Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage. [http://researchspace.auckland.ac.nz/feedback](http://researchspace.auckland.ac.nz/feedback)

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the [Library Thesis Consent Form](http://researchspace.auckland.ac.nz/feedback) and [Deposit Licence](http://researchspace.auckland.ac.nz/feedback).

Note: Masters Theses

The digital copy of a masters thesis is as submitted for examination and contains no corrections. The print copy, usually available in the University Library, may contain corrections made by hand, which have been requested by the supervisor.
Development of an Interoperable Cloud-based Manufacturing System

Xi (Vincent) Wang

A Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering

December 2012

Intelligent and Interoperable Manufacturing Systems (IIMS)
Department of Mechanical Engineering
Faculty of Engineering
The University of Auckland
Abstract

In modern manufacturing business, the supply chain consists of multiple tiers of partners and suppliers. Collaboration exists both among departments within an enterprise, and among the stakeholders within the supply chains. Communication and interaction between these participants are difficult due to the heterogeneous Information and Communication Technology environment, which consists of different protocols, data formats, programming languages and platforms.

To address the above issues, this research aims to develop a Cloud Manufacturing solution, namely, an Interoperable Cloud-based Manufacturing System (ICMS), to provide a manufacturing platform that integrates production services from design to manufacturing. Inspired by Cloud Computing technologies, the Cloud Manufacturing model provides an integration methodology to organise operational manufacturing tasks at a higher level. The STEP (STandard for Exchange of Product data) and STEP-NC (STandard for Exchange of Product data for Numerical Control) data models were enhanced to provide more complete data models for a fully developed Cloud Manufacturing service.

The ICMS system has three layers; the Manufacturing Capability Layer, Virtual Service Layer and Application Layer. The Manufacturing Capability Layer integrates and virtualises existing manufacturing abilities in the Manufacturing Cloud. The Storage Cloud is also developed at this layer to maintain the mirrored image and dynamic status of manufacturing capabilities. The Virtual Service Layer is organised based on the Smart Cloud Manager mechanism. This mechanism is responsible for analysing original user requests and mapping them to the virtualised Cloud Manufacturing Service in the Manufacturing Capability Layer. The original needs of the user are met by an appropriate service provider in the Cloud, without knowledge of the provider’s identity or its whereabouts. The Application Layer refers to the Cloud user domain. Different types of Cloud consumers are identified and assisted via different interaction environments.

The outcome of the research is the development of an interoperable and distributed manufacturing environment, which provides a platform that can integrate existing and future
manufacturing resources and abilities through Cloud Manufacturing Services. The system has been validated with capabilities, such as: (i) Smart Cloud Manager, able to take an original user request remotely and organise a series of Cloud services to fulfil the user request. It provides the neutral service establishment and organization mechanism for Cloud Manufacturing Paradigm; (ii) manufacturing resources, integrated in the Manufacturing Cloud and provided as a Cloud Service. Neutral data models compliant with international standards (e.g. STEP) were developed for the utilisation of Cloud Services. The current manufacturing resources are integrated into the Cloud Manufacturing environment, and the current STEP standard is extended to the Cloud area. (iii) the software agent and Function Block technology, utilised during the development of service integration. The amalgamation of these technologies forms a universal integration methodology called Virtual Function Block; (iv) a novel data exchange mechanism, the Data Packet mechanism, developed to provide an appropriate amount and scope of data to the right person.

Most of the research work in this thesis has been reported in eight research publications, i.e. four journal papers, one book chapter and three conference papers.

Keywords: Cloud, Cloud Manufacturing, Interoperability, STEP, STEP-NC, Data Exchange, Virtual Function Block
Acknowledgements

Many say doing PhD research is a journey. I began to believe that from Day One. As in a journey, I travelled to the southern hemisphere to take on this exciting research project. As in a journey, I went to countries overseas that I was not familiar with, to build up research connections and a new life from scratch. As in a journey, I have met so many wonderful people on the way, who shared their support, encouragement, knowledge and kindness with me. Those who are involved in the success of this research work, this page is for you.

First and foremost, I particularly thank my supervisor, Professor Xun Xu, who offered me the exciting opportunity to take on this research project. I can always remember the dizzy morning when I picked up the phone and thought it might be my roommate’s girlfriend. Professor Xu said on the other side of the ocean, “Hi, this is Xun Xu from the University of Auckland.” From that moment, I knew my journey had begun. During the past three and a half years, his supervision, guidance and professionalism throughout the project have given me consistent encouragement and support along the way. He guided me from being a “student” to an independent researcher.

I would like to express my sincere appreciation to my co-supervisor, A/Prof. Enrico Hämerle, for all his kind advice and contributions in paper writing. A special thank-you also goes to the Advisory Committee members of the University of Auckland, for their constructive comments and advice during my provisional year.

Many thanks to Sarath Pathirana and Kenneth Snow, the technicians of Manufacturing Systems Laboratory, for their kind assistance and suggestions during the development of the prototype system.

As in a journey, I have met great colleagues in the research group, Intelligent & Interoperable Manufacturing Systems (IIMS). I have been constantly surrounded and inspired by energetic and professional researchers. I would like to thank all the IIMS members for their support and comments that have always benefited my research; Tao Victor Peng, Zhiqian Jake Sang, Yuqian Michael Lu, Mick Ou, April Li, Prutvi Raj, Bins Pappachan and Ali Izadi. Much gratitude also goes to former IIMS members Dr. Derek Wang, Dr. Mohamad Bin Minhat, Dr.
Fiona Zhao, Dr. Firman Ridwan, Dr. Aini Zuhra Abdul Kadir, Dr. Terry Zhang and Dr. Ke Wang.

A special acknowledgment goes to Dennis Yeung for his contribution in enhancing the interpreter GUI for Creo Parametric. I also wish to thank Mary Bateman, for her kind assistance in proofreading this thesis.

I received very useful feedback and comments from a great number of experienced scientists during my visit to the Ecole Centrale de Nantes in France. I would like to take this opportunity to thank Professor Jean-Yves Hascoet and Associate Professor Matthieu Raunch for their advice on my research. I am thankful that in the midst of all their busy activities, they spent time sharing their knowledge and expertise with me, which has been very fruitful in shaping my ideas and research. I would also like to acknowledge my funding sources; the Chinese Scholarship Council (CSC), and Postgraduate Research Student Support (PReSS) funding from the University of Auckland. Without their sponsorship, this research would not have been possible.

I present this thesis as a special gift to my mother and father for their endless love and support throughout my life. Had it not been for their on-going encouragement and love, this thesis would have not been a reality. They have always been there for me, supporting and encouraging me all the times. I always do my best to be a son they can be proud of.

Finally, I owe loving thanks to my dear wife, Tianzi Wang. She has abandoned everything to accompany me in New Zealand and sacrificed a lot for my research. She is always the first reader of my writing, no matter how boring it is for a layperson. The view of her beautiful view making snacks during my late working nights is always the most wonderful sight in my life.

Xi (Vincent) Wang

4 December 2012
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAM</td>
<td>Application Activity Model</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>AIM</td>
<td>Application Interpreted Model</td>
</tr>
<tr>
<td>AP</td>
<td>Application Protocol</td>
</tr>
<tr>
<td>API</td>
<td>Application Protocol Interface</td>
</tr>
<tr>
<td>ARM</td>
<td>Application Reference Model</td>
</tr>
<tr>
<td>B2Cloud</td>
<td>Business-to-Cloud</td>
</tr>
<tr>
<td>BA</td>
<td>Broker Agent</td>
</tr>
<tr>
<td>BE-AU</td>
<td>BE AUstralia branch</td>
</tr>
<tr>
<td>BE-NZ</td>
<td>BE New Zealand branch</td>
</tr>
<tr>
<td>C2Cloud</td>
<td>Consumer-to-Cloud</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer-Aided Design</td>
</tr>
<tr>
<td>CAE</td>
<td>Computer-Aided Engineering</td>
</tr>
<tr>
<td>CAM</td>
<td>Computer-Aided Manufacturing</td>
</tr>
<tr>
<td>CAPP</td>
<td>Computer-Aided Process Planning</td>
</tr>
<tr>
<td>CAx</td>
<td>Computer-Aided capabilities</td>
</tr>
<tr>
<td>Computing</td>
<td>Cloud Computing</td>
</tr>
<tr>
<td>CIAgent</td>
<td>Customer Interface Agent</td>
</tr>
<tr>
<td>CIM</td>
<td>Computer-Integrated Manufacturing</td>
</tr>
<tr>
<td>CManufacturing</td>
<td>Cloud Manufacturing</td>
</tr>
<tr>
<td>CService</td>
<td>Cloud Service</td>
</tr>
<tr>
<td>CProvider</td>
<td>Cloud Provider</td>
</tr>
<tr>
<td>CCloud</td>
<td>Storage Cloud</td>
</tr>
<tr>
<td>CUser</td>
<td>Customer User</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>CU</td>
<td>Cloud User</td>
</tr>
<tr>
<td>DPacket</td>
<td>Data Packet</td>
</tr>
<tr>
<td>EIA</td>
<td>Enterprise Interface Agent</td>
</tr>
<tr>
<td>EUser</td>
<td>Enterprise User</td>
</tr>
<tr>
<td>FBlock</td>
<td>Function Block</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>IA</td>
<td>Interface Agent</td>
</tr>
<tr>
<td>ICMS</td>
<td>Interoperable Cloud-based Manufacturing System</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>MaaS</td>
<td>Manufacturing as a Service</td>
</tr>
<tr>
<td>MCapability</td>
<td>Manufacturing Capability</td>
</tr>
<tr>
<td>MCloud</td>
<td>Manufacturing Cloud</td>
</tr>
<tr>
<td>MResource</td>
<td>Manufacturing Resource</td>
</tr>
<tr>
<td>NC</td>
<td>Numerical Control</td>
</tr>
<tr>
<td>OKP</td>
<td>One-of-a-Kind Production</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PDM</td>
<td>Product Data Management</td>
</tr>
<tr>
<td>PDS</td>
<td>Product Design Specification</td>
</tr>
<tr>
<td>PLM</td>
<td>Product Lifecycle Management</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio-Frequency IDentification</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SAgent</td>
<td>Supervision Agent</td>
</tr>
<tr>
<td>SACloud</td>
<td>Service Application Cloud</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SCM</td>
<td>Smart Cloud Manager</td>
</tr>
<tr>
<td>SME</td>
<td>Small and Medium-sized Enterprise</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-Oriented Architecture</td>
</tr>
<tr>
<td>SProvider</td>
<td>Service Provider</td>
</tr>
<tr>
<td>ST</td>
<td>Service Template</td>
</tr>
<tr>
<td>STEP</td>
<td>STandard for the Exchange of Product data</td>
</tr>
<tr>
<td>STEP-NC</td>
<td>STandard for the Exchange of Product data for Numerical Control</td>
</tr>
<tr>
<td>UDF</td>
<td>User-Defined-Feature</td>
</tr>
<tr>
<td>VFB</td>
<td>Virtual Function Block</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Manufacturing</td>
</tr>
<tr>
<td>X2C</td>
<td>Everything-to-Cloud</td>
</tr>
<tr>
<td>XaaS</td>
<td>Everything as a Service</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
### Nomenclatures

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>Manufacturing Resource that includes all the resources required to carry out a service task, including hard resources $R_{Hard}$ and soft resources $R_{Soft}$</td>
<td></td>
</tr>
<tr>
<td>$i$</td>
<td>Phase number of Cloud Service</td>
<td></td>
</tr>
<tr>
<td>Cost($i$)</td>
<td>Cost of Cloud Service at Phase $i$</td>
<td>dollar</td>
</tr>
<tr>
<td>PD$_j$</td>
<td>Product Design as Service Phase $j$</td>
<td></td>
</tr>
<tr>
<td>S$_k$</td>
<td>Simulation as Service Phase $k$</td>
<td></td>
</tr>
<tr>
<td>CM$_l$</td>
<td>CNC Milling as Service Phase $l$</td>
<td></td>
</tr>
<tr>
<td>PW$_m$</td>
<td>Precision Welding as Service Phase $m$</td>
<td></td>
</tr>
<tr>
<td>G$_p$</td>
<td>Grinding as Service Phase $p$</td>
<td></td>
</tr>
<tr>
<td>SC$_{jk}$</td>
<td>Shipping Cost between Service Phase $j$ and $k$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>Time of Service</td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>Time of Shipping.</td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents

Abstract ...................................................................................................................................... i
Acknowledgements ................................................................................................................ iii
List of Abbreviations ............................................................................................................... v
Nomenclatures ...................................................................................................................... viii
Table of Contents .................................................................................................................... ix
List of Figures ....................................................................................................................... xiii
List of Tables ........................................................................................................................ xvii
List of Publications ............................................................................................................ xviii

