Modify the identified knowledge, again perform the transformation, and compare both earlier and later transformed information.

First experiment has been performed to show that transformation knowledge is necessary to perform transformation. The second experiment has been performed to show that the view specific knowledge is necessary for the validation of transformed information with view specific considerations. The third experiment has been performed to show that the integration between views depends on the knowledge in the knowledge layer.

A sprue bush - cavity insert assembly has again been used as a sample product in all the above-mentioned experiments. In these experiments, information of feeding and assembly sub functional views has been transformed to the information of assembly and machining views as illustrated in figure 7.23.
7.6.2 Information Transformation between Views

This section describes the experiment conducted to show that the necessity of transformation knowledge in the information transformation between views. It shows how the information of a feeding and assembly functional views of the sprue bush-cavity insert assembly can be transformed to the information of assembly view. This experiment has conducted in four steps:

a) Populate view information
b) Visualise populated information
c) Utilise transformation knowledge and perform information transform between views
d) Visualise populated and transformed information

Procedures to populate and visualise sub functional views have been described in section 7.4.2 and 7.4.3 as well as in appendixes D-5 and D-6. These populated sub
functional views information has been transformed to the assembly view of the cavity insert - sprue bush assembly, as described below.

Transforming information from one viewpoint to another requires knowledge of both viewpoints under consideration. This knowledge has been termed as transformation knowledge and captured in the integration knowledge layer, as detailed in chapter 5. This transformation knowledge needs to be retrieved from the knowledge layer to perform information transformation between views.

Figure 7.24 Interface used for the transformation and the diagrammatical illustration in the integration of information and knowledge layers

Once the information of sub functional views is selected for transformation, its required transformation knowledge needs to be identified and retrieved from the integration knowledge model. Figure 7.24 shows the interface used for this transformation, in addition to the diagrammatical illustration of the usage of information and knowledge layers during information transformation. This dialog visualises information of the feeding and the assembly sub functional views of a selected product. The required knowledge can be visualised in the same dialog, as shown in figure 7.25, after it has been identified and retrieved from knowledge model. The transformations from sub functional views to assembly view can be performed
with the help of the transformation knowledge. The transformed assembly view information can be visualised in figure 7.26 and populated in the product model by clicking “Populate Assembly View” button.

---

### FEEDING TO ASSEMBLY TRANSFORMATION

**All Products**
- Insert - Bush

**Related Fan Gate Features**
- **FAN GATE WITH SPRUE FOR GATING FUNCTION FOR Insert - I**
  - Sprue External Diameter: 21
  - Sprue Height: 30
- **FITS FOR CAVITY INSERT SPRUE BUSH ASSEMBLY FOR Insert - B**
  - Fit Type: TRANSITION FIT
  - Fit Function: PUSH FIT

**GET TRANSFORMATION KNOWLEDGE**
- **REQUIRED KNOWLEDGE**
  - Fit Type: TRANSITION FIT
  - Fit Function: PUSH FIT
  - Diameter Range: 18 to 30
  - Shaft Tolerance between 2 and 15 um
  - Hole Tolerance between 0 and 21 um

**Visualising required transformation knowledge retrieved from integration knowledge model**

---

**Figure 7.25** The visualisation of required transformation knowledge retrieved from the integration knowledge model

---
This experiment shows how the transformation knowledge can be used to transform information of sub functional views to the information of assembly view. However, it doesn't show how the populated and transformed information captured in a product model.

Dialog shown in figure 7.27 has been used to visualise the information of populated sub functional and transformed assembly views that captured in a product model. Hence, the visualisation of the information of involving views shows that the sub
functional views information transformed to assembly view information and captured in a product model.

**Figure 7.27 Interface to visualise products and their associated views information**

Detail procedures of the experiment described in section 7.6.2 are given in appendix D-10.1. In addition, this experiment shows that how the knowledge layer supports the transformation of information from a viewpoint to another. However, it is necessary that the transformed information need to be validated with the view specific considerations. Following section describes the experiment conducted to show the utilisation of view specific knowledge for the validation of transformed information.

### 7.6.3 Validation of Transformed Information

As explained in chapters 3 & 5, validation of transformed information with the view specific considerations is necessary. An experiment has been conducted to show that how a view specific knowledge is used for the validation of transformed information. As stated earlier, sprue bush – cavity insert has been used as a sample product and the information of sub functional views have been transformed to the information of machining view and then validated.
Information transformation from the sub functional views to machining view is similar to the experiment described in section 7.6.2. Transformed machining view information needs to be validated against machineability, where the machineability is an example of view specific knowledge. The view specific knowledge, which is used for the validation of transformed information, is captured in the knowledge layer. During the validation process, the required view specific knowledge needs to be retrieved from the knowledge layer and checked with the transformed information for acceptability.

Figure 7.28 Dialog to visualise transformed machining view information and the validation results

Cavity insert – sprue bush assembly can be considered as two components in the machining view, since cavity insert and sprue bush need to be machined separately. Sprue bush machining view has been selected for further explanation of validation process. Figure 7.28 visualise transformed information of machining view of sprue bush and cavity insert. The information of the machining view of a sprue bush has
been represented by an outer round, which is a machining feature. The tolerance range of the outer round has been validated against the capability of machining processes.

Figure 7.29 shows the example of the relationships between the diameter and the tolerance range of machining processes that can manufacture the outer round. This relationship knowledge, which is an example of view specific knowledge, has been retrieved from the integration knowledge model and used for the validation process.

**CONSIDERED MACHINING METHOD FOR OUTER DIAMETER**

Considered Machining Methods

- ROUGH TURNING
- TURNING

Example of the list of machining processes that can produce outer round

| Constant A | 1.88e-002 |
| Constant B | 4.3307    |
| Constant C | 108.14    |

**ANALYSIS**

Considered Diameter: 21. mm
Machinable Tolerance: 190.7939 um

Required value for C to make as possible machining: -242.9011

**Diameter Vs Tolerance**

\[ y = Ax^2 + Bx + C \]

Figure 7.29 Example of relationship between diameter and machineable tolerance range

This experiment shows that the knowledge layer supports the validation of transformed information during the integration between multiple viewpoints by providing view specific knowledge. Detail procedures of the experiments described in this section are given in appendix D-10.2.

Further, the experiments described in sections 7.6.2 and 7.6.3 have shown how integration knowledge is used for information transformation and validation.
However, these experiments do not show the influence of the contents of the knowledge layer for information transformation and validation activities. The next section describes the experiment conducted to explore the influence of the knowledge in the knowledge layer for the integration of multiple viewpoints.

7.6.4 Information Integration and Knowledge in the Knowledge Layer

Knowledge in the knowledge layer influences the integration activities. An experiment has been conducted to show the influence of the knowledge in the knowledge layer for the information integration between multiple viewpoints. In this experiment, a set of sub functional views information populated and transformed to the assembly view information with two different sets of transformation knowledge and both transformed information has been compared to show the influence of the knowledge in the knowledge layer.

As stated earlier, the information of sub functional views of cavity insert – sprue bush assembly has been transformed to the information of assembly view using interface shown in figure 7.30. Further, the transformation knowledge that is used for this transformation has been identified. The identified knowledge is an assembly function to assembly view transformation knowledge. This knowledge has been visualised and modified using the interfaces shown in figure 7.31. The transformation knowledge that used for this transformation is the relationships between push fit function and its assembly tolerances for diameter range 18 to 31. The identified transformation knowledge has been modified as follows:

Hole tolerance range 0 to 21 has been modified as 0 to 45 and
Shaft tolerance range 2 to 15 has been modified as 1 to 36.
After the transformation knowledge modified, transformation has been performed using the modified knowledge. Both transformed information before and after the modification of transformation knowledge, has been compared for the influence of knowledge in the transformation. These two sets of transformed information has been visualised in two separate dialogs as shown in figure 7.32.
Figure 7.31 Interfaces to visualise/modify transformation knowledge
Figure 7.32 Dialogs to visualise transformed assembly view information before and after the modification of transformation knowledge
In this experiment, the same set of assembly functional view information has been transformed with two different sets of transformation knowledge. The transformation that used the transformation knowledge before the modification has given the transformed information with hole tolerance range 0 to 21 and shaft tolerance range 2 to 15. However, the transformation using the modified transformation knowledge gives the transformed information with hole tolerance range 0 to 45 and shaft tolerance range 1 to 36. Hence, the result of this experiment clearly shows that the transformed information depends on the knowledge in the knowledge layer.

The experiments explained in section 7.6 show that the knowledge layer supports the information integration between multiple viewpoints by providing transformation and view specific knowledge. However, this experiment doesn't show how consistency of information can be maintained between multiple viewpoints. The experiments conducted to show that the proposed ontology provides support to maintain consistency among multiple viewpoints are described in the next section of this chapter.