Chapter 1. Introduction ........................................................................................................... 1
  1.1. Trends of the Manufacturing Industry ................................................................. 1
  1.2. Interoperability Issues ....................................................................................... 3
  1.3. Cloud Computing .............................................................................................. 5
  1.4. From Cloud Computing to Cloud Manufacturing ..................................................... 9
  1.5. Recap ....................................................................................................................... 12
  1.6. Thesis Structure Organisation ................................................................................. 13

Chapter 2. Literature Review ............................................................................................... 15
  2.1. Overview ................................................................................................................. 15
  2.2. Data-centric Interoperable Manufacturing Approaches ............................................ 16
    2.2.1. Use of a Proprietary Data Format .................................................................... 17
    2.2.2. Use of a Neutral Data Format .......................................................................... 17
    2.2.3. STEP/STEP-NC-based Research ..................................................................... 18
  2.3. Process-centric Interoperable Manufacturing Approach .............................................. 24
    2.3.1. Direct Data Translation .................................................................................... 24
    2.3.2. Two-way Data Conversion .............................................................................. 25
    2.3.3. Multiple Kernel Solutions ................................................................................ 27
    2.3.4. The Service-oriented Approach ....................................................................... 27
    2.3.5. Discussion and Summary ................................................................................ 30
  2.4. Cloud Manufacturing Approaches ........................................................................... 31
    2.4.1. Cloud Manufacturing ....................................................................................... 32
Chapter 3. System Framework .......................................................... 54

3.1. Overview ..................................................................................... 54
3.2. Advanced Manufacturing: Moving to the Cloud......................... 55
3.3. Manufacturing Capability and Manufacturing Resources .......... 58
3.4. Capability Virtualisation Philosophy ............................................. 61
3.5. System Architecture ................................................................... 66
  3.5.1. Application Layer: Customer and Enterprise User................. 69
  3.5.2. Virtual Service Layer .............................................................. 72
  3.5.3. Manufacturing Capability Layer ................................................. 75
3.6. Recap ......................................................................................... 77

Chapter 4. Smart Cloud Manager: Cloud Service Provision ............... 79

4.1. Overview ..................................................................................... 79
4.2. SCM Mechanism ........................................................................ 79
  4.2.1. SCM Involvement Levels ......................................................... 80
  4.2.2. SCM Process Flow ................................................................. 82
4.3. Cloud-based Data Models ............................................................. 85
  4.3.1. Service Provider Data Model .................................................... 86
  4.3.2. Cloud Service Template Data Model ........................................ 87
  4.3.3. Cloud Service Query Data Model ............................................. 91
  4.3.4. Cloud-based Data Model Set ..................................................... 96
4.4. Recap ......................................................................................... 99

Chapter 5. Service Application Cloud: Virtualised Manufacturing Capability .... 102

5.1. Overview ..................................................................................... 102
5.2. Function Blocks..................................................................................................... 103
  5.2.1. History of Function Blocks................................................................. 103
  5.2.2. IEC 61499 Standard and Function Blocks................................. 105
5.3. Software Agent...................................................................................................... 108
5.4. Virtual Function Block .......................................................................................... 111
  5.4.1. Cloud-Module-Agent Structure...................................................... 112
  5.4.2. Agent-based VFB ........................................................................ 112
  5.4.3. MTConnect to Integrate Hard Resource....................................... 115
  5.4.4. VFB and CMService................................................................. 118
5.5. VFB Data Model ................................................................................................... 121
5.6. Recap ..................................................................................................................... 123

Chapter 6. Storage Cloud: Product Data Exchange Mechanism ................................. 125
  6.1. An Overview of the STEP Data Model................................................ 125
    6.1.1. STEP – ISO 10303........................................................................ 125
    6.1.2. Objects of STEP........................................................................... 128
    6.1.3. STEP Architecture....................................................................... 128
    6.1.4. Application Protocols................................................................. 130
    6.1.5. Part 21 Implementation Methods................................................ 132
  6.2. STEP-NC ........................................................................................................... 133
    6.2.1. STEP-NC Content and Structure............................................... 136
    6.2.2. Machining Features................................................................. 138
    6.2.3. Machining Operation................................................................. 140
  6.3. Data Packet Mechanism.................................................................................... 141
    6.3.1. Collaborative Product Data Exchange..................................... 142
    6.3.2. Data-Localisation Mechanism.................................................. 144
    6.3.3. Data-Integration Mechanism................................................... 147
  6.4. Recap ..................................................................................................................... 150

Chapter 7. System Implementation and Case Studies................................................. 152
  7.1. Software Development Environment and Tools ..................................... 152
    7.1.1. NetBeans IDE.............................................................................. 152
    7.1.2. ST-Developer™ ......................................................................... 153
    7.1.3. Microsoft® Access....................................................................... 153
    7.1.4. Creo™ Parametric ....................................................................... 153
Table of Contents

7.2. Case Study 1: Cloud Service Provision ................................................................. 153
  7.2.1. Cloud Service for a Customer User ............................................................... 154
  7.2.2. Cloud Service for Enterprise User ................................................................. 158

7.3. Case Study 2: Data Packet Implementation .......................................................... 161
  7.3.1. Converting EXPRESS Schema into Java Class ............................................. 162
  7.3.2. Implementation of the ST-Developer Interfacing with Java ......................... 162
  7.3.3. Data Packet Generator ................................................................................... 163

7.4. Case Study 3: VFB-based Service Integration ...................................................... 167
  7.4.1. VFB Development ......................................................................................... 168
  7.4.2. Interpreter Development ................................................................................ 171

7.5. Recap ..................................................................................................................... 180

Chapter 8. Conclusions and Future Work ........................................................................ 182
  8.1. Conclusions ........................................................................................................... 182
  8.1.1. Summary of the Research .............................................................................. 182
  8.1.2. Intellectual Contributions .............................................................................. 185
  8.2. Recommended Future Work ................................................................................ 188

References ............................................................................................................................. 190

Appendix I ICMS Product Files ......................................................................................... 202
  A. Updated STEP Part 21 File for Example 1 of ISO14649 - Part 11 .................... 203
  B. Updated STEP Part 21 File for Example 2 of ISO14649 - Part 11 .................... 207
  C. Output of Data Packet VFB ................................................................................... 224

Appendix II Research Publication Abstracts .................................................................... 226
  A. Journal Papers ........................................................................................................ 227
  B. Book Chapter ....................................................................................................... 231
  C. Conference Papers ............................................................................................... 232
  D. Academic Posters ............................................................................................... 236
List of Figures

Figure 1.1 Percentage of Enterprises by Employee Size Group [2] ........................................ 3
Figure 1.2 Main Interoperation Difficulties of Systems in the CAD-CAM-CNC Chain [5] .... 4
Figure 1.3 Status of Post-Processing in the CAD-CAM-CNC Chain ................................. 5
Figure 1.4 Cloud Computing Models .................................................................................. 6
Figure 1.5 User Interaction with Cloud ............................................................................. 7
Figure 1.6 Market-oriented Cloud Architecture[16] ............................................................ 10
Figure 1.7 Layered Architecture of a Cloud Manufacturing System [17] ......................... 11
Figure 2.1 Manufacturing Data Chain within STEP ........................................................ 18
Figure 2.2 Universal Manufacturing Structure [38] ........................................................... 19
Figure 2.3 Data Exchange in a Typical Production Echelon at the Shop Floor Level [39] ... 20
Figure 2.4 STEP-NC Multi-process Manufacturing Concept [41] .................................... 20
Figure 2.5 Major Element Technologies of UbiDM [42] .................................................. 21
Figure 2.6 INFELT STEP Three-layered Structure [44] ................................................. 22
Figure 2.7 Schematic View of the Architecture [45] ....................................................... 23
Figure 2.8 Direct Data Translation ...................................................................................... 25
Figure 2.9 Two-way Data Conversion .............................................................................. 26
Figure 2.10 Module Users and Providers of the openCBM Approach [62] ....................... 28
Figure 2.11 Main Elements of the plug-and-play Framework [63] .................................... 29
Figure 2.12 NESIS Architecture [64] ............................................................................... 30
Figure 2.13 Cloud Manufacturing Platform [67] ............................................................... 32
Figure 2.14 Architecture of Cloud Manufacturing Running Model [70] ......................... 33
Figure 2.15 Processing of Manufacturing Service Descriptions in the MaaS Environment [74] ................................................................. 34
Figure 2.16 XMLAYMOD Structure and Layers [76] ......................................................... 35
Figure 2.17 Resource Virtualisation Model [78] ............................................................... 36
Figure 2.18 Structure of Cloud Computing-based Optimisation [98] .............................. 39
Figure 2.19 IBM SmartCloud [101] .................................................................................. 40
Figure 2.20 Federation and Delegation in Cloud Application Support ............................... 44
List of Figures

Figure 3.1 Traditional Manufacturing VS Cloud Manufacturing ............................................ 56
Figure 3.2 Collective Definition of MaaS (Manufacturing as a Service) ............................... 57
Figure 3.3 MCapability and MResource ........................................................................ 59
Figure 3.4 Cloud Service Virtualisation Process ................................................................. 62
Figure 3.5 CMServe Modelling Group ............................................................................. 63
Figure 3.6 Smart Cloud Manager Mechanism ................................................................. 65
Figure 3.7 ICMS Service-Oriented Architecture ................................................................. 67
Figure 3.8 ICMS Architecture ......................................................................................... 68
Figure 3.9 Customer User and Enterprise User ................................................................. 70
Figure 3.10 Virtual Service Layer ..................................................................................... 73
Figure 4.1 Multiple Engagement Levels of Cloud Users ..................................................... 81
Figure 4.2 Cloud Service Procedure ................................................................................ 82
Figure 4.3 Logic Flow within Smart Cloud Manager .......................................................... 84
Figure 4.4 Cloud Service Provider Model in EXPRESS-G .................................................. 86
Figure 4.5 Cloud Service Model ...................................................................................... 88
Figure 4.6 Service Request Model ................................................................................. 92
Figure 4.7 Meta-Model of Service Request .................................................................... 95
Figure 4.8 EXPRESS-G Models Supporting Cloud Services ............................................. 98
Figure 5.1 Basic Function Block Type: Interface, ECC and Algorithm REQ ..................... 105
Figure 5.2 Implementing X2-Y2 as a Network of Function Blocks .................................... 106
Figure 5.3 Example of an FBlock Type Conformant to IEC 61499 .................................... 107
Figure 5.4 General Function Block Model ....................................................................... 108
Figure 5.5 A Partial View of an Agent Typology [189] ....................................................... 109
Figure 5.6 Three Types of Software Agent ...................................................................... 110
Figure 5.7 Three-tier Architecture ................................................................................. 112
Figure 5.8 MResource Encapsulated by Virtual Function Block ....................................... 113
Figure 5.9 Composite VFB ............................................................................................ 114
Figure 5.10 VFB Improving the Network Communication ................................................ 115
Figure 5.11 Integration of Manufacturing System with MTConnect ................................ 116
Figure 5.12 Sample Device in MTConnect Description [195] .......................................... 117
Figure 5.13 VFB-based CMServe Procedure .................................................................. 119
Figure 5.14 Creo VFB Module ....................................................................................... 120
Figure 5.15 VFB Data Model ......................................................................................... 122
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Design-Manufacturing Data Exchange Enabled by STEP</td>
<td>127</td>
</tr>
<tr>
<td>6.2</td>
<td>Structure of STEP standard [26]</td>
<td>129</td>
</tr>
<tr>
<td>6.3</td>
<td>AAM for Digital Manufacturing [33]</td>
<td>131</td>
</tr>
<tr>
<td>6.4</td>
<td>Example of Part 21 File</td>
<td>133</td>
</tr>
<tr>
<td>6.5</td>
<td>CAx Chain Enabled by STEP-NC</td>
<td>135</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparison of G-code and STEP-NC Data [209]</td>
<td>136</td>
</tr>
<tr>
<td>6.7</td>
<td>Structure of the STEP-NC Data Model</td>
<td>137</td>
</tr>
<tr>
<td>6.8</td>
<td>Geometry, Technology and Process Information in STEP-NC [33]</td>
<td>139</td>
</tr>
<tr>
<td>6.9</td>
<td>Examples of STEP-NC Features</td>
<td>140</td>
</tr>
<tr>
<td>6.10</td>
<td>Synchronisation and Confidentiality Issues for Data-centric Approaches</td>
<td>141</td>
</tr>
<tr>
<td>6.11</td>
<td>Manufacturing Data at Different Levels</td>
<td>143</td>
</tr>
<tr>
<td>6.12</td>
<td>Data Packet Concept</td>
<td>144</td>
</tr>
<tr>
<td>6.13</td>
<td>Parent-Child Logic</td>
<td>145</td>
</tr>
<tr>
<td>6.14</td>
<td>Data Packet Analysis Process</td>
<td>146</td>
</tr>
<tr>
<td>6.15</td>
<td>Meta-Model Example</td>
<td>147</td>
</tr>
<tr>
<td>6.16</td>
<td>Data Integration</td>
<td>148</td>
</tr>
<tr>
<td>6.17</td>
<td>Data Packet Model</td>
<td>149</td>
</tr>
<tr>
<td>7.1</td>
<td>ICMS Interface</td>
<td>154</td>
</tr>
<tr>
<td>7.2</td>
<td>XML Representation of a Service Request</td>
<td>155</td>
</tr>
<tr>
<td>7.3</td>
<td>Cloud Service Results Pool</td>
<td>156</td>
</tr>
<tr>
<td>7.4</td>
<td>Product Example: T14859 Driver</td>
<td>159</td>
</tr>
<tr>
<td>7.5</td>
<td>Integrated CMService Data</td>
<td>160</td>
</tr>
<tr>
<td>7.6</td>
<td>Typical STEP-based Production Procedure</td>
<td>161</td>
</tr>
<tr>
<td>7.7</td>
<td>Converting EXPRESS Schema into Java Classes</td>
<td>162</td>
</tr>
<tr>
<td>7.8</td>
<td>Importing ST-Developer Library into Java</td>
<td>163</td>
</tr>
<tr>
<td>7.9</td>
<td>STEP-NC Data Structure</td>
<td>164</td>
</tr>
<tr>
<td>7.10</td>
<td>STEP-NC Interpreter/DPacket Generator</td>
<td>165</td>
</tr>
<tr>
<td>7.11</td>
<td>Data Packet Export Options</td>
<td>165</td>
</tr>
<tr>
<td>7.12</td>
<td>Converting Entity Instances into Java Vectors</td>
<td>166</td>
</tr>
<tr>
<td>7.13</td>
<td>DPacket Collection Method</td>
<td>166</td>
</tr>
<tr>
<td>7.14</td>
<td>Output Data Packet File</td>
<td>167</td>
</tr>
<tr>
<td>7.15</td>
<td>VFB Case Study</td>
<td>168</td>
</tr>
<tr>
<td>7.16</td>
<td>Application Registration and Command</td>
<td>169</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7.17</td>
<td>Socket-Server Communications between Virtual Function Blocks</td>
<td>170</td>
</tr>
<tr>
<td>7.18</td>
<td>Remote Communication Structure</td>
<td>171</td>
</tr>
<tr>
<td>7.19</td>
<td>Web-based Messenger</td>
<td>171</td>
</tr>
<tr>
<td>7.20</td>
<td>VFB Functions Integrated with Creo Interface</td>
<td>172</td>
</tr>
<tr>
<td>7.21</td>
<td>Mapping Coordinate Systems</td>
<td>172</td>
</tr>
<tr>
<td>7.22</td>
<td>Configurations for Workpiece Information</td>
<td>173</td>
</tr>
<tr>
<td>7.23</td>
<td>ISO 14649 Example 1 in Creo Parametric</td>
<td>174</td>
</tr>
<tr>
<td>7.24</td>
<td>Contents of a Closed Pocket UDF</td>
<td>175</td>
</tr>
<tr>
<td>7.25</td>
<td>Diagram for Angle Taper</td>
<td>176</td>
</tr>
<tr>
<td>7.26</td>
<td>Sketch Used in a UDF of a Round Hole with a Tapered End</td>
<td>176</td>
</tr>
<tr>
<td>7.27</td>
<td>Creo Compound Feature Mapped to Creo Feature Group (ISO14649-11 example 2)</td>
<td>177</td>
</tr>
<tr>
<td>7.28</td>
<td>Replicate Feature Example</td>
<td>178</td>
</tr>
<tr>
<td>7.29</td>
<td>Extracting Manufacturing Operations from a Reference Model</td>
<td>179</td>
</tr>
<tr>
<td>7.30</td>
<td>Tool Path for a Plane Rough Milling Operation</td>
<td>180</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1 Comparison between Data-centric and Process-centric Approaches .................. 31
Table 2.2 Major Cloud Service Providers.............................................................................. 41
Table 7.1 Sequence Matrix of Cloud Service Phases ............................................................ 157
Table 7.2 Cloud Service Mapping Results from Smart Cloud Manager ........................... 158
Table 7.3 Task List of the Case Study .................................................................................. 169
List of Publications

Journal Papers


Book Chapter


Conference Papers


Chapter 1. Introduction

Manufacturing business faces great challenges from global competition. Process Interoperability and Data Interoperability issues have been identified as two of the key issues in manufacturing. Recently, Cloud Computing technology has made a great impact on the information and communication paradigm. It provides opportunities to improve manufacturing interoperability, thus forming a new manufacturing model: Cloud Manufacturing. In this chapter, the concept of Cloud Computing (CManufacturing) is introduced. A prospective view of which CManufacturing (CManufacturing) systems should prevail in manufacturing is also discussed. They represent the basis of this research in formulating research objectives. The scope of the research is also outlined. The structure of this thesis is presented at the end of this chapter.

1.1. Trends of the Manufacturing Industry

Over the years, manufacturing business has become increasingly challenging and competitive. Globalisation makes manufacturing business no longer a game “in-house”. Resources, materials, knowledge and expertise are often outsourced and shared among business participants world-wide.

Affordable aviation and low-cost shipping service have fostered a competitive international market. Meanwhile, the rise of the Internet and WWW (World Wide Web) has cut the cost of communication among different countries. Regardless of the location of participants, engineering design, accounting, data transition and computing can be achieved over a connected network, which has led to a widely distributed manufacturing environment. Suppliers are not clustered geographically. Factories and shop floors are located anywhere to secure either “close-to-resources” or “close-to-market”. A highly globalised market does not only mean cheaper and better resources. It also leads to challenges from business competitors all around the world.

These days, it is almost impossible to carry out production tasks without the support of suppliers, contractors, retailers and many other types of business partners. Collaboration exists at every operational stage of a supply chain. Interactions between business participants
take the form of material (e.g. products and raw material) and nonmaterial (e.g. funds, leasehold and knowledge). The term, virtual enterprise, describes the temporary alliance of businesses in which resources, skills, facilities and core competencies are shared and integrated. Thus a broad virtual enterprise concept surfaces, which is a consortium of different departments and companies that come together to quickly exploit fast-changing worldwide product manufacturing opportunities [1]. In a virtual manufacturing organisation, enterprises are integrated through cooperation, contracts, trading, hiring, etc. Costly resources and risks can be shared and distributed among business partners. Manufacturing tasks that cannot be accomplished individually are now attainable. The collaborative network is able to respond faster and better to business opportunities and a dynamic market. Meanwhile, individual enterprise performance is improved due to a higher level of strategic thought and a wider pool of knowledge and experiences.

According to the New Zealand Ministry of Economic Development [2], a Small and Medium-sized Enterprise (SME) is defined as an enterprise with 19 or fewer employees, which means over 90 percent of manufacturing enterprises in this country are SMEs (Figure 1.1). To survive increased competition, it is even more important for them to seek partnerships and achieve the economies of scale that larger businesses benefit from. Motivated by cost reduction and better service performance, allied SMEs are able to share the risks and opportunities. In a background of globalised and widely distributed manufacturing, it is even more important for manufacturers to collaborate for mutual benefit and competency in the market place.
1.2. Interoperability Issues

Unfortunately, the integration of enterprises cannot be achieved automatically. Interoperability issues are often tangled with risks, conflicts and complex communications among stakeholders. Connectivity between business partners requires a highly collaborative, well distributed and agile regime. Because of this, interoperability has been identified as one of the most challenging technical issues that manufacturing industry is facing.

Recently, Microsoft® [3] commissioned a survey of over 152 Information Technology (IT) and business decision makers in manufacturing enterprises, including automotive, high-tech and electronics, industrial machinery manufacturers and related companies. The results show that the top stress faced by the IT system of a company is facing comes from globalisation, increased competition in the global market and shorter product lifecycles. In this investigation, the integration of collaboration tools and lack of access to data and processes are addressed as the biggest challenges within the current IT infrastructures. In practice, it is difficult to interconnect software systems and integrate their output at the data level.
In today’s manufacturing industry, production and related business processes are based on various software applications that form the “CAD-CAM-CNC” chain [4]. Product information is created by Computer-Aided Design (CAD) systems, while the Computer-Aided Manufacturing (CAM) systems enable users to add manufacturing information to their design, such as machining processes, tool paths and fixtures. The output of a CAM system is usually a Numerical Control (NC) part program. Such a program is then read by a Computer Numerical Control (CNC) system as input and to operate the machining unit. In addition to the systems mentioned above, other software products exist, i.e. Computer-Aided Engineering (CAE) and CAPP (Computer-Aided Process Planning) tools. Product Data Management (PDM) and Product Lifecycle Management (PLM) systems are used to deal with product data, from design and manufacture to service and disposal. All of these systems are developed by using different software tools, data formats, interfaces and databases, thus forming a highly heterogeneous data environment. Based on a survey among 251 executive officers of German-speaking enterprises [5], the main interoperation difficulties of systems in the CAD-CAM-CNC chain are stated in Figure 1.2. It is evident that more than 75 percent of the large problems (light blue columns in the figure) are directly related to the use of different CAD versions or systems, different file formats and conversions.

![Figure 1.2 Main Interoperation Difficulties of Systems in the CAD-CAM-CNC Chain [5]](image)

There are even more obstacles to achieving interoperability at the tail end of the CAD-CAM-CNC chain. Currently, there are more than 4500 configurations just for post-processing data from a CAM system to a machine tool (Figure 1.3). Despite improvements in design and manufacturing, the current situation in NC programming is still that of being compartmentalised and isolated. Over the years, machine tools have been improved immensely. The programming language is, however, still based on the G & M codes, which
only document instructional and procedural data, leaving most of the design and process information behind. This machine-specific part programme generated by a postprocessor supports only a one-way data flow (i.e. from CAM to CNC). Data translation and conversion are common ways of coping with data heterogeneity in the CAD-CAM-CNC chain. These exercises often lead to huge data loss. Thus there is a strong need for an integration platform throughout production.

1.3. Cloud Computing

In recent years, Cloud Computing has evolved from web-based technologies, e.g. Grid Computing, Peer-to-Peer, Client-Server model, etc.. Cloud Computing is the usage of computing resources (e.g. hardware and software) as a service over a network. It provides new models and opportunities to integrate business processes and resources, including manufacturing. According to the definition from National Institute of Standards and Technology (NIST) [6], Cloud Computing is “a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. This Cloud model is composed of three service models, four deployment models (Figure 1.4) and five essential characteristics.
The three typical types of Service Models are:

- **Software as a Service (SaaS).** The capability provided to the consumer is the use of the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g. web-based email), or a program interface. The consumer does not manage or control the underlying Cloud infrastructure (i.e. network, servers, operating systems, storage), or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

- **Platform as a Service (PaaS).** The capability provided to the consumer is to deploy onto the Cloud infrastructure, consumer-created or acquired applications created using programming languages, libraries, services and tools supported by the provider. The consumer does not manage or control the underlying Cloud infrastructure, including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

- **Infrastructure as a Service (IaaS).** The capability provided to the consumer is to provide processing, storage, networks and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the
underlying Cloud infrastructure but has control over operating systems, storage and deployed applications, and possibly limited control on selected networking components (e.g. host firewalls).

In different service models, the level of interaction needed from the Cloud user is different (Figure 1.5). From traditional computing infrastructure to SaaS mode, the user engagement with the system decreases. Accordingly, Management and maintenance increasingly depend on the service provider from traditional solutions to SaaS.

Deployment Models:

- **Private Cloud.** The Cloud infrastructure allows exclusive use by a single organisation comprising multiple consumers (e.g. business units). It may be owned, managed and operated by the organisation, a third party or some combination of them, and it may exist on or off premises.
• **Community Cloud.** The Cloud infrastructure is provided for exclusive use by a specific community of consumers from organisations that have shared concerns (e.g. mission, security requirements, policies and compliance considerations). It may be owned, managed and operated by one or more of the organisations in the community, a third party or some combination of them, and it may exist on or off premises.

• **Public Cloud.** The Cloud infrastructure is supplied for open use by the general public. It may be owned, managed and operated by a business, academic, or government organisation or some combination of them. It exists on the premises of the Cloud provider.

• **Hybrid Cloud.** The Cloud infrastructure is a composition of two or more distinct Cloud infrastructures (e.g. private, community or public) that remain unique entities, but are bound together by standardised or proprietary technology that enables data and application portability (e.g. Cloud bursting for load balancing between Clouds).

**Essential Characteristics:**

• **On-demand self-service.** A consumer can unilaterally use computing capabilities, such as server time and network storage as needed, automatically without requiring human interaction with each service provider.