7.7 Maintaining Consistency in the Information of Multiple Views with the Support of Information and Knowledge Layers

7.7.1 General Description

Consistency of information can be maintained in all the views with the help of the information and the knowledge layers. If the information of one view is modified, that should be propagated to all the related views in order to maintain consistency in the information of all related views. Each of the features has to contain the information of its related features, in order to identify the related features of a modified feature. The part of the class structure of the product model, which is shown in figure 7.33, helps to capture the information of the related features. The class called Dependent Feature captures related fractures information. Has relationship relates Dependent Feature class and Feature Representation classes as shown in figure 7.33. This part of the product model structure has been detailed in chapter 4.10.3.
The following sub-sections describe the experiments conducted to show how the proposed ontology, which contains information and knowledge layers, supports to maintain consistency between the information of multiple views. In these experiments, sprue bush - cavity insert assembly has been used as a sample product. Information of the feeding sub functional and assembly sub functional views has been populated and transformed to the assembly and the machining views. After the information of these four related views has been captured in the product model, two kinds of experiments has been performed as listed below:

1) A modification performed in a functional view, which is the core view, information and explored how the ontology maintains consistency in the information of all related views.

2) A modification performed in non-functional view information and explored how the ontology maintains consistency in the information all the related views.

These two kinds of experiments are described in the following sub sections.

7.7.2 Experiment to Test the Consistency Maintenance in Multiple Viewpoint Information After the Modification in a Functional View

In this experiment, the cavity insert – sprue bush sub functional view information has been modified and tested for the propagation of the modification to all its related views. Figure 7.34 shows the diagrammatical illustration of this experiment. The fit function of cavity insert – sprue bush assembly, which is the information of assembly sub functional view, has been change to location fit from push fit. The information of related views has been identified from the information of involving views. The assembly and the machining views have been identified as related views for this
change. This identification has been performed with the help of the part of the product model structure as stated above. The information of the identified related views has been modified according to the feeding functional view modification and the rest of the functional view requirements. These modifications have been performed with the help of integration knowledge model. The procedures has been followed to perform this experiment is briefly described below.

![Diagram of Product Model Structure](image)

**Figure 7.34** An illustration of consistency maintenance after the modification in a functional view

Figures 7.35 and 7.36 show the dialogs that have been used for this experiment. The top dialog shown in figure 7.35 is the main modification control dialog, which facilitates the selection of the required kind of modification. In this particular experiment, the sprue bush external diameter has been modified by clicking the “Feed Function Fan Gate with Sprue” button. The bottom dialog of figure 7.35 has been used to modify the information of the feeding functional view. Once the modification in feeding functional view has been performed, it should propagate to the all related views.
Chapter 7

Figure 7.35 Dialogs to perform a modification in a sub functional view information

The top dialog shown in figure 7.36 has been used to visualise the related features of the modified feature in addition to the verification against the view specific considerations. Once a related feature has been selected, its information of both before and after the modification has been visualised in the same dialog simultaneously. The bottom dialog of figure 7.36 shows an example of verification, which is machining feature against the machineability. The verified information of all related features has
been populated in the product model by a single click after each of the modified features has been approved by the user. Detail procedures of this experiment are given in appendix D-11.1.

![Diagram](image)

Figure 7.36 The example of the modified features verification

This experiment shows that the proposed ontology helps to maintain consistency, if functional view information modified. However, in some occasions, modification may
be necessary in non-functional view information. The experiment described in the
next sub section shows how the ontology supports the maintenance of information
consistency, if non-functional view information is modified.

**7.7.3 Experiment to Test The Consistency Maintenance in Multiple Viewpoint
Information after the Modification in Non-Functional View Information**

In this experiment, the cavity insert – sprue bush non-functional view information has
been modified and tested for the propagation of the modification to all its related
views. The diameter of sprue bush, which is the machining view of cavity insert-
sprue bush, has been change to 29.63 from 21 mm. In addition, the tolerances of sprue
bush have been changed from the range 2 to 15 μm to the range 22 to 35 μm. Figure
7.37 shows the diagrammatical illustration of this experiment. This experiment is
similar to the experiment explained in the above section and it has been performed
with the same procedures as explained the above experiment.

![Diagram](image.png)

Figure 7.37 An illustration of consistency maintenance after the modification in a
non-functional view
Figure 7.38 User Interface to perform the experiment to show the consistency maintenance in the information of multiple viewpoints

Figure 7.38 shows the user interfaces have been used to perform the experiment. This experiment shows how the ontology supports the maintenance of consistency of the
information of multiple views, if non-functional view information is modified. However, the experiments described in section 7.7 have been performed for simple modifications in features. Therefore, this approach needs to be extended to support radical changes such as adding and removing of features.

Further, experiments described in sections 7.3 to 7.7 have been performed for a single product. However, integration of multiple viewpoints of more than one product is necessary. Therefore, integration between viewpoints of multiple products has been explored and described in the following section.

7.8 Integration Between Views of Multiple Products

7.8.1 General Description

![Diagram](image.png)

Figure 7.39 Example of relationships between plastic and mould products
In the design and manufacture environment, information of a product may have influence on the information of its related products. For example, design and manufacture activities of plastic and mould products are highly interrelated to each other. Figure 7.39 shows the example of relationships between plastic product, mould core and mould cavity insert products, where the shape of the plastic product offered by the cavity and core inserts during the moulding operation. This kind of relationship has been detailed in chapter 4.7 and product-has-product relationships have been proposed for the product model structure as shown in figure 7.40 in order to capture the relationships of multiple products.

![Figure 7.40 A part of the product model structure to capture relationships of multiple products](image)

This section explains the experiments conducted to show how the ontology supports the integration between views of multiple products. A rectangular shaped plastic product and its related mould products have been taken as sample products for the experiments described in this section. Mouldability view of the plastic container and shaping, machining views of the cavity and core inserts have been considered. The mouldability view of plastic container has been represented by the mouldable features as shown in figure 7.41. The cavity and the core inserts provide respectively the external and the internal shapes of the plastic container. Therefore, the shaping functional view of the cavity and the core inserts has been considered for the experiments described in this section. The shaping functional view has been represented by the shaping functional features. To perform the shaping function, the inserts need to be manufactured to the required shape. Machining method has been selected as the example of the manufacturing method for the experiments described in this section. Hence, the machining view features have been used to represent the cavity and core inserts in the machining view.
Figure 7.41 shows the mouldability features of a simple rectangular shaped container. However, the dimensions of these mouldability features are necessary to perform the experiments to test the integration between views of multiple products. Hence, the dimensions of simple rectangular shaped container as shown in figures 7.42 & 7.43 in front and plan views respectively have been used as example for the experiments described in this section. Similarly, the shaping functional view and the machining view of the inserts have been represented by the shaping functional features and the machining features respectively.
Figures 7.42 *Front* view of rectangular shaped container

Figures 7.43 *Plan* view of rectangular shaped container
This section describes two kinds of experiments as follows:

1) Transform the information from the mouldability view of a rectangular shaped plastic container to the shaping functional view of the cavity and the core inserts. This experiment shows how the ontology supports the integration between the views of two different products.

2) Transform the information of the shaping functional view to the machining view of the cavity and the core inserts in order to show how the transformed information can be further transformed to the related views.

To perform these experiments, instances of the related multiple products with their information have been populated in the product model. The following sub section explains the procedures to populate and relate multiple products.

7.8.2 Population of Multiple Related Product Instances and their Information

7.8.2.1 Populating and Relating Multiple Product Instances

Figure 7.44 The relationships between the rectangular shaped plastic product and inserts of the mould

As stated earlier, the rectangular shaped plastic product and the mould inserts have been used as sample products to perform the experiments. Figure 7.44 shows these related products. Instances of these multiple products have been populated in the
product model using the dialog shown in figure 7.45. After all the required product instances have been populated, the mould instances have been assigned to its related plastic product instances. Figure 7.46 shows the dialog that has been used to assign the plastic product instances to its related mould instances and create relationships to each other.

![Figure 7.45 The dialog to populate product instances](image)

![Figure 7.46 Relating the plastic and the mould product instances](image)
Chapter 7

After the product instances have been populated and related to each other, the information of the plastic product has been populated in order to perform the experiments. In these experiments, the populated information of the plastic product has been transformed to the related mould product instances.

Shrinkage value of the plastic product material has been populated in the product model in order to support the experiments, where the shrinkage value influences the definitions of the mould cavities. Figure 7.47 shows the dialog that has been used to populate the shrinkage value of plastic product material.

![Figure 7.47 Dialog to populate the shrinkage characteristics of the plastic material](image)

The experiments described in this section, the information of the mouldability view of the plastic product has been transformed to the shaping functional views of the cavity insert and the core insert. Then the transformed functional view information has been further transformed to the machining view information. Hence, the population of the information of the mouldability features is necessary.

Figure 7.48 shows the example of the mouldability features of a rectangular shaped plastic product. Interface shown in figure 7.49 has been used to populate the mouldability features of the plastic product in the product model. The following sub
section briefly explains the experiments conducted to show the information integration of the views of multiple products.