• **Broad network access.** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g. mobile phones, tablets, laptops and workstations).

• **Resource pooling.** The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence, in which the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (e.g. country, state or datacenter). Examples of resources are storage, processing, memory and network bandwidth.

• **Rapid elasticity.** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward, commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
• Measured service. Cloud systems automatically control and optimise resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g. storage, processing, bandwidth and active user accounts). Resource usage can be monitored, controlled and reported, providing transparency for both the provider and consumer of the utilised service.

**1.4. From Cloud Computing to Cloud Manufacturing**

In recent times, platforms have been provided by the major software vendors to support these Cloud concepts or architecture, such as Amazon’s [7] Elastic Compute Cloud (EC2), Google’s [8] App engine, Microsoft’s [9] and Sun’s Cloud [10]. Applications based on the Cloud concept are making big impacts on businesses, e.g. iCloud [11], Google Play [12], and Microsoft Office 365 [13]. According to recent Forrester research [14], CComputing business reached $40.7 billion globally in 2010 while impacting the $948 billion Information and Communication Technology (ICT) market. Even though some obstacles still exist, for instance unpredictability and confidentiality, it is also foreseen that the Cloud Computing market will keep growing and developing in the future [15].

Among various types of models, the key characteristic of CComputing is not to pay the whole cost of a hardware/software resource, instead of paying for the amount of service provided (pay-as-you-go). The barriers of entry for costly computing and storage devices become much lower. The true spirit of the Cloud concept can be summarised as the capability of providing globally distributed, fast-responding, on-demand and quantifiable services.

Figure 1.6 illustrates a typical market-oriented Cloud architecture. Physical machine equals the collection of computing hardware, e.g. computers, servers, data centers etc.. Virtual Machine is software that is implemented as abstraction of these hardware. In general, Physical Machines are presented as Virtual Machines at the application layer of a Cloud system and virtualised and provided as scalable services. Thus, it is possible to map manufacturing resources to the Cloud-based architecture and deploy computing capabilities and hardware in a service-centric environment. In the past few decades, a number of novel technologies have been proposed to improve the environment of the manufacturing industry, e.g. collaborative manufacturing, virtual manufacturing, agile manufacturing, etc. To promote the integrated manufacturing environment to a higher level, it is desirable to adapt and adopt the Cloud concept in manufacturing. Cloud technology provides a promising solution and an
even broader definition for the concept of “Design Anywhere and Make Anywhere (DAMA)”. A Cloud version of DAMA can be described as completing, designing and manufacturing procedures via Clouds, while users and business partners are loosely networked by a Cloud-based network, regardless of how and where the specific manufacturing facilities are. Interoperability can also be enhanced by Cloud, since it is the creation of an agreed-upon framework/ontology, syntax or open protocols/APIs that makes migration and integration of applications and data much easier. Interoperability allows applications to be ported between Clouds, infrastructures or providers before the service is delivered from the Cloud.

![Market-oriented Cloud Architecture](image)

Much as in CComputing, manufacturing infrastructure, platform and software application can be packaged as a service to be delivered to a Cloud User (CU). Extending the concept to a broader scope, all production objects, tasks and features can be provisioned as a service, i.e. Everything as a Service (XaaS). Capabilities (e.g. production, design, analysis and engineering abilities) can be published and shared on the Cloud. This forms the CManufacturing concept.

Based on NIST’s Cloud Computing definition, CManufacturing can be defined as a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable manufacturing resources (e.g. manufacturing software tools, manufacturing
equipment and manufacturing capabilities), which can be rapidly provisioned and released with minimal management effort or service provider interaction [17]. A typical CManufacturing framework should contain four systematic layers, i.e. a manufacturing resource layer, virtual service layer, global service layer and application layer (Figure 1.7).

![Figure 1.7 Layered Architecture of a Cloud Manufacturing System [17]](image)

These days, a manufacturing business may not survive in the competitive market without the support of Computer-Aided capabilities (CAx) and IT. Cloud technology can improve the environment through product design, manufacturing process management, enterprise resource planning and manufacturing resource management.

Users can benefit from Cloud Computing features such as reduced cost, flexibility, mobility and automation. In the CManufacturing environment, it is not necessary for an enterprise to acquire the entire software/hardware package that is only occasionally used. Based on the “pay-as-you-go” principle, this cost can be reduced and so can maintenance and labour expense. The user is provided with more flexible computing methods. There are more choices for companies to process a specific task, and more freedom to launch/terminate a service, with fewer issues. Users/employees can access the application/information via various devices connected to the network or Cloud. It provides an organisation with an environment of distributed collaboration. From the service provider’s perspective, the cost of service is reduced as well. Updating and maintenance procedures are simplified. By amending the applications in the Cloud, traditional shipping and repackaging costs are avoidable. For both
Cloud users and providers, application release and updating is made easier. By updating the Cloud, personal/business users get the latest product and service without additional effort.

1.5. Recap

Cloud technology creates new opportunities for the manufacturing industry by providing open sharing methods between business stakeholders located around the world. Business risks and opportunities are shared on the partner network that is organised by the Cloud. In the globalised manufacturing context, customer-oriented manufacturing is a promising approach to improving service quality and competitiveness. In the CManufacturing environment, customised requests can be fulfilled by the elastic characteristic of a Cloud. Manufacturing applications and resources can be easily integrated with interoperability and flexibility.

CManufacturing solutions can be divided into two categories. The first category concerns deploying manufacturing software on the Cloud, i.e. a “manufacturing version” of CComputing. CAx software can be supplied as a service on the Manufacturing Cloud (MCloud). The second category has a broader scope, cutting across production, management, design and engineering abilities in a manufacturing business. Unlike with computing and data storage, manufacturing involves physical equipment, monitors, materials and so on. In this kind of CManufacturing system, both material and non-material facilities are implemented on the Manufacturing Cloud to support the whole supply chain. Costly resources are shared on the network. This means that the utilisation rate of rarely-used equipment rises and the cost of expensive equipment is reduced. According to the concept of Cloud technology, there will not be direct interaction between Cloud Users and Service Providers. The CU should neither manage nor control the infrastructure and manufacturing applications.

To facilitate the second type of CManufacturing architecture throughout the whole supply chain, the main thrust of this research was to develop CManufacturing architecture to support interoperable manufacturing tasks from design to production. This research project proposes a new CManufacturing system that supports a collaborative and interoperable manufacturing environment. The following tasks were undertaken:

(1) developing an Interoperable Cloud-based Manufacturing System
Chapter 1 – Introduction

(2) developing a service-oriented mechanism to virtualise manufacturing abilities as Cloud Services

(3) developing an integration mechanism to control and componentise CManufacturing Services

(4) developing an advanced data-sharing mechanism to support communications within Cloud.

1.6. Thesis Structure Organisation

This research is focused on developing a Cloud-based manufacturing system that integrates existing CAx applications and manufacturing abilities.

Chapter 2 gives a thorough review of interoperable manufacturing and CManufacturing approaches in recent years. It identifies the research trends in Cloud-related manufacturing methods. Following the review is an investigation of technologies that can potentially support a CManufacturing system in the areas of CComputing, information systems and enterprise systems. Finally, the research gaps are identified at the end of this chapter.

In Chapter 3, the framework of the proposed CManufacturing system is presented. It consists of three layers: the Smart Cloud Manager Layer, which acts as the brain or supervisor to control and coordinate the Cloud activities; the User Cloud Layer, which interacts with distributed Cloud clients connected by the Internet; and the Manufacturing Cloud Layer, which integrates existing manufacturing applications, resources and their providers.

Chapter 4 describes the Smart Cloud Manager, which consolidates CManufacturing services. Explanations are given for service-oriented architecture, followed by the classification of manufacturing capability and manufacturing resource. As the bridge between Cloud client and manufacturing applications, Smart Cloud Manager maps an original user request to a specific capability and the resources that lie behind it. The mechanism of Smart Cloud Manager and supporting data models are given.

Chapter 5 describes the Application Cloud, which is the first half of the Manufacturing Cloud Layer. The IEC 61131 standard was introduced into the system to facilitate a componentised manufacturing service model. Combining function blocks with software agents,
manufacturing capabilities are packaged as service modules by controlling the event and data flows of virtual function blocks.

Chapter 6 gives the second half of the Manufacturing Cloud Layer, which is the Storage Cloud. The synchronisation and confidentiality challenges experienced by the industry are discussed in depth. To address these issues, a neutral data format was utilised in the Storage Cloud, and a related data integration/localisation mechanism is proposed. With the help of a novel philosophy, the Data Packet, product data is able to be partially shared over Cloud and integrated after processing.

In Chapter 7, the implementation of the proposed system is explained. Algorithms and methodologies were developed and implemented in the software prototype and were discussed in detail. This is followed by the introduction of three case studies that were used to test the proposed CManufacturing concept and data models.

Chapter 8 gives a detailed discussion of the proposed system and the methods that support it, including their advantages and benefits. Then conclusions are drawn for further research, ending with suggestions for future research and recommendations.
Chapter 2. Literature Review

Manufacturing interoperability can be achieved in two ways; data-centric interoperability based on common data models and process-centric interoperability based on format conversions. This chapter first reviews the systems in these two approaches. As a service-oriented concept, CManufacturing can be understood as a new process-centric approach. Next Cloud Computing is discussed from a manufacturing perspective. Parallel manufacturing technologies are reviewed. Finally, based on the review, research gaps and motivation are identified. Major work in this chapter is summarised in a paper for the Intentional Journal of Production Research [18].

2.1. Overview

In a modern manufacturing business, collaboration exists not only among its own departments, but also among its business partners. Advanced manufacturing technologies are consistently reshaping the landscape of global competition. The connections among these partners are required in an agile, highly collaborative and distributive environment. In such an environment, a better means of communication and integration of existing CAx systems is a necessity among different organisations and processes. A tight data flow, sharing and exchanging data, is also essential.

As the Cloud concept is implemented in terms of computing, infrastructure, platform, and software application can be directly provisioned as a service to the CU. The research question is how to integrate physical facilities and machines. By extending the SOA concept to a broader range, all production objects and features can be provided as a service, which is XaaS. CUs will have no additional hardware or software to buy, install, maintain or update. Alternatively, the resources can be delivered as self-contained service packages.

As a new model of a manufacturing network, CManufacturing combines Cloud Computing with manufacturing under SOA. It is set to fundamentally change how products are designed, manufactured, shipped and maintained. The workload within the manufacturing will shift to higher level functions, which are provided by public Cloud suppliers. CManufacturing enables a methodology for manufacturing process as a service application. So the manual
process is digitalised and integrated into a common platform. In this instance, the component makes the service more flexible, cheaper and easier to deliver than any previous structure. From the viewpoint of the manufacturer, the integration and interoperability of a business network can be achieved more easily. Thus production capacity and performance is improved.

Moving to the Cloud would be an on-going transformation. It would lead the industry to a sustainable environment with intelligence, flexibility and interoperability. In recent years, research has been carried out world-wide in an attempt to develop an interoperable and collaborative environment with heterogeneous software applications. In the following part of this section, recent research is reviewed and discussed in regard to CManufacturing research and related technologies seeking an interoperable manufacturing environment.

## 2.2. Data-centric Interoperable Manufacturing Approaches

There are two types of interoperability; process interoperability and data interoperability. Data interoperability is defined as the ability to communicate product data across different production activities [19]. Process interoperability focuses on service exchanges between systems and business departments. It can be defined as the ability to accomplish end-user applications using different types of computer systems, operating systems and application software [20]. Unfortunately, system integration and interoperability is addressed as one of the key issues in a manufacturing firm, according to Panneto and Molina’s research [21].

There are two broad categories of approach to achieve an interoperable manufacturing environment: the data-centric approach and the process-centric approach. With a data-centric approach, all software applications use the same data syntax, whereas with a process-centric approach, integration is achieved at the process level with multiple data formats accommodated. To promote the current manufacturing systems to a higher level, it is possible to adopt the research contributions from both categories into CManufacturing systems that further enhance a collaborative and intelligent manufacturing environment.

A widely recognised information model is needed, particularly for a collaborative and distributed environment. As mentioned above, syntax harmonisation is the prerequisite for the data-centric approach. To achieve such harmonisation, researchers and software vendors have used a number of methods, e.g. use of proprietary data formats, neutral data formats and international standard formats.
2.2.1. Use of a Proprietary Data Format

Use of proprietary data formats has been considered by both researchers and software vendors. CATIA and its PLM system ENOVIA® provide a production-proven SOA to support collaborative manufacturing. Siemens NX and their PLM solution Teamcentre® and Sinumerik® 840D NC solutions aim to offer a complete solution for industries, such as automotive, from design all the way to manufacturing. Likewise, PTC’s Creo™ together with its PLM solution, Windchill®, caters for product information throughout all engineering processes, with associative CAD, CAM and CAE applications spanning from conceptual design to NC tool-path generation. Proprietary data formats are dependent on solid modelling kernels and their wrappers. Data portability is readily achieved across different CAD/CAM systems, as the same kernel was deployed.

2.2.2. Use of a Neutral Data Format

It has been suggested that a commonly used data model/schema should be utilised for a wide range of a [22]. The data management should be encapsulated by schema and manipulating rules on one data model. During the last few decades, many neutral data formats have been developed to enable data exchange. The most commonly used formats for geometric data exchange are the Drawing Exchange Format (DXF) [23], the Initial Graphics Exchange Standard (IGES) [24], the Product Data Exchange Specification (PDES) [25] and the STandard for the Exchange of Product data (STEP ISO 10303) [26].

A number of modelling languages may also be used, for example; Web Services Description Language (WSDL) for web-service description, Web Ontology Language (OWL) for knowledge representations and Web Services Business Process Execution Language (WS-BPEL) for executable business process with web-services. In parallel, Universal Description Discovery and Integration (UDDI) provides a mechanism to register and locate web service applications, and DARPA Agent Markup Language (DAML) provides machine-readable representations for the Web. However, these methods are not capable of supporting automatic discovery of manufacturing service capabilities [27].

To standardised method to describe, model and support the supply chain, international standards, e.g. STEP and especially the Standard for Exchange of Product data for Numerical Control (STEP-NC), can play a central role in the CManufacturing environment. STEP is one of the most successful standards, providing mechanisms for describing product information.
throughout the product lifecycle. Being different from its predecessors, STEP provides a neutral mechanism by specifying a form capable of describing the entire product data throughout the life cycle of a product. STEP consists of different Application Protocols (APs), which provide data models relevant to individual targeted applications, activities or environments. As illustrated in Figure 2.1, different types of information can be stored in a STEP file; individual software suits can extract the information relevant to the application itself. For instance, a CAM solution is able to read the information to acquire a single manufacturing process and pass it on to a CNC machine without any need for other data. Compared with previous standards, these data models offer a set of effective tools for interoperability solutions in the computer-aided manufacturing context [28].

![Figure 2.1 Manufacturing Data Chain within STEP](image)

In practice, a STEP-compliant information framework can be built based on the Standard Data Access Interface (SDAI). SDAI defines an abstract Application Programming Interface (API) to work on application data according to a given data model defined in EXPRESS [29], C++ [30], C [31] and JAVA language [32]. Detailed interfaces are defined to achieve portability from one implementation to another.

### 2.2.3. STEP/STEP-NC-based Research

In 2003, the data model for computerised numerical controllers was formally published as an international standard that is also known as STEP-NC [33, 34]. As a data model to connect CAD/CAM systems with CNC machines, STEP-NC completes the integrated loop of CAD/CAM/CNC. It remedies the shortcomings of the traditional G-Code centred language (ISO 6983) [35] by specifying machining processes instead of machine tool movements. Data at the task level (what-to-do) is defined in a STEP-NC file rather than “how-to-do” data at the method level. It has been proved that STEP-NC can support both system interoperability and
Chapter 2 – Literature Review

data traceability [36]. From the CManufacturing perspective, it is possible to achieve interoperability by adopting international standards that address various stages or applications in the STEP-based manufacturing chain.

Nessahi et al. [37] proposed a framework to address the incompatibility issue occurring in heterogeneous CAx environments. Much as in an IaaS structure, software tools are connected to the platform through specific interfaces, acting as “plug-and-play” applications. The integrated applications work with Intercommunication Bus, Manufacturing Data Warehouse, and Manufacturing Knowledgebase as shown in Figure 2.2 [38]. In this architecture, mobile agents are utilised to support the communication bus and CAx interfaces, while STEP is the neutral data format for different applications. STEP is utilised as the basis for representing manufacturing knowledge augmented with XML (eXtensible Markup Language) schema, while a comprehensive data warehouse is utilised to store CNC information. The plug-and-play feature is specifically suitable for application integration via the Cloud approach.

More recently, Mokhtar and Houshmand [39] reported a similar manufacturing platform, using axiomatic design theory to realise interoperability and production optimisation (Figure 2.3). The methodology of axiomatic design is used to generate a systematic roadmap of an optimum combination of data exchange via direct (using STEP), or indirect (using bidirectional interfaces), solutions in the CAx environment. This research work provides some insight into how manufacturing resources (device and software tools) can be described and encapsulated, and how the resources can be utilised and organised in a Global Service Layer.
Chapter 2 – Literature Review

Figure 2.3 Data Exchange in a Typical Production Echelon at the Shop Floor Level [39]

Laguionie et al. studied a manufacturing system that can integrate a multi-process numerical chain [40, 41]. This system is called STEP-NC Platform for Advanced and Intelligent Manufacturing (SPAIM). Manufacturing processes are connected in the system through a standardised data exchange carrier, i.e. STEP-NC. Figure 2.4 shows how the manufacturing process can be integrated via a neutral data format and how manufacturing procedures may be organised at the IaaS level.

UbiDM®, a concept of Design and Manufacture via Ubiquitous computing technology, has been proposed in [42]. The key aspect of UbiDM® is the utilisation of the entire product lifecycle information obtained via ubiquitous computing technologies, for product design and manufacture (Figure 2.5). A unified product lifecycle data model, which is compliant with the...
existing international standards (i.e. STEP and STEP-NC), is utilised for data exchange. The model represents the input and output information used in a lifecycle activity, which is divided into Begin-Of-Life, Middle-Of-Life and End-Of-Life. To support the concept, a Ubiquitous Product Life Cycle Support system has also been developed [43].

In 2010, a system named INFELT STEP was proposed to enable integration of CAD/CAM/CNC operations based on STEP data models [44]. In this three-layer system (Figure 2.6), different sections are defined in each layer to provide interfaces between different CAD, CAPP, CAM and CNC software packages. INFELT STEP has the capability of enabling collaboration with different enterprise-wide CAD/CAPP/CAM/CNC systems using multiple APs of the STEP standard.

---

**Figure 2.5 Major Element Technologies of UbiDM [42]**

In 2010, a system named INFELT STEP was proposed to enable integration of CAD/CAM/CNC operations based on STEP data models [44]. In this three-layer system (Figure 2.6), different sections are defined in each layer to provide interfaces between different CAD, CAPP, CAM and CNC software packages. INFELT STEP has the capability of enabling collaboration with different enterprise-wide CAD/CAPP/CAM/CNC systems using multiple APs of the STEP standard.
Rosen [45] proposed an information system based on STEP (Figure 2.7). Sharing information over time and across the lifecycle in a virtual enterprise is a task that involves more effort than the task of exchanging product data of the same type (e.g. design data, CAD files) between two companies. ISO 10303 application protocols have been proved to be suitable for use in an enterprise product data management environment. ISO 10303 is utilised to transform data in different layers of a software system, the data layer, the business object and services layer and the user interfaces that are similar to the resource layer, global service layer, virtual service layer and application layer in the CManufacturing environment. It has been shown that STEP is capable of being a basis for implementing and sharing product databases and archiving across different levels.
In the past few years, many companies have implemented PDM systems that focus on cost-cutting and shortening the product development cycle. To provide a solution via a common method of sharing standard product and design information, a STEP-compliant PDM system was developed to fulfil the demand for logically integrated product data that is stored physically in a distributed environment [46]. In this system, a STEP-based PDM schema was defined in the XML format to support the Web service connecting PDM systems of several partners through an open network accessible via the Internet. Also using XML, Makris et al. [47] developed an approach providing data-exchanging mechanisms, whereby the Web is utilised as a communication layer. Combining the STEP concept with XML, this work supports the integration of decentralised business partners and enables the information flow within the value-added chain [48].

To recap, the data-centric approach enables total compatibility and communication across the design-manufacturing chain. It is easy to maintain data integrity and realise high-level synchronisation. Since data consistency is achieved at the data level, it is easy to consolidate all the data of a product. The need for coordination or supervision can be minimised. It has been proved that the standardised neutral data method can strengthen interoperability along
the entire CAD/CAM/CNC chain. The approaches mentioned above can support IaaS implementation. Operational processes can be offered as plug-and-play features in the MCloud, with strengthened capability, functionality and availability. However, drawbacks have been observed in this approach. Deployment of one single type of proprietary data format may limit a user’s choice of service application, which is against the true spirit of Cloud. In a Cloud-based manufacturing environment, the user can be provided with a wide range of services for solutions in multiple scales.

2.3. Process-centric Interoperable Manufacturing Approach

In the CAD/CAM/CNC chain, systems may be integrated and consolidated at the process level. This process-centric approach utilises the data flow in a process as a common thread to “string” various data entities together. In this way, no common data format is required. There are four dominant types of approaches: direct data translation, two-way data conversion, dual kernel solutions and service-oriented solutions.

2.3.1. Direct Data Translation

Direct data translation is between paired existing software (Figure 2.8). For example, Teamcenter® [49] is utilised by General Dynamics Land Systems (GDLS) in the U.S. to manage data exchange among systems such as NX, Creo™ as well as Siemens PLM components, e.g. PLM Vis, Parasolid® and NX™ Nastran® SDK. A CAD conversion engine called CrossCAD [50] provides not only stand-alone translators and plug-ins for CAD systems, but also components that can be integrated by software companies. Additional tools (i.e. importing, analysing, healing and exporting models) are available in the toolkit as well.
In 2009, Choi et al. [51] proposed a framework for virtual engineering. The system is described as a Middleware for Exchanging Machinery and Product Data in Highly Immersive Systems (MEMPHIS). As part of MEMPHIS, a lightweight CAE middleware for CAE data exchange is developed for the purpose of exchanging data relying in heterogeneous CAE systems [52]. A generic CAE kernel is designed to store analysis data from various CAE systems and translate it into different formats. In a CManufacturing environment, the quality of data translation is particularly important because of the heterogeneous CAx applications. Since Cloud applications may not understand each other naturally, it is necessary to provide reliable translation functions between these applications.

2.3.2. Two-way Data Conversion

In this approach, data conversion is through a common data format (Figure 2.9). For instance, this common data format can be a proprietary data format or a neutral data format, e.g. STEP or IGES. Using its internal conversion modules, CATIA® can insert, assemble and modify design models of a non-CATIA format. Automatic updates in the context of non-CATIA data are also possible.
As another commercial example, a multi-CAD solution developed by Theorem utilises Generic Collaboration Objects (GCO) [53], which are used to represent and hold all forms of data. Besides direct CAD conversion and visualisation, Theorem also provides STEP processors for AP203 [54] and AP 214 [55].

Targeting interoperability solutions for digital design and PLM markets, Elysium’s [56] portal-based multi-CAD systems provide four major functionalities with its four sub-modules:

- CADporter™: CAD-to-CAD translation and exchange of parts, assemblies, and Product Manufacturing Information (PMI)
- CADdoctor™: geometry checking, healing and verification as well as geometry simplification for CAE, CAM and PLM
- CADfeature™: feature-based design exchange for re-mastering CAD and legacy CAD files and process control
- Integrated Systems: integrated solutions for data translation and exchange, design data management and supply chain integration.

Based on the technology called Universal Product Representation, Proficiency’s [57] feature-based translation solution enables the transfer of complete design intelligence between major CAD systems, e.g. geometry, features, sketches, manufacturing information, metadata, assembly information and drawings in the conversion process. 3D_Evolution® provides a
similar conversion engine [58]. It offers feature-based translation for popular systems such as CATIA®, NX®, Creo®, I-DEAS, SolidWorks, Robcad®, JT® and STEP.

2.3.3. Multiple Kernel Solutions

Two-way data conversion may also take place in a system in which multiple different modelling kernels co-exist, even though such conversions are hardly noticeable to the user. Visionary Design Systems’ IronCAD® [59], formerly an ACIS®-only system, now has incorporated Parasolid® into the system to become the first dual-kernel CAD system. Additionally, CAXA® [60] also utilises both ACIS® and Parasolid® kernels. At the implementation stage of CManufacturing, multiple-kernel application may be a good option since it provides a reliable performance without the risks of data conversion. However, it is not practical to implement multiple kernels in all CAx applications. Software vendors are unwilling to share their technology and embed their kernels into other’s system.

2.3.4. The Service-oriented Approach

The SOA is based on the principle of well-defined functionalities that can be reused for different purposes. Usage of SOA can aid collaborative manufacturing services and resource sharing [61]. It provides distributed architecture focusing on interoperability, easy integration, simplicity, extensibility and secure access, which echoes the CManufacturing concept. SOA reflects a new way of thinking about processes, resources and their information counterparts – the service-oriented agents reinforce the value of commoditisation, reuse, semantics and information, and create business value. CManufacturing can be regarded as a functionality-sharing pool with high reusability and operability that promotes the SOA system to a higher level.