![Diagram of mouldability features of rectangular shaped plastic product](image)

**Figure 7.48** An example of mouldability features of rectangular shaped plastic product

![Interface to populate/ view mouldability features](image)

**Figure 7.49** Interface to populate/ view mouldability features
7.8.3 Information Integration Between Mouldability View of Plastic Product and Functional and Machining Views of Mould Products

As stated earlier, two kinds of experiments have been performed. First kind of experiment has been performed to test how the ontology supports the transformation of information between views of two different products, where the mouldability features of a rectangular shaped plastic product have been transformed to the shaping functional features of the mould inserts. Second kind of experiment has been performed to test how the ontology supports the propagation of the information of the transformed view, where the transformed shaping functional features have been further transformed to the machining features. In both of these experiments, transformation knowledge has been used to transform information between views. The transformation knowledge that used for the transformation of information has been captured in the integration knowledge model. Figure 7.50 shows the diagrammatical illustration of the experiments. The following sub sections briefly describe the procedures followed to perform the experiments.
7.8.3.1 Information Transformation Between Views of Two Related Products

In the experiment one, the populated mouldability features of the plastic product have been transformed to the shaping functional features of the inserts. Figure 7.51 shows the interface that has been used to transform mouldability view features to shaping functional view features.

Figure 7.51 Interface to transform mouldability features to shaping functional features

7.8.3.2 Information Transformation between Functional and Other Viewpoints

In the experiment two, the shaping functional features have been transformed to the machining features of the inserts. Figure 7.52 shows the interface that has been used to transform the shaping functional features to the machining features of the inserts. In addition, figure 7.53 shows the dialog that has been used to visualise populated features of the mouldability view of the plastic product and the transformed features of the shaping functional and the machining views of the inserts. Further, the examples of transformation knowledge that have been used in the experiments described in section 7.8 are mouldable to shape, shape to geometry and geometry to machining views. This transformation knowledge is documented in Appendix C.
Figure 7.52 Interface to perform information transformation from shaping functional features to machining feature

Figure 7.53 Dialog to visualise related multiple product information
The experiments described in section 7.8 show how the ontology supports the integration of the information of multiple views of the plastic product and its related mould parts. Hence, the two-layered ontology is capable of integrating views of multiple products.

7.9 Summary

The experiments described in this chapter have shown the following:

1. The structure of the product model is capable of capturing five main groups of the product related information as phase, purpose, architecture, characteristics and views in order to manage product related information effectively.
2. The product model structure is capable of capturing the information of multiple views of products.
3. The knowledge layer is capable of capturing integration knowledge in two main groups as transformation and view specific knowledge.
4. The knowledge layer supports the information integration between multiple views of a single product.
5. The ontology supports consistency maintenance in the information of all the related views.
6. The ontology supports the information integration between views of related products.

The experiments conducted during the research reported in this thesis has been limited to the injection mould environment and to the simple rectangular shaped plastic container and cavity insert – sprue bush assembly due to the time limit of the research. However, it is believe that this can be extended to the complex shaped plastic products and to the other production environments with the same concept. The information and the knowledge structures have been defined with branches to extend the validity of the concept over the various production methods and the different shapes of the products.
8. DISCUSSION, CONCLUSIONS AND FUTURE WORK

8.1 Introduction

The research reported in this thesis has explored an ontology, which is defined with a product model and an integration knowledge model as well as knowledge links, within an integrated CAE environment, and presented how such an ontology can support the integration of multiple viewpoints of products in a design for manufacture environment. The proposed structures of the product and knowledge models have been explored and experimental systems developed to verify the results.

This chapter presents a discussion of the approach and the assumptions taken in this research, followed by the conclusions reached and recommendations for future work. Section 8.2, presents a discussion of the major issues explored in this thesis. Conclusions and suggestions for future work are presented in sections 8.3 and 8.4 respectively.

8.2 Discussion

This research has shown how an ontology can be defined to support the integration of multiple viewpoints of products for design for manufacture activities. This ontology is constructed with a product model, an integration knowledge model and knowledge links. The product model is defined to capture multiple viewpoint information of products. The integration knowledge model captures the knowledge for the integration of multiple viewpoints. Knowledge links support communication of product and knowledge models as well as retrieval of information/knowledge from models. This work provides a novel contribution to the integration of multiple viewpoints within the context of integrated information systems.

The research reported in this thesis has been performed within the overall concept of a model driven application. In a model driven application, product related information is separated from the application and captured in databases with common data structure, which is understandable and handled by all software vendors, to support a wide range of product design and manufacturing applications. Conceptually this approach solves the interoperability problems of multiple software systems. However,
software tool vendors have considered proprietary data representations - a significant source for interoperability problems - as part of their competitive advantage (Szykman et al., 01). The model driven approach eliminates barriers of interoperability, but it is often viewed by a software vendor as something that will make it easier for customers to purchase and use a competitor's product rather than those sold by their own company. The alternative approach, such as an agent-based approach, is more appropriate than a model driven approach, as agents need not know about the internal workings and the implementation of other agents (they only need to know and understand the domain) and they communicate with other agents in a high-level language that allows them to make statements about their internal state and the domain is which they interact (Labrou, 02).

The product model structure that was defined for the MIM project with three classified groups of information (Dorador, 01) has been used as a basis for the research reported in this thesis. However, the necessity of capturing early phases and functional information has been identified in this research (Wolter and Chandrasekaran 91; Hirtz et al., 02; Zhang et al., 03). Hence, the product model structure has been extended with two more classified groups of classes to capture phases and purposes information in addition to architecture, characteristics and views information. The phase group of classes has been defined to capture different phases information especially early stage product development information. The purpose group of classes has been defined to capture functional information of products. Further, the views classes have been extended to capture multiple viewpoints information and their information representation languages. Therefore, the structure defined for the product model has the ability to capture and integrate multiple viewpoint product related information from conceptual to disposal phases.

The work reported in this thesis uses features as a representation language for view specific information (Tichkiewitch, 96; Tichkiewitch, 97; Bidarra et al., 97). This approach needs the definition of features of each view to be considered in the product development activities. This is a major limitation of the approach reported in this thesis, since the integration is based on defined feature sets.

The approach taken in this research excludes the inclusion of integration knowledge in the product model. In the author's view the product model is a central repository
for the information related to the life cycle of products, and hence the inclusions of knowledge in such an information model can significantly increase its complexity and leads to the definition of different structures for different products (Canciglieri and Young, 03). Therefore, it is proposed that the integration knowledge is captured separately.

The approach taken in the research to capture the integration knowledge differs from traditional integration approaches, where knowledge is mainly captured in applications such as translation or mapping mechanisms. Capturing knowledge in applications is an inflexible and hard-coded method (Kugathasan and McMahon, 01). Therefore, an additional integration knowledge model has been proposed to capture the integration knowledge.

This work identifies the necessity for a central viewpoint to perform transformation to and from other views in order to minimise the number of translation mechanisms (Teeuw et al., 96; Urban et al., 99). The functional view has been identified as the central and fundamental view for product development activities, because, in engineering design, the final goal is the creation of an artefact, product, system or process that performs a function or functions to fulfil customer need(s). In addition, each view of the product has to satisfy and not to undermine the functional view requirements (Wolter and Chandrasekaran, 91). Therefore, the functional view has been used as a central and fundamental view for the integration of multiple views.

This work has explored the injection moulded product design and manufacture environment for the integration of multiple viewpoints. The injection-moulded product design and manufacture environment has complex relationships between multiple viewpoint considerations and hence, is suitable for the exploration of the integration of multiple viewpoints. Nevertheless, relationships between multiple viewpoints of design and manufacture environment of injection-moulded products are well structured and well defined. Therefore, structuring integration knowledge of injection moulded product design and manufacture environment is straightforward. Hence, the applicability of the proposed ontology for the integration of multiple viewpoints of an entirely new product development environment is limited and requires more exploration.
This research has identified two main kinds of integration knowledge: transformation knowledge and view specific knowledge. This is the outcome of the exploration of injection moulded product development for integration of multiple viewpoints. If another product development environment, such as the automobile industry or electronic product manufacturing, were explored, other kinds of integration knowledge may be identified. Therefore, exploration of several kinds of product development environment is necessary for the definition of the general structure of the integration knowledge model.

Maintaining consistency between multiple viewpoint information has been recognised as an issue to be resolved in the context of the integration of multiple viewpoints (Kraker et al., 99; Hofmann and Arinyo, 00; Noort et al., 02). The research explored in this thesis has contributed to tackling this issue by defining the product model structure to capture information about relationships between features. The structure of the product model has been defined in a way that each feature has a repository to capture information of its dependent features. If a feature is modified, captured information about dependent features helps to track and identify the dependent features that may be affected by the modification. Hence, this approach is supportive to maintain consistency between multiple viewpoints. However, this approach helps to maintain consistency between multiple viewpoints only for simple changes, not for radical changes such as the adding and removing of features. Therefore, this research needs to be extended to identify methodologies to maintain consistency between multiple viewpoints for radical changes.