Applications in an information integration environment can be organised in a service-oriented way. Brecher et al. [62] proposed a module-based, configurable platform for interoperable CAD-CAM-CNC planning. The goal is to combat the problems of software inhomogeneity along the CAD-CAM-NC chain. The approach is called open Computer-Based Manufacturing (openCBM) in support of co-operative process planning (Figure 2.10). To implement the architecture and integrate the inspection task into a sequence of machining operations, the STEP standard is utilised to preserve the results of the manufacturing process and feed them back to the planning process. The openCBM platform is organised through a SOA mode, providing the abstractions and tools to model the information and connect the
models. Applications are not realised as monolithic programs, but as a set of services that are loosely linked to each other, which guarantees the modularity and reusability of a system. From the CMaufacturing perspective, openCBM is much like the PaaS concept, in that the system is delivered to the end user directly at the Application Layer. The service provider can deployed at the Manufacturing Resource Layer. The service is ready to be organised at the Global Service Layer.

![Figure 2.10 Module Users and Providers of the openCBM Approach][62]

To achieve a run-time configurable integration environment for engineering simulations, van der Velde [63] reported a plug-and-play framework for the construction of modular simulation software. In this framework (Figure 2.11), the user (at the Application Layer as in a CMaufacturing system) is allowed to select a target of simulation and assign the performer of the simulation, called “component”, before running the selected components. These components are effectively software entities (otherwise known as SaaS, as in CComputing/CMaufacturing). They are modularised, self-contained, mobile and pluggable. After the simulation, the output is post-processed through the components. In such architecture, software modules are detected, loaded and used at run-time with the framework (i.e. the Global Service Layer) needing no prior knowledge of the type and availability of components, thus providing true plug-and-play capabilities.
Lee et al. [64] proposed a Web-based neutral data format amongst heterogeneous simulation software types. The data model named NEutral SImulation Schema (NESIS) defines and categorises product elements at different levels, and clearly describes product, process, sim_list (multiple simulation versions) and configuration information. In the four-layer architecture, NESIS acts as the central internal data structure of the system. At the Client Layer, interfaces have been developed to enable collaboration of the commercial simulation applications and NESIS (Figure 2.12). Developed using Java programming language, these interfaces automatically generate simulation models using simulation information and relevant data received from NESIS, and conversely send simulation information and relevant data generated by commercial simulation applications. Thanks to the interfaces and natural data format, communications between various software applications can be realised, and the reusability of a simulation data model can be achieved. Thus, commercial simulation tools encapsulated with interfaces can work collaboratively in a distributed environment, which provides a function similar to SaaS.
Li et al. [65] introduced a four-layer application service integration platform that is able to bridge multiple Clouds with intra-Information Systems (IS). Interactions across organisation boundaries are supported by a collaboration point, which acts as an interface providing data exchange, command transfer, monitoring and so forth. Implemented in a small metal product manufacturer, the system integrated manufacturing business processes with the help of collaboration agents. This research work examines the possibility of integrating existing manufacturing applications in the Cloud environment.

### 2.3.5. Discussion and Summary

The process-centric approach respects existing vendor-specific software suites in that each activity in a business process can use its native data format. It supports the “Best of the Best” concept, which provides the most excellent or optimized solution among all the feasible options. The data confidentiality problem is alleviated. However, a number of data translators (e.g. plug-ins and convertors) need to be developed and deployed. Data modification may lead to difficult and complex updates. According to the calculation done by Parasolid's business development manager, approximately 20 percent of the models imported from a different kernel still contain errors that need manual fixing, not to mention data loss during conversions between different software applications [66].
Table 2.1 summarises different methods in terms of user-friendliness, practicality and efficiency. Regardless of different approaches (data-centric or process-centric), data translation always takes place, albeit to varying degrees. Many software packages in a modern manufacturing enterprise are usually provided by multiple vendors, using different development toolkits and programming languages. It is obvious that using software with the same kernel is not a practical solution. Use of common file formats (e.g. IGES and STEP) is still the most commonly used approach. The problem, however, lies in the common data formats themselves, which are often incapable of capturing the complete package of the data being dealt with. For example, current STEP (AP 203) files do not yet fully support tolerance data. Applications that operate on multiple kernels perform well, but limitations still exist. Use of a direct data translator appears to be more user-friendly, although its practicality and efficiency level is low, and requires further research. Methodologies that can consolidate heterogeneous data syntax in the Cloud are also needed. CManufacturing can benefit from the abovementioned existing technologies and enterprise systems, however.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Data Translation Method</th>
<th>User Friendly</th>
<th>Practical</th>
<th>Efficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-Centric</td>
<td>Use of software with the same kernels</td>
<td>****</td>
<td>*</td>
<td>****</td>
</tr>
<tr>
<td></td>
<td>Use of common file formats</td>
<td>****</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Process-Centric</td>
<td>Applications that operate on dual kernels</td>
<td>**</td>
<td>**</td>
<td>****</td>
</tr>
<tr>
<td></td>
<td>Use of a direct data translator</td>
<td>***</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>

Note: Number of asterisks ‘*’ denotes the degree to which an approach is rated for a criterion.

2.4. Cloud Manufacturing Approaches

As mentioned above, CManufacturing is a new SOA network that can be considered as an advanced process-centric manufacturing model. It provides a mechanism of multi-objective resource access that is particularly suitable for geographically separate and organisationally diverse business models. Integration and interoperability of manufacturing applications can be achieved at the service level.
2.4.1. Cloud Manufacturing

Li et al. [67] proposed a service-oriented networked manufacturing model as a Cloud Manufacturing model. In this research, a number of relevant technologies are discussed. Intelligent agent, product lifecycle management, resource modelling and evaluating technologies are identified potential support for Cloud architecture. However, there is no method that integrates various manufacturing processes in the Cloud. The model was further discussed by Zhang et al. in which a dynamic Cloud Service (CService) centre in Cloud Manufacturing was defined as MCloud [68]. OWL is used to describe the Cloud resources and support service management. It is necessary to extend existing OWL models into the manufacturing context, to meet a dynamic industry environment (Figure 2.13).

Tao et al. [69] proposed a framework of CMfg Manufacturing and discussed some of the key advantages and challenges for future CMfg Manufacturing systems. It is predicted that a CMfg Manufacturing system can reduce the cost of resources, and increase their utilisation. The relationship between CCmputing and CMfg Manufacturing is discussed. A seven-layer running model was proposed to support and provide a public manufacturing platform (Figure 2.14) [70]. However, feedback during operations cannot be utilised in the proposed system, which
means bidirectional communications between operational layers and service providers are not supported.

Figure 2.14 Architecture of Cloud Manufacturing Running Model [70]

In 2010, a Cloud-based manufacturing research project was launched, sponsored by the European Commission [71]. The goal of this project (named ManuCloud) is to provide users with the ability to utilise the manufacturing capabilities of configurable and virtual production networks, supported by a set of SaaS applications (Figure 2.15). In the proposed system, customised production of technologically complex products is enabled by dynamically configuring a manufacturing supply chain [72, 73]. It is considered that the development of a front-end system with a Cloud-based manufacturing infrastructure is able to better support the specifications and on-demand manufacture of customised products. Based on the conceptual architecture, two main types of users who interact with the front-end system are identified: the manufacturing service consumer (e.g. a product designer) and the
manufacturing service provider (e.g. a lighting product manufacturer). Compared with a service consumer, more interaction is required between a service provider and the MCloud. Nevertheless, there is still a lack of ability to support the activity provider. In this Research work, Manufacturing as a Service (MaaS) was proposed to achieve configurable and customised manufacturing [74]. Manufacturing Description Service Language (MDSL) was developed to model and represent different types of product characteristics, for example, shape, size, mechanical, electrical, etc. However, it is envisaged that this language may have difficulty in integrating with existing CAD models because of the different data syntaxes. Furthermore, data duplication exists between MSDL documents and 2D/3D product images. Synchronisation between these files is also a challenging task due to different data syntaxes being used.

In 2011, Houshmand and Valali [75] proposed a collaborative environment named LAYerd MODular platform (LAYMOD) to realize CAx collaboration and product data integration. LAYMOD aims to overcome the shortcomings of the international standards (ISO10303), which is the cost and time of APs implementation. It is also able to eliminate duplication and repeated documentation of the same data entries in different APs. Very recently, LAYMOD was extended to XMLAYMOD by adopting an XML data structure and CComputing paradigm (Figure 2.16) [76]. An XML Service Cloud and its database are inserted between

![Diagram](image-url)
the Interface Layer and Modular Interpretation Layer, unlike the original system. XML Service Cloud plays a central role in XMLAYMOD, which translates product data from an original format into XML, or vice versa. The new system benefits from the portability of the XML data structure. However, the advantages of Cloud are not fully utilised except accessing in remote data.

![XMLAYMOD Structure and Layers](image)

**Figure 2.16 XMLAYMOD Structure and Layers [76]**
2.4.2. Resource Management in CManufacturing

To facilitate a CManufacturing environment, existing resources need to be scaled, modelled and integrated into an MCloud. Wu and Yang [77] defined CManufacturing as an integrated supportive environment for both sharing and integration of resources in an enterprise. A resource sharing method is also proposed to describe and scale the manufacturing resource in a Cloud. Manufacturing resources are divided into four layers: a manufacturing resources layer, concrete web service layer, logical service layer and application layer (Figure 2.17) [78]. However, the method does not provide the data model with domain-specific data for different manufacturing processes.

Cheng et al. [79] studied a utility model and utility equilibrium of resource service transaction in CManufacturing. Decision-making methods have been developed to maximize the utility of a resource demander and resource provider. Yet it is difficult to represent the manufacturing resources without a standardised schema, not to mention that original user requests would be difficult to fulfil.

Hu et al. [80] analysed the factors which influence classification of virtual resources in CManufacturing. Examples are introduced to validate the effect of these factors on task assignment. Luo et al. [81] discussed the CManufacturing system from the viewpoint of network, function and running. A multi-dimensional information model was proposed to describe manufacturing abilities [82]. This knowledge-based data model is helpful in providing the user with a manufacturing service via network.
Flexibility of the Cloud has been classified into task flexibility, resource service flexibility, Quality of Service (QoS) flexibility, correlation flexibility and flow flexibility [83]. Selection of the composition of a resource service can be achieved through quantitative evaluation of flexibility. To control and manage the flexibility of the service composition in CManufacturing, Zhang et al. [84] proposed architecture that considers various major dynamic-change factors in the life-cycle of a resource service. A monitoring module is suggested to connect manufacturing resources and users, for flexible remote management. A multi-agent technology is reported to be an effective tool for solving problems through sharing knowledge in CManufacturing [85]. It has been pointed out that an agent-based mechanism provides flexible and effective sharing and utilisation of elastic resources. A Cloud-based structure has been proven capable of supporting the whole supply chain, building up business relations and, finally, better matching the demand-and-supply capacity data [86].

The challenge in resource integration comes after resource reorganisation. Fan et al. [87] proposed an integrated architecture of CManufacturing based on a federation mode. Federation integration rules are applied before resources are connected to the system. Thus, connecting or releasing a resource will not influence the operation of the whole Cloud environment. To maintain CManufacturing resources, an Optimal Allocation of Computing Resources (OACR) system is proposed [88]. In OACR, an improved Niche immune algorithm is introduced to solve the resource scheduling problem in grid systems or CComputing systems associated with the Niche strategy. However, user interactions are fully involved during service execution, which may work against the spirit of Cloud. In a CManufacturing system, CUs hire the manufacturing service, instead of directly investing in devices and machines. CService should be well-defined in a project requirement, and CU interaction minimised, particularly after the requirement is submitted.

Li et al. [89] discussed the resource access issue in a CManufacturing environment. As an on-demand and multi-objective networked manufacturing model, a CManufacturing system should be able to provide multi-scale and on-demand service capabilities through a variety of loose coupling service applications. In this research, a multi-target, backward-authority covering graph and a migration authority path-confirming algorithm was proposed. Manufacturing processes can be scheduled based on capacity constraints, with minimum
calculation time required. Combined with pre-defined rules, e.g. limit of size, material, price and time, the CService can be optimized to fulfil the demand of user.

An ontology-based method was proposed by Zhang and Zhong to profile a CManufacturing resource [90]. A peer-to-peer registry network was suggested for building the CManufacturing database, though peer-to-peer infrastructure faces portability and synchronisation issues due to lack of standardised data structures and coordinating methods.

2.4.3. Design, Simulation and Optimisation in CManufacturing

The realisation of CManufacturing requires implementation of each manufacturing application in the MCloud. In a very recent ASME (American Society of Mechanical Engineers) web seminar, it was specifically pointed out that Cloud providers can support manufacturing applications with cheaper and more scalable access [91]. Actually, the importance of the Cloud approach has already been recognised by some industry. NVIDIA, one of the largest Graphics Processing Units (GPU) providers, has developed a Cloud-based GPU that is capable of delivering virtualised workstation performance and capabilities to desktops [92]. It is predicted that Cloud-based CAD systems will be available in a very near future [93, 94]. In fact, Autodesk can already provide Cloud-based applications to support collaborative design. Via the environment called Autodesk 360, users are able to access a number of simulations such as mechanical simulation for linear and nonlinear static stress and dynamics, computational fluid dynamics analysis for fluid flow and thermal simulations, rendering, design optimisation, energy analysis, and structural analysis [95].

From the cooperation perspective, Tai and Xu [96] discussed the complexity of collaboration in a CManufacturing environment. Cooperation is classified into four types, i.e. random, operational, tactical and strategic. In a matured CManufacturing environment, the system is required to process all types of collaborative interactions at multiple levels, including typical manufacturing applications such as simulation and optimisation.

Chai et al. [97] established a High-Performance Cloud Simulation Platform for efficient execution and problem solving for complex simulation tasks. As a specific type of manufacturing application, a simulation task is akin to a CComputing service. Computing performance can be improved by the collaboration of computing nodes in CComputing approaches.
To facilitate a Cloud-based on-line optimisation system, Chandrasekaran et al. [98] developed a system named CComputing-based optimisation (CCBOD) (Figure 2.18). In CCBOD, the main server keeps the repository of data and carries out optimisation. Its data input and retrieval mechanisms were developed to cooperate with an online optimisation process. However, the method of promoting an optimisation process in the Cloud is not as clear in the report and this system is not capable of identifying user/data in various domains.

![Figure 2.18 Structure of Cloud Computing-based Optimisation](image)

**Figure 2.18 Structure of Cloud Computing-based Optimisation [98]**

### 2.5. Cloud Computing Applications in Manufacturing

Much of the recent literature discusses the architecture, deployment method, characteristics and implementation of CComputing systems. In this subsection, the latest CComputing technologies are discussed from a manufacturing perspective.

#### 2.5.1. Public and Private Clouds

Public Clouds provide a self-service basis over the Internet. Their applications are organised by a third-party provider that shares resources in the Cloud. It is a typical pay-as-you-go model for selling the computing utility as a service. Private Clouds are used in-house for an organisation. It is similar to traditional IT solutions, but it is enhanced by the adoption of a resource-pooling approach that enables dynamic provision.

In late 2011, IBM released its SmartCloud strategic solutions. Based on the same reliable technology as the public Cloud offerings, IBM has developed a comprehensive product portfolio for building a business’s own Private or Hybrid Cloud. As illustrated in Figure 2.19,
SmartCloud Foundation features a range of Cloud architecture components including physical infrastructure, virtual infrastructure, management, middleware, integration, security and consulting services. These components offer unparalleled time-to-market, integration and management capabilities [99]. Zhang and Zhou [100] discussed bridging the power of SOA and virtualisation in the context of a CComputing ecosystem with architectural principles, e.g. integrated management for Cloud, SOA for service reuse and unified data representation. The above concept methodology of SOA is the foundation of IBM SmartCloud Enterprise-IBM Smart Business Cloud solutions for New Zealand [101].

Figure 2.19 IBM SmartCloud [101]

In the early 2012, paper CComputing Best Practices, which is published by Microsoft Cloud 2.0 White paper, extends smart integration to social business architecture [102]. Cloud providers will have to open their software platform with added customised features. In terms of manufacturing, the adopt of the Cloud 2.0 pushes the industry to not only integrate enterprises’ internal services but also provide an open platform to collect various costumers’ individual requirements, and then integrate them into real-time requirements. Table 2.2 lists some major Cloud providers on the market.
Table 2.2 Major Cloud Service Providers

<table>
<thead>
<tr>
<th>Provider</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpSource</td>
<td>Cloud servers and Cloud files, and Cloud network</td>
</tr>
<tr>
<td>Rackspace</td>
<td>Cloud servers, Clout files and Cloud sites</td>
</tr>
<tr>
<td>IBM</td>
<td>IBM SmartCloud Enterprise</td>
</tr>
<tr>
<td>OneNet</td>
<td>New Zealand Cloud provider</td>
</tr>
<tr>
<td>Amazon</td>
<td>Amazon Web Services (AWS), SAP and EC2</td>
</tr>
<tr>
<td>Google</td>
<td>Google Apps</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Biztalk Server</td>
</tr>
<tr>
<td>IBM</td>
<td>IBM SmartCloud Foundation products</td>
</tr>
<tr>
<td>BitCloud</td>
<td>BitCloud Private Cloud</td>
</tr>
</tbody>
</table>

After adding the Cloud elements, traditional smart-client infrastructure can be upgraded to smart-client Cloud or Cloud-client architecture. A Client + Cloud seamless architecture was evaluated by Grochow et al. [103]. The enhancement is identified including interaction, seamlessness and fast Cloud processing in 2010. Kyung Hee University developed a multi-platform mobile architecture in the Cloud environment [104]. In their research, a new market trend with a rich interactive experience was introduced, which can be used in manufacturing to create individual markets. Thin client architecture was proposed to support authentication and management during Cloud deployment.

Interest in Cloud is constantly growing in the industry. The interest in Cloud is gaining attention in industry. Osterman Research [105] completed a survey, which found that the increased enterprise security protection in Cloud-client infrastructure, low cost, improved performance, ability to frequently update, and fast response, are among the most desired features. A StratusLab Case started in June 2010 to compare the cost of public Cloud with private Cloud [106]. The result shows that deploying Amazon Cloud on a pay-as-you-go basis is more cost-effective than investment in a 20-computer IT structure with an average utilisation of lower than 70 percent. Deployment from large-scale Cloud providers such as Amazon improves the level of utilisation. Indeed, the larger the utilisation (more VMs running and thus fewer CPU cycles wasted), the higher the return on investment becomes. In a Cloud-based approach, the data flow is established in the Cloud with minimum network
overhead since the digital product is transferred over the network. In particular, the Cloud typically does not provide advanced graphics support for fast visualisation. At the present stage, this problem can only be alleviated by data compression or using an optical fibre cable network.

Computing is rapidly moving from early adopters to mainstream organisations. It partially reduces the duties of IT departments, for example outsourced software upgrades and hardware maintenance. Cloud is a one-stop shop that does not rely on anyone else, and the service is scheduled for the moment that it is required. From early 2012, One-Stop-Shop added on Cloud has been discussed by Dell [107], HP [108], and even tested by Amazon [109]. Pre-configured service and applications could easily be added to an existing service solution, to fulfil the demand and expand the scope and ability of an enterprise.

2.5.2. Hybrid Cloud

Community Cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organisations that have shared concerns. To meet the need for greater flexibility, scalability, cost benefits, capacity and agility in an enterprise, it is necessary to compose two or more distinct Cloud infrastructures (private, public, or community), thus forming a Hybrid Cloud structure. In practice, Hybrid Cloud is normally developed in terms of enterprise Cloud, which contains the features of private Cloud and community Cloud. Multiple Clouds are bound together by standardised or proprietary technology that enables data and application portability. It reduce IT complexity and cost for the enterprise user. The major enterprise solutions are listed below:

- Amazon [7] Elastic Compute Cloud (EC2) provides a virtual environment to run industrial applications. With the help of a pre-configured, templated Amazon Machine Image (AMI) which contains a user’s applications, libraries, data and associated configuration settings, the consumer is able to start, terminate and monitor EC2 instances within AMI, using the web service APIs or a variety of management tools provided.

- Eucalyptus [110] is a Cloud platform offering private CService. As an open-source implementation, it provides IaaS Cloud, which is compatible with applications under Amazon Web Services™ (AWS) API, such as AMI, Amazon Elastic Compute Cloud, and Amazon simple storage service.
Nimbus [111] Platform is an integrated set of tools that delivers infrastructure Clouds to scientific users. As another platform that is compliant with Amazon EC2, it enables users to release remote resources and build a computing environment by deploying virtual machines.

Google [8] App Engine enables the user to develop web-based applications using Python, Java and Go programming language. It provides a rich set of APIs for searching, large data storage, high-speed data access and email applications.

Microsoft [9] Azure is designed to create, host, manage and scale both web and non-web applications including website, virtual machines, mobile, CService, large data and media stream. Multiple industrial standards, APIs and programming languages are supported for development and maintenance. It is proven that Azure is capable of processing visualisation, simulation and complex modelling applications that are similar to domain-specific tools in manufacturing [103].

Abiquo [112] enterprise Cloud provides functions of server controlling, storage and network resources. An industry standard is deployed to enable an enterprise to deploy and manage Cloud resources.

Aneka is a service-oriented Cloud platform that supports multiple application models, persistence and security solutions and communication protocols in private and public network environments [113].

OpenNebula [114] is an open-source platform for building and managing virtualised enterprise data centres and Cloud infrastructures. Customised enterprise data centres and private Cloud infrastructures can be developed, and users are able to deploy and maintain these virtual applications on physical resources.

Apache CloudStack [115] is open source software designed to deploy and manage large networks of virtual machines, such as IaaS Computing platform. It is utilized by service providers to offer Public, Private Cloud, or as part of Hybrid Cloud infrastructures. Cooperated with CloudStack, Apache Hadoop [116] software library is a framework that allows for the distributed processing of large data sets across clusters of computers using unified programming models. It is designed to detect and handle failures at the application layer.

OpenStack [117] is a global project producing open source Computing platforms. The project aims to deliver solutions for multiple types of Clouds by being simple to implement with scalability and rich features. The technology consists of a series of
projects delivering components offer a Cloud infrastructure e.g. OpenStack Compute, Object Storage, Image Service, Identity, Dashboard and so forth.

Due to its fast, flexible and affordable characteristics, Enterprise Resource Planning (ERP) systems were suggested to achieve a higher level, resilient virtual infrastructure [118]. As an SaaS application, Cloud-based ERP can improve the performance of manufacturing, human resource, supply chain management, and project and data management. Motalab and Shohag [119] proposed a Cloud-based ERP system to support automatic business activities. It is stated that combining Cloud with ERP can reduce the cost of implementation, support, licensing and maintenance in comparison with traditional methods.

When using the existing Cloud infrastructure, particularly the various enterprise Cloud infrastructures, it is necessary to build a collaborative connection network above these Clouds. Villegas [120] proposed a layered service model to establish federation at multiple levels, i.e. SaaS, PaaS and IaaS layers (Figure 2.20). Different combinations of resources are allocated according to different types of execution parameters, constraints (time, cost, quality limits) and resources. An elastic federation mechanism is particularly suitable for contractors who supply equivalent final parts or services to the primary provider, facilitating elasticity to support a dynamic market.

From the viewpoint of human resources, Computing is capable of improving the training system, for example education in engineering expertise [121]. The Cloud structure allows unfettered access, which is particularly suitable for ever-changing training participants. Customised and elastic services offer no additional risk for time-distributed attendees, while specific requirements and budgetary allocations can be met in a timely fashion.
2.5.3. Cloud Technology and SMEs

Computing is more suited to small-to-medium size manufacturers because it offers significant cost-saving and flexibility. SMEs show a strong expectation of deploying Cloud. According to Microsoft’s study [122], 39 percent of small businesses expect to be paying for at least one Computing service within the next three years. That is an increase from the 29 percent who are currently paying for Services. In practice, SMEs are frequently troubled by IT solutions because of the risk of fire, power loss and low IT skills. According to Gartner’s research, SMEs should be quicker to adopt Computing as a means of creating, managing, and reusing product and process content, than larger manufacturers [123].

Computing is identified by Mezgár as an important technology since it offers significant financial advantages and high-level collaboration possibilities, particularly for networked companies [124]. Private Cloud can be the best ICT solution for manufacturing enterprises. It supports supply chain visibility, transportation management and supplier/contract negotiation. As a good start, a private Cloud is a suitable platform for collaboration and supporting business processes, particularly for geographically-dispersed teams, who travel frequently and are responsible for activities in the field, in factories and with partner firms.

Being available without substantial investment, Computing provides a better solution than most SMEs can realise with their modest investment levels [125]. Computing needs no upfront investment, which gives cash-strapped SMEs more flexibility with their capital. For example, when they try a new ERP system, the traditional alternatives would be too expensive initially. Historically high-priced tools can become affordable, thanks to the elastic pricing of the Cloud.