8.3 Conclusions

I.) An ontology has been defined which can support flexible information integration between multiple viewpoints of design for the manufacture of products. The ontology is constructed with a product model, an integration knowledge model and knowledge links.

II.) The product model structure defined in the MIM project has been extended in the research reported in this thesis to capture multiple viewpoint information of products from the conceptual to disposal phases with five
classified groups of product related information in order to support the information integration between multiple viewpoints

III.) Integration knowledge has been separated from the product model in order to define the general structure of the product model for all kinds of products. The inclusion of the integration knowledge in the product model can significantly increase the complexity of the product model and leads to the definition of different structures for different products.

IV.) Separated integration knowledge has been captured in the knowledge model in order to offer greater flexibility and to minimise hard coded programs. The approach taken in the research to capture the integration knowledge differs from traditional integration approaches, where knowledge is mainly captured in applications such as translation or mapping mechanisms. Capturing knowledge in applications is an inflexible and hard-coded method.

V.) The necessity for a core view of the product has been identified in order to minimise the number of point-to-point view translations by replacing the core-to-point view translations. The functional view has been identified as the core and fundamental view for product development activities since the final goal of the creation of an artefact, product, system or process is to fulfil function(s). Therefore, the functional view has been used as a central and fundamental view for the integration of multiple views.

VI.) It has been shown that multiple viewpoints of single as well as related products can be integrated by the support of a product model structure, where the product-has-product relationship captures the information of related products.

VII.) It has been shown that consistency between information of multiple viewpoints for simple changes can be maintained by the extended product model structure with the repository to capture information of related features.
VIII.) In this research, knowledge links have been used to link the product model and integration knowledge model as well as to retrieve the required information/knowledge from models.

8.3 Recommendation for future work

I.) The work reported in this thesis has been performed based on the model driven methodology. Hence, future research is necessary for the exploration of integration methodologies other than the model driven approach for multiple viewpoints.

II.) Future research is necessary for the identification of the suitability of agent technology for the integration of multiple viewpoints without altering the present software tool vendors' proprietary data representations, as software tool vendors consider proprietary data representations as part of their competitive advantage.

III.) The methodology proposed in this research requires multiple view specific features for the representation of multiple viewpoint information. Therefore, future work is necessary to identify more effective view specific information representation methods.

IV.) The proposed methodology has been tested for a simple shaped plastic product. Therefore, there is a need to test this methodology for a more complex shaped plastic product for its applicability.

V.) The work reported in this thesis has been tested for the injection moulded plastic product development environment, where the relationships of multiple viewpoints are well structured. The reported work needs to be extended to an entirely new product development environment, where relationships between multiple viewpoints may be difficult to define.

VI.) The method proposed in the research reported in this thesis is only able to maintain consistency between multiple viewpoints for simple changes. Therefore, further work is necessary to extend this approach to support radical changes.
9. REFERENCES


Bronsvoort WF and Jansen FW, 1993, Feature Modelling and Conversion – Key Concept to Concurrent Engineering, Computers in Industry; 21(1); 61-86.


References


References


Lovett PJ, Ingram A and Bancroft CN, 1999, A Little Knowledge is a Wondrous Thing, Manufacturing Engineer; June 1999; 125-127.


Mckay A, Bloor MS and De Pennington A, 1996, A Frame Work for Product Data, IEEE Transaction of Knowledge and Data Engineering; 8(5); 825-837.


Miao Y and Haake JM, 1999, Supporting Concurrent Design in SCOPE, Concurrent Engineering: Research and Applications; 7(1); 55-65


MOSES, 1994, MOSES Seminar: Information System for Competitive Manufacture, Loughborough University, UK.


Net AJVD, Vries WAHD, Debressine FLM and Wolf ACHVD, 1996, A Relation-Based Product Model Suited for Integration Design and Manufacturing, Annals of the CIRP; 45(1); 161164.


Shaw NK, Bloor MS, De Pennington A, 1989, Product Data Models, Research in Engineering design, 1; 43-50.


Teknowledge, 2002, Ontology website http://ontology.teknowledge.com


References


References


Young RIM, Guerra D, Gunendran AG, Das B and Cochran C, 2004, Sharing manufacturing information and knowledge in design decision support, Proceedings of 5th International Conference on Integrated Design and Manufacturing in Mechanical Engineering, 5 - 7 April 2004, University of Bath, Bath, UK


Appendix A. TOOLS

A.1 Introduction

To support the exploration of the research and the implementation of the experimental system tools have been used. Rational Rose UML (Unified Modelling Language) tool has been used to support the exploration of the research idea and the development of information and knowledge structures. In the experimental implementation of this research, the ObjectStore database and Visual C++ programming environment have been employed. While the ObjectStore database has been used to realise the information and knowledge structures, the Visual C++ has been selected to realise the functionality of the experimental system. These tools are addressed in the following sections.

A.2 Unified Modelling Language (UML)

The notation provided by the UML (Unified Modelling Language) has been selected to support the representation of the information and the knowledge models. UML is a standard language to support the design and modelling of multiple perspectives of information systems. Recently, it has become recognised and accepted as a potential notation standard by the OMG (Object Management Group). Such notation defines a set of basic diagrams that provide multiple perspectives (structural/static and behavioural/dynamic) of the system (object-oriented) under analysis or development, allowing a real world representation of the system in development.

UML diagrams include Use Case Diagrams, Class Diagrams, Interaction (Sequence and Collaboration) Diagrams, Activities Diagrams, State and Transition Diagrams, Deployment Diagrams, etc. However, only the class diagram has been used in this research to explore and define the structures of information and knowledge models. The elements of the class diagram are briefly explained in the following sub sections.

A.2.1 Object

An object is a representation of an entity, either real world or conceptual. An object is a concept, abstraction, or thing with well-defined boundaries and meaning for an
application. Each object in a system has three characteristics: state, behaviour, and identity.

A.2.2 State, Behaviour, and Identity

The state of an object is one of the possible conditions in which it may exist. The state of an object typically changes over time, and is defined by a set of properties (called attributes), with the value of the properties, plus the relationships the object may have with other objects.

Behaviour determines how an object responds to request from other objects and typifies everything the object can do. Behaviour is implemented by the set of operations for the object.

Identity means that each object is unique—even if its state is identical to that of another object.

A.2.3 Class

A class is a description of a group of objects with common properties (attributes), common behaviour (operations), common relationships to other objects, and common semantics. Thus, a class is a template to create objects. Each object is an instance of some class and objects cannot be instances of more than one class.

A good class captures one and only one abstraction—it should have one major theme.

Classes should be named using the vocabulary of the domain. The name should be singular noun that best characterises the abstraction. Acronyms may be used if the acronym has the same meaning for all involved, but if an acronym has different meanings for different people then the full name should always be used. If a class is named with an acronym, the full name should also be contained the class documentation.

In the UML, classes are represented as compartmentalised rectangles. The top compartment contains the name of the class, the middle compartment contains the
structure of the class (attributes), and bottom compartment contains the behaviour of the class (operations). A class is shown in figure A.1.

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute: type = initval</td>
</tr>
<tr>
<td>operation(argname): return</td>
</tr>
</tbody>
</table>

Figure A.1 UML notation for a class

A.2.3.1 Representing Behaviour and Structure in a Class

A class embodies a set of responsibility that define the behaviour of the objects in the class. The responsibilities are carried out by operations defined for the class. An operation should only one thing and it should do it well.

Structure of an object is desired by the attributes of the class. Each attribute is a data definition held by objects of the class. Objects defined for the class have a value for every attribute of the class.

It is possible to define all attributes and operations for a class. It captures the behaviour and structure of the class as shown above. It is advisable to document all the class attributes and operations.

A.2.3.2 Displaying Attributes and Operations of Classes

Attributes and operations may be displayed on a class diagram. Often, a class diagram is created specifically for this purpose—it shows the structure and behaviour of the classes. Figure A.2 shows the attributes and operations of classes.

Different kinds of classes can be used in the class diagram to represent the objects in the real world. Association and inheritance classes have been used in this research to define the structures of the information and knowledge layers. The following sections briefly describe these kinds of classes.
A.2.4 Association Classes

A relationship may also have structure and behaviour. This is true when the information deals with a link between two objects and not with one object itself. The structure and behaviour belonging to a relationship is held in an association class.

A.2.5 Inheritance Classes

Inheritance defines a relationship among classes where one class shares the structure and behaviour of one or more classes. A hierarchy of abstractions is created in which a subclass inherits from one or more supper classes. Inheritance is also called an "is-a" or "kind-of" hierarchy. A subclass inherits all attributes, operations, and relationships defined in any of its super classes. Thus, attributes and the relationships are defined at the highest level in the hierarchy at which they are applicable, which allows all lower classes in the hierarchy to inherit them. Subclasses may be augmented with additional attributes and operations that apply only to that level of hierarchy. A subclass may its own
implementation of an operation. Since an inheritance relationship is never named, role names are not used, and multiplicity does not apply.

There is no limit to the number of classes allowed in an inheritance hierarchy. However, practical experience has shown that typical C++ class hierarchy contains three and five layers.

Inheritance is the key to reuse. A class can be created for one application and then subclass may create to add more information needed for a different application.

A.2.6 The Relationships between Classes

All systems are made up of many classes and objects. System behaviour is achieved through the collaborations of the objects in the system. This is often referred to as an object sending a message to another object. Relationships provide the conduit for object interaction. Two types of relationships discovered during analysis are associations and aggregations.

A.2.6.1 Association relationships

An association is a bi-directional semantic connection between classes. It is not a data flow as defined in structured analysis and design-data may flow in the either direction across the association. An association between classes means that there is a link between objects in the associated classes.

![Diagram showing association relationship between classes]

Figure A.3 Association relationship between classes

The number of objects connected depends upon the multiplicity of the association. In the UML, association relationships are shown as a line connecting the associated classes, as shown in figure A.3.
A.2.6.2 Aggregation relationships

An aggregation relationship is a specialised form of association in which a whole is related to its part(s). Aggregation is known as a "part-of" or containment relationship. The UML notation is shown in figure A.4.

![Aggregation relationship between two classes](image)

Figure A.4 Aggregation relationship between two classes

The following tests may be used to determine if an association should be an aggregation:

- Is the phrase "part of" used to describe the relationship?
- Are some operations on the whole automatically applied to its parts?
- Is there an intrinsic asymmetry to the relationship where one class is subordinate to the other?

A.2.7 Naming the Relationships

An association relationship may be named. Usually the name is an active verb or verb phrase that communicates meaning of the relationship. Since the verb phrase typically implies a reading direction, it is desirable to name the association so it reads correctly from left to right or top to bottom. Figure A.5 shows an example of the name an association relationship.

![Naming the association relationship](image)

Figure A.5 Naming the association relationship

It is important to note that the name of the association is optional. Names are added if they are needed to add clarity to the model. Aggregation relationships typically are not named since they are read using the words "has" or "contains".
A.2.8 Role Names

The end of an association where it connects to a class is called an association role. Role names can be used instead of association names. A role name is a noun that denotes the purpose or capacity where in one-class associates with another. The role name is placed on the association near the class that modifies, and may be placed one or both ends of an association line. Figure A.6 shows an example of role name indicated in an association relationship. It is not necessary to have both role name and association name.

```
MouldDesign +apply MouldabilityConstraints
```

Figure A.6 Indicating role name

A.2.9 Multiplicity Indicators

Although multiplicity is specified for classes, it defines the number of objects that participate in a relationship. Multiplicity defines the number of objects that are linked to one another. Figure A.7 shows an example of the multiplicity indication of a relationship between two classes. There are two multiplicity indicators for each association or aggregation—one at each end of the line. Some common multiplicity indicators are:

- **1** Exactly one
- **0..*** Zero or more
- **1..*** One or more
- **0..1** Zero or one
- **5..8** Specific range (5,6,7, or 8)
- **4..7,9** Combination (4,5,6,7, or 9)

```
MouldDesign +apply MouldabilityConstraints
```

0..1

0..*

Figure A.7 An example of multiplicity indication
A.2.10 Inheritance Trees

The basis for specialisation (i.e., why subclasses were created) in an inheritance relationship is called the discriminator. The discriminator typically has a finite set of values and subclasses that may be created for each value. Figure A.8 shows an example of inheritance relationships between parent and child classes.

Figure A.8 Inheritance relationships between parent and child classes

A.2.11 Single Inheritance Versus Multiple Inheritances

Single inheritance is simple and easy to use. If a class has one set of parents—that is, there is one chain of super classes. But multiple inheritances involve more than one chain of super classes. This multiple inheritances have numerous problems such as name clashes and multiple copies of inherited features. Multiple inheritance also leads to less maintainable code. Use multiple inheritances only when it is needed and always use it with care.

A.2.12 Inheritance Versus Aggregation Relationships

Inheritance should be used to separate commonality from specifics. Aggregation should be used to show composite relationships. Often the two types of relationships can be use together.
A.3 ObjectStore Database

ObjectStore is a pure object-oriented database tool and has been used to implement the model schemas of the Product Model and the Integration Knowledge Model. Besides the ObjectStore database application itself, two other tools, named ObjectStore Database Designer and ObjectStore Inspector, have been used to support the computational implementation process.

The ObjectStore Database Designer uses compatible UML class diagrams as input to generate the actual C++ code for each class in the database schema. The classes generated define the information structure (persistent data) of the information/knowledge models mentioned above, i.e. the product model and the integration knowledge model.

The ObjectStore Inspector is a tool that allows visualisation of both, object's data and relationships, in the database file after it has been populated.

A.4 Visual C++

The functionality of the software application has been realised by using Microsoft Visual C++, as a computational programming environment. This tool allows the development and implementation of visual interfaces between the user and the information/knowledge models stored in the databases. The tool uses an object-oriented approach and is based on Windows MFC (Microsoft Foundation Classes).
Appendix B. STRUCTURES OF THE INFORMATION AND THE KNOWLEDGE MODELS

B.1 Introduction

The structures of the information and the knowledge models have been defined to support the flexible information integration of multiple views. This appendix gives the detail structures of the information and the knowledge models.

B.2 Structure of the Information Model

Product model has been defined as an information layer to capture product related information. The following sections detail the product model structure.

B.2.1 Top Level Classes of Product Model

Figure B.1 Top level classes of product model
Figure B.1 shows top-level classes of product model, which includes five product related information and their relationships. In addition, classes those help to capture views representation information shown in figure B.1.

**B.2.2 Characteristics Branch Classes**

The characteristics branch has been defend to capture the product characteristics. The plastic material characteristics have been used in the research. Figure B.2 shows the classes of characteristics branch.

![Characteristics Branch of Classes](image)

Figure B.2 Characteristics Branch of Classes
B.2.3 Views Branch of Classes

The class structure has been defined to capture the information of multiple views. Figure B.3 shows the involving classes of views branch.

![Diagram of Views Branch of Classes]

Figure B.3 Views branch of classes

B.2.4 Class Structure to Capture Feature Representation

![Diagram of Class Structure to Capture Feature Representation]

Figure B.4 Classes to capture multiple views representations
B.2.4.1 Class Structure to Capture Functional Features

The class structure has been defined to capture mould functional features information.

Figure B.5 Class structure to capture mould functional feature
B.2.4.2 Class Structure to Capture Manufacturing Features

The class structure has been defined to capture manufacturing features, which includes machining, casting, forging, and moulding manufacturing methods as shown in figure B.6. However, in this research mouldable and machining features have been used to explore the flexibility integration of the information of multiple views. Figures B.7 and B.8 show the class structures to capture mouldability and machining features respectively.

![Class Structure Diagram]

Figure B.6 Top level class structure to capture manufacturing features
Figure B.7 Class structure to capture mouldability features
Figure B.8 Class structure to capture machining features
B.3 Structure of the Knowledge Model

Integration knowledge model has been defined as a knowledge layer to capture integration knowledge to support the flexible information integration of multiple viewpoints. The following sections detail the structure of the integration knowledge model.

B.3.1 Top Level Classes of Integration Knowledge Model

Figure B.9 shows the top-level classes of integration knowledge model, which includes transformation and view specific knowledge classes. The following sub sections detail the structures of the transformation and the view specific knowledge branches.

B.3.2 Class Structure to Capture Transformation Knowledge

Figure B.10 shows the top-level classes of transformation knowledge branch of integration knowledge model. Function to process transformation knowledge has been
used to explore the research idea. Therefore, the following sub sections detail the class structure of the function to process transformation knowledge. The function to process transformation knowledge contains two main branches as auxiliary process and main process transformation knowledge.

B.3.2.1 Class Structure to Capture Function to Auxiliary Process

Figure B.11 shows the structure of classes to capture function to auxiliary process transformation knowledge.

```
        Auxiliary
               |
               v
  OtherAuxiliaryFunction  FeedingFunction  VentingFunction  CoolingFunction
               |
               v
  OtherFeeding  FeedingForInjectionMoulding
               |
               v
  FeedingViaSprueBush  OtherFeedingMethods
               |
               v
  CavityInsertAndSprueBush  OtherComponents
               |
               v
  FeedFunctionToFeedAssembly  FeedFunctionToFeedManufacture

Figure B.11 Class structure to capture function to auxiliary process transformation knowledge
```
B.3.2.2 Class Structure to Capture Function to Main Process

Figure B.12 shows the structure of classes to capture function to main process transformation knowledge.

Figure B.12 Class structure to capture function to main process transformation knowledge
B.3.3 Class Structure to Capture View Specific Knowledge

Figure B.13 shows the class structure of view specific knowledge branch of integration knowledge model. Manufacturing view specific knowledge has been used to explore the research idea.

![Diagram of class structure](image)

**Figure B.13** Class structure to capture view specific knowledge
Appendix C. CLASS STRUCTURES FOR FEATURE SETS AND THE EXAMPLES OF POPULATED INTEGRATION KNOWLEDGE

C.1 Introduction

The research reported in this thesis has been used the sets of features of mouldability, shaping functional, machining, assembly views to explore the flexible information integration of multiple views. Definitions of these feature sets have been obtained from STEP standards and previous researches. Section C.2 details the class structures of the feature sets, which have been defend in this research. In addition, section C.3 documents examples of integration knowledge that have used in the experimental system.

C.2 Class Structures of Feature Sets

Product model has been defined to capture multiple views information of products. Information of multiple views has been represented by feature sets. Following sub sections explain the class structures of mouldability, shaping functional, machining and assembly feature sets.

C.2.1 Class Structure for Mouldability Features

The mouldability features have been defined by Canciglieri OJ (Canciglieri OJ, 99) for his research. The class structure of mouldability features has been defined based on his mouldability feature definitions. Figure C.1 shows the class structure of mouldability features with their attributes.
Figure C.1 The class structure to represent mouldability features
C.2.2 Class Structure for Shaping Functional Features

The shaping functional features have been defined based on the definitions of mouldability features, which have been defined by Canciglieri OJ (Canciglieri OJ, 99). The class structure of shaping functional features has been derived based on the definitions of the shaping functional features. Figure C.2 shows the class structure of shaping functional features with their attributes.
C.2.3 Class Structure for Machining Features

The definitions of machining features have been obtained from AP 224 of STEP standard (ISO 10303-224, 00). The class structure of machining features has been defined based on the definitions of the machining features. Figure C.3 shows the top-level classes of machining features. Further, figures C.4 and C.5 shows the detail of the structure of machining features with their attributes.

```
Machining Feature

Knurl  Multi axis feature  Outer round  Revolved feature  Thread  Marking  Spherical cap  Compound feature
```

Figure C.3 Top-level classes of machining features
Figure C.4 Class structure to represent multi axis machining features
Figure C.5 Detail class structure to represent machining features
C.2.4 Class Structure for Assembly Features

The class structure has been defined to represent the information of assembly view features. This class structure has been defined based on the assembly view requirements. Figure C.6 shows the class structure defined to represent the assembly view features.

Figure C.5 Detail class structure to represent assembly features
C.3 Examples of Populated Integration Knowledge

Examples of integration knowledge have been populated in the integration knowledge model to perform the experiments. The following sub sections give the detail of populated transformation and view specific knowledge.

C.3.1 Examples of Transformation Knowledge

<table>
<thead>
<tr>
<th>Diameter</th>
<th>Clearance Fits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hole (H11)</td>
</tr>
<tr>
<td></td>
<td>Min</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>50</td>
<td>65</td>
</tr>
<tr>
<td>65</td>
<td>80</td>
</tr>
<tr>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
</tr>
<tr>
<td>120</td>
<td>140</td>
</tr>
<tr>
<td>140</td>
<td>160</td>
</tr>
<tr>
<td>160</td>
<td>180</td>
</tr>
<tr>
<td>180</td>
<td>200</td>
</tr>
<tr>
<td>200</td>
<td>225</td>
</tr>
<tr>
<td>225</td>
<td>250</td>
</tr>
<tr>
<td>250</td>
<td>280</td>
</tr>
<tr>
<td>280</td>
<td>315</td>
</tr>
<tr>
<td>315</td>
<td>355</td>
</tr>
<tr>
<td>355</td>
<td>400</td>
</tr>
<tr>
<td>400</td>
<td>450</td>
</tr>
<tr>
<td>450</td>
<td>500</td>
</tr>
</tbody>
</table>

Figure C.6 The example of loose running functional fit to the assembly feature transformation knowledge

Assembly function to assembly transformation knowledge has been used to test the transformation between assembly functional view information to assembly view information. Figures C.6, C.7 and C.8 gives the examples of the assembly function to the assembly view transformation knowledge.
<table>
<thead>
<tr>
<th>Diameter</th>
<th>Clearance Fits</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hole (H9)</td>
<td></td>
<td></td>
<td>Hole (H9)</td>
<td></td>
<td></td>
<td>Hole (H8)</td>
<td></td>
<td></td>
<td>Hole (H7)</td>
<td></td>
<td></td>
<td>Hole (H7)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shaft (d10)</td>
<td></td>
<td></td>
<td>Shaft (e9)</td>
<td></td>
<td></td>
<td>Shaft (f7)</td>
<td></td>
<td></td>
<td>Shaft (g6)</td>
<td></td>
<td></td>
<td>Shaft (h6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loose Running</td>
<td>Easy Running</td>
<td>Normal Running</td>
<td>Sliding &amp; Location</td>
<td>Location</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0.025</td>
<td>-0.060</td>
<td>-0.020</td>
<td>0</td>
<td>0.025</td>
<td>-0.039</td>
<td>-0.014</td>
<td>0</td>
<td>0.014</td>
<td>-0.016</td>
<td>-0.006</td>
<td>0</td>
<td>0.010</td>
<td>-0.008</td>
<td>-0.002</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0.030</td>
<td>-0.078</td>
<td>-0.030</td>
<td>0</td>
<td>0.030</td>
<td>-0.050</td>
<td>-0.020</td>
<td>0</td>
<td>0.018</td>
<td>-0.022</td>
<td>-0.010</td>
<td>0</td>
<td>0.012</td>
<td>-0.012</td>
<td>-0.004</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0.036</td>
<td>-0.098</td>
<td>-0.040</td>
<td>0</td>
<td>0.036</td>
<td>-0.061</td>
<td>-0.025</td>
<td>0</td>
<td>0.022</td>
<td>-0.028</td>
<td>-0.013</td>
<td>0</td>
<td>0.015</td>
<td>-0.014</td>
<td>-0.005</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>0</td>
<td>0.043</td>
<td>-0.120</td>
<td>-0.050</td>
<td>0</td>
<td>0.043</td>
<td>-0.075</td>
<td>-0.032</td>
<td>0</td>
<td>0.027</td>
<td>-0.034</td>
<td>-0.016</td>
<td>0</td>
<td>0.018</td>
<td>-0.017</td>
<td>-0.006</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>0</td>
<td>0.052</td>
<td>-0.149</td>
<td>-0.065</td>
<td>0</td>
<td>0.052</td>
<td>-0.092</td>
<td>-0.040</td>
<td>0</td>
<td>0.033</td>
<td>-0.041</td>
<td>-0.020</td>
<td>0</td>
<td>0.021</td>
<td>-0.020</td>
<td>-0.007</td>
</tr>
<tr>
<td>30</td>
<td>50</td>
<td>0</td>
<td>0.062</td>
<td>-0.180</td>
<td>-0.080</td>
<td>0</td>
<td>0.062</td>
<td>-0.112</td>
<td>-0.050</td>
<td>0</td>
<td>0.039</td>
<td>-0.050</td>
<td>-0.025</td>
<td>0</td>
<td>0.025</td>
<td>-0.025</td>
<td>-0.009</td>
</tr>
<tr>
<td>50</td>
<td>80</td>
<td>0</td>
<td>0.074</td>
<td>-0.220</td>
<td>-0.100</td>
<td>0</td>
<td>0.074</td>
<td>-0.134</td>
<td>-0.060</td>
<td>0</td>
<td>0.046</td>
<td>-0.060</td>
<td>-0.030</td>
<td>0</td>
<td>0.030</td>
<td>-0.029</td>
<td>-0.010</td>
</tr>
<tr>
<td>80</td>
<td>120</td>
<td>0</td>
<td>0.087</td>
<td>-0.260</td>
<td>-0.120</td>
<td>0</td>
<td>0.087</td>
<td>-0.159</td>
<td>-0.072</td>
<td>0</td>
<td>0.054</td>
<td>-0.071</td>
<td>-0.036</td>
<td>0</td>
<td>0.035</td>
<td>-0.034</td>
<td>-0.012</td>
</tr>
<tr>
<td>120</td>
<td>180</td>
<td>0</td>
<td>0.100</td>
<td>-0.305</td>
<td>-0.145</td>
<td>0</td>
<td>0.100</td>
<td>-0.185</td>
<td>-0.084</td>
<td>0</td>
<td>0.063</td>
<td>-0.083</td>
<td>-0.043</td>
<td>0</td>
<td>0.040</td>
<td>-0.039</td>
<td>-0.014</td>
</tr>
<tr>
<td>180</td>
<td>250</td>
<td>0</td>
<td>0.115</td>
<td>-0.355</td>
<td>-0.170</td>
<td>0</td>
<td>0.115</td>
<td>-0.215</td>
<td>-0.100</td>
<td>0</td>
<td>0.072</td>
<td>-0.096</td>
<td>-0.050</td>
<td>0</td>
<td>0.046</td>
<td>-0.044</td>
<td>-0.015</td>
</tr>
<tr>
<td>250</td>
<td>315</td>
<td>0</td>
<td>0.130</td>
<td>-0.400</td>
<td>-0.190</td>
<td>0</td>
<td>0.130</td>
<td>-0.240</td>
<td>-0.110</td>
<td>0</td>
<td>0.081</td>
<td>-0.108</td>
<td>-0.056</td>
<td>0</td>
<td>0.052</td>
<td>-0.049</td>
<td>-0.017</td>
</tr>
<tr>
<td>315</td>
<td>400</td>
<td>0</td>
<td>0.140</td>
<td>-0.440</td>
<td>-0.210</td>
<td>0</td>
<td>0.140</td>
<td>-0.265</td>
<td>-0.125</td>
<td>0</td>
<td>0.089</td>
<td>-0.119</td>
<td>-0.062</td>
<td>0</td>
<td>0.057</td>
<td>-0.054</td>
<td>-0.018</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>0</td>
<td>0.155</td>
<td>-0.480</td>
<td>-0.230</td>
<td>0</td>
<td>0.155</td>
<td>-0.290</td>
<td>-0.135</td>
<td>0</td>
<td>0.097</td>
<td>-0.131</td>
<td>-0.068</td>
<td>0</td>
<td>0.063</td>
<td>-0.060</td>
<td>-0.020</td>
</tr>
</tbody>
</table>

Figure C.7 The example of clearance fit function to the assembly feature transformation knowledge
### Diameter Transition Fits

<table>
<thead>
<tr>
<th>Diameter</th>
<th>Transition Fits</th>
<th>Interference Fit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Push Fit</td>
<td>Tight Assembly Fit</td>
</tr>
<tr>
<td></td>
<td>Hole (H7)</td>
<td>Shaft (k6)</td>
</tr>
<tr>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.010</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.012</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0.015</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>0.018</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>0.021</td>
</tr>
<tr>
<td>30</td>
<td>50</td>
<td>0.025</td>
</tr>
<tr>
<td>50</td>
<td>80</td>
<td>0.030</td>
</tr>
<tr>
<td>80</td>
<td>120</td>
<td>0.035</td>
</tr>
<tr>
<td>120</td>
<td>180</td>
<td>0.040</td>
</tr>
<tr>
<td>180</td>
<td>250</td>
<td>0.046</td>
</tr>
<tr>
<td>250</td>
<td>315</td>
<td>0.052</td>
</tr>
<tr>
<td>315</td>
<td>400</td>
<td>0.057</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>0.063</td>
</tr>
</tbody>
</table>

**Figure C.8** The example of transition and interference fit function to the assembly feature transformation knowledge

In addition to the assembly function to assembly view transformation knowledge,

1) mouldability features to shape functional features,

2) shape functional features to geometry features,

3) geometry features to machining features

transformation knowledge have been used to test the information integration of the views of multiple products. The examples of this transformation knowledge have been given in figures C.9, C.10 and C.11.
<table>
<thead>
<tr>
<th>No. of PP</th>
<th>Product Type</th>
<th>Mould Type</th>
<th>Wall Type</th>
<th>Position WRT PP</th>
<th>Inclination WRT PP</th>
<th>Required Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY PRODUCT</td>
<td>Primary</td>
<td>Below-Touchining</td>
<td>Perpendicular</td>
<td>OutSide</td>
</tr>
<tr>
<td>Option 2</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY PRODUCT</td>
<td>Transition</td>
<td>Below-Touchining</td>
<td>Perpendicular</td>
<td>OutSide</td>
</tr>
<tr>
<td>Option 3</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY PRODUCT</td>
<td>Transition</td>
<td>Below-NonTouchining</td>
<td>Not Applicable</td>
<td>OutSide</td>
</tr>
<tr>
<td>Option 4</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CAVITY PRODUCT</td>
<td>Primary</td>
<td>Below-NonTouchining</td>
<td>Parallel</td>
<td>OutSide</td>
</tr>
<tr>
<td>Option 5</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE PRODUCT</td>
<td>Primary</td>
<td>Below-Touchining</td>
<td>Perpendicular</td>
<td>InSide</td>
</tr>
<tr>
<td>Option 6</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE PRODUCT</td>
<td>Transition</td>
<td>Below-Touchining</td>
<td>Perpendicular</td>
<td>InSide</td>
</tr>
<tr>
<td>Option 7</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE PRODUCT</td>
<td>Transition</td>
<td>Below-NonTouchining</td>
<td>Not Applicable</td>
<td>InSide</td>
</tr>
<tr>
<td>Option 8</td>
<td>1 PLASTIC PRODUCT</td>
<td>MOULD CORE PRODUCT</td>
<td>Primary</td>
<td>Below-NonTouchining</td>
<td>Parallel</td>
<td>InSide</td>
</tr>
</tbody>
</table>

Figure C.9 Example of mouldability to shape functional features transformation knowledge

<table>
<thead>
<tr>
<th>Mould Part</th>
<th>Per Primary</th>
<th>Per Trans</th>
<th>Bottom Primary</th>
<th>Bottom Trans</th>
<th>Bottom Corner</th>
<th>Geometry Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>RECTANGULAR CLOSED CAVITY</td>
</tr>
<tr>
<td>Option 2</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>RECTANGULAR OPEN CAVITY</td>
</tr>
<tr>
<td>Option 3</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>RECTANGULAR SOLID</td>
</tr>
</tbody>
</table>

Figure C.10 Example of shape functional to geometry features transformation knowledge

<table>
<thead>
<tr>
<th>Geometry Shape</th>
<th>Feature Type</th>
<th>Machining Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1 CYLINDER</td>
<td>POSITIVE</td>
<td>OUTER ROUND</td>
</tr>
<tr>
<td>Option 2 HOLE</td>
<td>NEGATIVE</td>
<td>ROUND HOLE</td>
</tr>
<tr>
<td>Option 3 RECTANGULAR CLOSED CAVITY</td>
<td>NEGATIVE</td>
<td>RECTANGULAR POCKET</td>
</tr>
<tr>
<td>Option 4 RECTANGULAR OPEN CAVITY</td>
<td>NEGATIVE</td>
<td>RECTANGULAR POCKET</td>
</tr>
<tr>
<td>Option 5 RECTANGULAR SOLID</td>
<td>POSITIVE</td>
<td>RECTANGULAR PROTRUSION</td>
</tr>
</tbody>
</table>

Figure C.11 Example of geometry to machining features transformation knowledge
C.3.2 Examples of View Specific Knowledge

As explained in chapter 5, graph format of machining limitation knowledge, which is an example for view specific knowledge, has been modified to Ax²+Bx+C format and captured in the integration knowledge model. Populated coefficients of A, B and C for different machining processes is given in the figure C.12.

<table>
<thead>
<tr>
<th>Machining Process</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>FINE BORING</td>
<td>0.0009</td>
</tr>
<tr>
<td>BORING</td>
<td>0.0103</td>
</tr>
<tr>
<td>FINE TURNING</td>
<td>0.0027</td>
</tr>
<tr>
<td>GRINDING</td>
<td>0.0019</td>
</tr>
<tr>
<td>DRILLING</td>
<td>0.0188</td>
</tr>
<tr>
<td>ROUGH TURNING</td>
<td>0.0188</td>
</tr>
<tr>
<td>ROUGH BORING</td>
<td>0.0188</td>
</tr>
<tr>
<td>MILLING</td>
<td>0.0136</td>
</tr>
<tr>
<td>SLOTTING</td>
<td>0.0136</td>
</tr>
<tr>
<td>PLANNING</td>
<td>0.0136</td>
</tr>
<tr>
<td>TURNING</td>
<td>0.006</td>
</tr>
<tr>
<td>CENTER LATHE BORING</td>
<td>0.006</td>
</tr>
<tr>
<td>REAMING</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Figure C.12 Example of machining limitation knowledge
Appendix D. EXPERIMENTAL SYSTEM INTERFACES AND PROCEDURES OF EXPERIMENTS

D.1 Creating New Product Model

New information model can be created from the database drop down menu of the mainframe. The procedure of creating new product model is shown in figure D.1. In addition, functionalities of the involving dialogs are highlighted in the same figure. Similarly, existing product models can be opened by following the same procedures. Following section is demonstrating how product instances can be populated to the opened/ created product model.

D.2 Populating Product Instances

New product instances can be populated to the product model from InformationLayer dropdown menu of mainframe. The procedure of populating new product instances is
shown in figure D.2, in addition to the functionalities of involving dialogs. Product population dialog has the facility to visualise populated products and the ability to create involving instances of views. This ability has been added to the product population dialog to avoid experimental system implementation errors. Required viewpoint consideration instances can be populated and relate to the product instance by selecting check boxes of product population dialog.

![Product Population Dialog](image)

Figure D.2 Procedures to populated product instances and its related view instances

### D.3 Populating Product Related Information Instances

In this research, product related information has been classified in five main groups as phase, purpose, architecture, characteristics, and views. Related views information instances can be populated with the creation of product instances as explained in the above section. Other four groups of product related information instances could be populated to the product model by *ProductProperties* of *InformationLayer* dropdown menu. Clicking *ProductProperties* menu opens dialog named *ProductInformationMainDialog*. This dialog has the facility to populate classified product related information instances by different dialogs as shown in figure D.3.
D.4 Visualising Populated Product Related Information

Populated product related information can be visualised from the product model by ObjectStore Inspector. Figure D.4a shows ObjectStore Inspector window, which allow visualising detail of populated instances of database. In addition, it has the facility to show the relationships of populated instances in different window. Visualising procedure of the relationships of populated instances is shown in figure D.4a. Figure D.4b shows examples of the relationships between product and its classified groups of related instances.
Figure D.4a ObjectStore Inspector window shows information of populated instances and the procedure to visualise their relationships.

Figure D.4b ObjectStore Inspector window shows relationships of populated instances.
D.5 Population of the Instances of Multiple Views Information Representation

Multiple views information can be represented by multiple sets of features. For the experimental system implementation, sub functional views such as feeding and assembly has been taken as example and populated to the product model. Procedure to populate sub functional views information is given in figures D.5a, D.5b and D.5c. By clicking Functional menu of InformationLayer dropdown menu opening dialog named Functional View Main Dialog as shown in figure D.5a. By clicking button named Fan Gate with Sprue of Functional View Main Dialog popup dialog shown in figure D.5b, which allows the population of feeding functional view feature. Similarly, assembly sub functional view feature can be populated to with the dialog shown in figure D.5c.

Figure D.5a Procedure to populate sub functional views information
Related Gating Features

CREATE NEW FAN GATE

Sprue External Diameter 40
Sprue Height 45

CREATE FAN GATE

Figure D.5b Dialog to populate feeding function

Related Functional Feature

Fit Type
CLEARANCE
TRANSITION
INTERFERENCE

Fit Function
Slack Running
Loose Running
Easy Running
Normal Running
Sliding Location
Location

CREATE NEW FUNCTIONAL FEATURE

Figure D.5c Dialog to populate assembly function
D.6 Visualising Multiple Viewpoint Information

Multiple viewpoint information of products can be captured in the product model. Either this information can be populated as mentioned in the above section or transform from populated viewpoints. Procedure of transforming information from populated views is detailed in appendix D.10.1. Information of populated sub functional views can be visualised with the same dialogs use to populate. However, dialog shown in figure D.6 can visualise information of involving multiple viewpoints irrespective of the method followed to populate.

![Figure D.6 Dialog to visualise multiple viewpoint information](image)

D.7 Creating New Integration Knowledge Model

New integration knowledge model can be created from the KnowledgeModel of Database drop down menu. The procedure of creating new integration knowledge model is shown in figure D.7. Similarly, existing integration knowledge models can be opened by following the same procedures of creating new knowledge model. 

227
Following section is demonstrating how knowledge instances can be populated to the opened/ crated integration knowledge model.

Figure D.7, Creating new integration knowledge model

D.8 Populating Integration Knowledge

Knowledge population main dialog can be opened by clicking KnowledgeDlg of KnowledgeLayer drop down menu as shown in figure D.8a. KnowledgePopulation main dialog has been designed to populate transformation and view specific knowledge.

Tabulated relationships between fit function and hole-shaft tolerances, which are extracted from British Standard Selected ISO Fits- Hole basis (BS 4500, 1969), has been used as an example of assembly functional view to assembly view transformation knowledge for the experimental system. Figure D.8b shows some examples of relationships between fit function and hole-shaft tolerances. These transformation knowledge can be populated by clicking MANUAL button of hole and shaft fixing assembly transformation knowledge as shown in figure D.8c.
Figure D.8a Procedures to open knowledge population main dialog

<table>
<thead>
<tr>
<th>Diameter</th>
<th>Clearance Fits</th>
<th>Loose Running</th>
<th></th>
<th></th>
<th></th>
<th>Hole (H9)</th>
<th>Shaft (d10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.025</td>
<td>-0.060</td>
<td>-0.020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.030</td>
<td>-0.078</td>
<td>-0.030</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0.036</td>
<td>-0.098</td>
<td>-0.040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>0.043</td>
<td>-0.120</td>
<td>-0.050</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>0.052</td>
<td>-0.149</td>
<td>-0.065</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>50</td>
<td>0.062</td>
<td>-0.180</td>
<td>-0.080</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>80</td>
<td>0.074</td>
<td>-0.220</td>
<td>-0.100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>120</td>
<td>0.087</td>
<td>-0.260</td>
<td>-0.120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>180</td>
<td>0.100</td>
<td>-0.305</td>
<td>-0.145</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>250</td>
<td>0.115</td>
<td>-0.355</td>
<td>-0.170</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>315</td>
<td>0.130</td>
<td>-0.400</td>
<td>-0.190</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>400</td>
<td>0.140</td>
<td>-0.440</td>
<td>-0.210</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>0.155</td>
<td>-0.480</td>
<td>-0.230</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure D.8b Example of transformation knowledge
Figure D.8c Example of manual population of transformation knowledge

However, manual population of integration knowledge is a difficult task since it considers huge number of options. Therefore, auto population method has been used for the population of knowledge instances. Auto population can be performed clicking AUTO buttons of KnowledgePopulation main dialog. Auto population method assists the user to populate integration knowledge by single clicks.

**D.9 Visualisation of the Integration Knowledge**

Populated integration knowledge can be visualised by clicking Knowledge of Viewing dropdown menu. Figure D.9a shows the procedure to visualise hole and shaft assembly function view to assembly view transformation knowledge. In addition to the visualisation of the transformation knowledge, it can be modified as shown in figure D.9b. This modification secured by password in order to prevent unauthorised
modification. Further, the password protection highlights the significance of the contents of the knowledge in the knowledge model in the integration.

Other kind of integration knowledge can be populated/modified in the integration knowledge by following similar procedures.

Figure D.9a Procedure to view assembly function to assembly view transformation knowledge
**Appendix D**

### Detail of Clearance Fit of Slack Running Fit

<table>
<thead>
<tr>
<th>Diameter Range</th>
<th>Hole Upper Tolerance</th>
<th>Hole Lower Tolerance</th>
<th>Shaft Upper Tolerance</th>
<th>Shaft Lower Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 0 to 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b) 3 to 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c) 6 to 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d) 10 to 18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e) 18 to 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f) 30 to 40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(g) 40 to 50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(h) 50 to 65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(i) 65 to 80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(j) 80 to 100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(k) 100 to 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(l) 120 to 140</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(m) 140 to 160</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(n) 160 to 180</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(o) 180 to 200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(p) 200 to 225</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q) 225 to 250</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(r) 250 to 280</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s) 280 to 315</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t) 315 to 355</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u) 355 to 400</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Do Modification**
  - Hole Upper Tolerance
  - Hole Lower Tolerance
  - Shaft Upper Tolerance
  - Shaft Lower Tolerance

### Procedure to modify assembly function to assembly view transformation knowledge

- **Figure D.9b**

---

232
D.10 Knowledge Layer to Support Flexible Information Integration between Views

As explained in chapter 7.6, three different experiments have been performed to show how the integration knowledge layer supports the flexible information integration of multiple views. The following sub sections describe the procedures of these three experiments.

D.10.1 Transformation of information between views

This experiment has been performed in four steps as follows:

A) Populating assembly functional views
B) Visualising populated information
C) Performing information transformation between views
D) Visualising populated & transformed information

Appendixes D-5 and D-6 are detail the procedures of the first two steps. Populated sub functional information can be transformed to assembly information by clicking transformation dropdown menu. The procedures to perform the transformation are illustrated in the figures 10.1a, 10.1b and 10.1c.

Figure 10.1a Procedures to perform transformation of information from functional view to assembly view (figure 1 of 3)
Figure 10.1b Procedures to perform transformation of information from functional view to assembly view (figure 2 of 3)
Figure 10.1c Procedures to perform transformation of information from functional view to assembly view (figure 3 of 3)
Populated sub functional view information and transformed assembly view information can be visualised by clicking *FeedAssMIC* button of *viewing* dropdown menu. The procedure of visualisation of multiple views information is illustrated in the figure 10.1d.

![Visualisation of Inputed views Information](image)

**Figure 10.1d Procedure to visualise multiple views information**

**D.10.2 Validation of transformed information**

The necessity of verification of transformed information has been detailed in the chapters 3 and 5. Chapter 7.6.3 describes the experiment to test the necessity of the view specific knowledge in the information integration of multiple views. In this experiment, information of sub-functional views have been transformed to machining view and then verified with machineability. The procedures to transform sub-functional view to machining view are similar to the experiment described in the above section. Figures 10.2.a and 10.2.b illustrates the procedures of the verification of transformed machining view information for machineability.
Figure 10.2a Procedure to transform information of sub functional view to machining view
Figure 10.2a Procedure to verify the information of machining view for machineability