Moreover, SMEs are not the only group benefiting from Computing. Large enterprise architecture is often complex, and consists of different technologies, techniques and tools that make up the enterprise computing environments in which various applications are delivered and used as explained by Chen et al [126]. This heterogeneous environment demands process and data interoperability. Cloud infrastructure offers a large-scale solution on a higher level, compared with the historical attempts. With the help of control and management mechanism, integration and communication risks can be efficiently predicted and controlled ahead of service execution.
2.6. Manufacturing Technologies supporting Cloud Manufacturing

CManufacturing is still a new concept. Yet studies of distributed manufacturing, collaborative manufacturing and virtual enterprise have given rise to new technologies that are capable of supporting CManufacturing. The rest of this section discusses some of these research works.

2.6.1. Approaches to Achieving Product Information Integration

There is a need for developing methodologies for data management during software and database integration [127]. In particular, in a Cloud environment, it is necessary to build catalogues of products, resources and services that can be replicated or shared throughout the Cloud. From the service point of view, Cloud resources and services need to be treated as a way to digitalise and standardise manufacturing abilities.

To work on multiple versions and views of a shared model, Sadeghi et al. [128] proposed a collaborative architecture to allow experts to share and exchange design information. In this architecture, product design is exchanged through a standardised, constraint-based model to maintain complex relationships in multi-disciplinary collaborative designs. Thanks to this data model, conflicts happening during the synchronisation process can be described and resolved via a notification mechanism.

To achieve an effective product data-sharing environment, Do and Chae [129] developed a product data management architecture for supporting collaborative product design. In this architecture, an additional data model was proposed as an extension linked to the STEP standard. With the help of this system, different configurations or modifications made by various engineers can be brought together. Hardware engineers and software programmers will be able to share the same user environment, on a consistent database during the process of collaborative product development. In another piece of research into Engineering Change (EC), Hwang et al. [130] proposed a data model representing and propagating EC information. In this collaborative product development environment, a neutral reference model was developed, based on the STEP data structure. EC that consists of collaborating companies can be applied and reflected in the product design. Within the reference model, a neutral skeleton model and an external reference model were developed to support a widely-distributed collaborative design environment.
Apart from the design applications, research was also carried out to integrate the whole CAD/CAM/CNC chain. To facilitate a Web-based design-manufacturing environment, Alvares et al. [131] proposed a Web-based system using a data structure similar to that of an international standard. In this system, files in neutral formats are passed along a serial software chain that consists of WebCADFeatures, WebCAPP and WebTurning applications. In a heterogeneous environment, data exchange is a challenging issue when proprietary software tools are integrated within the same architecture. Oh and Yee [132] presented a method for semantically mapping different business documents to a unified document format, given the inevitable existence of multiple product representations. In this research, the XML format was adopted to support Web-based applications and a model through the Internet.

In the middleware of a CAE system developed by Song et al. [133], a structure of the proprietary file format was proposed to interface with multiple CAE software tools. Using VRML, heterogeneous CAE data was translated into “chunks” described by entity-attribute data structures that are similar to STEP structures. It was proven to be effective at deriving data in a chunk-type form. Even if new entities are subsequently added, the structure is able to be read in a supplementary form, in spite of the addition; these entities are described as new chunks. To integrate heterogeneous business organisations, a Collaboration Point (CP) concept was proposed by Li et al. [65]. CP is located on the boundary of different organisations, acting as the interface for processes to be interoperated across various organisations. The operational processes of the enterprise and CServices can be described by business process models of CP. Also, the common activities of two kinds of processes are identified. CPs were introduced and connected to common activities such as the modelling interpretation of inter-operation. This interface can support data exchange, command transference, monitoring and more.

In addition to the approaches mentioned above, more methods have been developed to strengthen interoperability along the STEP/STEP-NC-based CAD/CAM/CNC chain. For instance, Vichare et al. [134] developed data models to describe all the elements of a CNC machine tool. In this approach, machine specific data is defined in the form of a STEP-compliant schema. This data model acts as a complementary part of the STEP-NC standard to represent various machine tools in a standardised form.

Choi et al. [135] defined a standard data format using XML for a neutral file containing Product, Process and Resource (PPR) information, named PPRX (PPR eXchange). The
information model, mapped from ISO 10303-214 STEP models, supports PPR information exchanges between commercial heterogeneous PLM systems and other systems. With the XML-based data exchange methodology, information exchange can be made without loss, which reduces unnecessary effort and supports effective integration and information sharing.

As an example of a specific application protocol of STEP, Jardim-Goncalves et al. [136] proposed a knowledge framework called funSTEP, which provides enterprise and manufacturing systems with semantically seamless communication with other stakeholders up and down the supply chain. Based on STEP AP236 standard [137], semantically enriched international product data standards and knowledge representation elements are utilised as a basis for achieving seamless enterprise interoperability.

In order to speed up a specific task, a web service architecture called Web Service for CAD was proposed by Kim et al. [138]. It can support the collaborative product development of CAD assembly and part data. XML product models have been developed based on multiple STEP APs. Using these models, parallel processing can be deployed to make an assigned task run faster because more than one processor can be used to run the tasks. The result of the experiment demonstrates that it is possible to partially retrieve product data, and improve computing performance by processing these data subsets.

In order to display a specific range of data, ST-Developer™ [139] enables users to view allocated types of entities defined in a STEP Part 21 file. By using the functionality “working set” embedded in the STEP file browser, the user is able to exclude/include assigned types of entity, e.g. showing all the machining working steps in the document only, or hiding all the material information in a project. It demonstrates the capability of the object-oriented STEP data structure in the context of processing data subsets.

When a product file is shared and exchanged in the collaborative environment, the quality of the data itself needs to be considered as well. Kikuchi et al. [140] proposed a Product Data Quality model as a resource model for STEP standards. In this way quality of product data, in particular shape data, can be modelled and stored along with the product document.

In addition to the aforementioned STEP-based solutions, a data structure called Linked Data was proposed by Graube et al. [141]. Through this generic data structure, distributed information spaces from different domains were condensed into an interlinked Cloud with two integration methods. The first is to merge them into a single Linked Data Cloud using
appropriate adapters and converters, and the second is the complete migration of the databases to native Linked Data stores. In this approach the graph theory is utilised, in which it is possible to describe an object-oriented data structure such as STEP.

To recap, standardised data formats (e.g. STEP and STEP-NC) enable a seamless data exchange environment along the CAD/CAM/CNC chain. Despite this, there is a lack of a suitable solution for integrating product information throughout the lifecycle at high levels, and providing user-specific data at the same time. In particular, in the CManufacturing environment, data communication takes place at multiple levels for specific users and needs. Thus there is an emerging need for a widely recognised data sharing methodology capable of fitting in different and specific domains at the same time.

2.6.2. Enterprise Systems

To efficiently interact with current enterprise processes, CManufacturing is able to seek support from Enterprise Systems (ES). Manufacturing systems can be re-evaluated from the Cloud perspective, e.g. IaaS and SaaS.

Besides the related research approaches discussed, ERP and modelling systems are able to contribute to the CManufacturing environment. After manufacturing activities are properly modelled by standardised models, the next step is to integrate these operational processes. To integrate a business into an MCloud, the first step is to understand and model an enterprise.

The ERP system has been studied extensively [142-146], including its inter-organisational performance [147] and behaviour throughout the supply chain with multiple stakeholders [148]. It is believed that SaaS is a promising approach for implementing ERP systems in the Cloud. ERP systems are integrated software packages with a common database that supports business processes in companies [149], which match concept of an enterprise Cloud well. Enterprise information systems should support frequent users such as large companies, and occasionally connected clients with a limited budget, e.g. SMEs [150]. With the help of ERPs, inter-organisation behaviours/reactions can be modelled and mapped in a standardised manner, i.e. neutral APIs. Based on APIs, MCloud can be established via integrating these reactions in standardised semantics, without changing the organisational structure of an enterprise.
Papazoglou and van den Heuvel [151] proposed a framework named Enterprise Service Bus, which is an integration platform that utilises web service standards to support SOA applications within an enterprise. The extended SOA system can be further adopted by CManufacturing, which supports capabilities such as service orchestration, “intelligent” routing, provisioning, integrity and security of message as well as service management. In a SOA system, business procedures can be modelled and componentised to support seamless business integration [152]. Schmidt et al. [153] proposed architecture declaring clear definitions of service capability and requirements in a service-oriented context. Models have been proposed to evaluate the quality/feedback in the business-to-business context [154].

2.7. Recap

CManufacturing is a new manufacturing model that has evolved from SOA, networked manufacturing and CComputing. It bears some resemblance with, albeit has differences from, other approaches such as grid manufacturing, agile manufacturing, networked manufacturing, Application Service Provider (ASP), Computer-Integrated Manufacturing (CIM) and CComputing.

- Grid manufacturing focuses on connecting multiple manufacturing resources using a network. It provides a one-to-many service model, in which distributed resources are organised for one consumer. From the provider’s point of view, the workload of the whole system heavily relies on user demands with less flexibility. When the demand is low, the utility rate is also low. In contrast, CManufacturing provides a many-to-many service model, in which workloads are balanced and delivered to multiple service consumers and service providers. CManufacturing can also deliver a globally optimised solution.

- Agile manufacturing aims to allow an organisation to respond quickly by changing the operational process, resource and products. It provides rapid resources, data and knowledge combination to create new product, with cost and quality under control. In practice, there is no emphasis on resource sharing in agile manufacturing. The range of change is limited in terms of budget, resources and experience. In CManufacturing, manufacturing agility is achieved by a collaborative environment based on Cloud. Production tasks can be accomplished by utilising outsourced resources from MCloud.
Networked manufacturing is mostly about connecting manufacturing applications within one enterprise or organisation via the network. Resources and abilities are offered with common interfaces, protocols and data formats. It delivers a one-to-one service model between a service consumer and a provider. There is little consideration of resource management and sharing. Difficulties may occur when two networked systems cooperate. CManufacturing builds up a virtual supply chain among various service participants centralized by the MCloud. Standardised service packages can be shared in the Cloud that are contributed by multiple manufacturing enterprises and organisations.

ASP's are third-party service firms which deploy, manage and remotely host software applications through centrally-located services with a rental or lease agreement [155]. It offers opportunities for SMEs who have little knowledge or experience of ICT. It implements the concept of software applications outsourcing. However, it is also simply a one-to-one service model between the client and application provider. Software solutions are limited by the knowledge and capability of the provider. CManufacturing establishes the resource-sharing pool in a neutral manner. This means that the service providers are treated equally in the supply network. CUs benefit from the “Best of Best” service, and the weakness of an individual provider is overcome by the partners integrated in the Cloud.

CIM focuses on computerisation of the entire production process. The utilisation of computers allows individual manufacturing processes to exchange information with each other at a faster speed. CAx applications may be considered as subsystems of CIM. Nevertheless, communication and interaction difficulties still exist between different CIM systems from different companies. CManufacturing offers a service-centric perspective to coordinate and execute an operational process throughout the value chain. It provides a collaboration model beyond the boundaries of applications and organisations.

CComputing establishes the fundamental concept of resource pooling and sharing. It mainly focuses on computing objects. In manufacturing, the concept of Cloud needs to be extended to physical resources, e.g. material, machines, sensors and so on. CManufacturing takes care of the operational process throughout the supply chain. It provides a methodology to identify, componentise and integrate manufacturing abilities into service packages.
Manufacturing helps the industry transform its business models, align product innovation with business strategy, and create intelligent factory networks that encourage effective collaboration. From the economic point of view, three aspects can be observed in Manufacturing:

- lower entry requirements of implementation allow an enterprise to start or improve their resource without unaffordable upfront investment
- ability to lease short-term creates the opportunity to outsource an expensive resource in an enterprise, hence improving the utilisation rate (productivity) of facilities
- more manufacturing resources are made available, which provides a wide range of support to meet a fast-changing market.

Manufacturing is more than the implementation of Computing in manufacturing. Manufacturing enterprises and their resource/capability need to be componentised and virtualised for them to be integrated in the MCloud. Manufacturing offers new possibilities for building a virtual value chain without a full portfolio of manufacturing facilities. Thanks to its flexibility, high value-added and customised products can be achieved at a lower cost. It creates an intelligent factory network that encourages effective collaboration, and connects resources with little management effort.

A thorough review of interoperable approaches and Manufacturing models has been presented, along with discussions of related integration methods, e.g. information systems and enterprise systems. Although a number of approaches have been attempted in recent years, there is still a lack of a comprehensive solution to support the entire manufacturing supply chain. Research gaps can be observed from multiple angles. There is no system that can support a complete Manufacturing environment. There is a need to build representations of services that can be replicated and shared throughout the Cloud. Service can be treated as a way to digitalise and standardise manufacturing services. Distributed resources need to be encapsulated into Services and managed in a centralised way. A possible solution may be that to bridge the existing advanced manufacturing models with a Computing technology. Some bottleneck issues include effective models, software integration, data integration, resource virtualisation, service implementation and security.

This research aims to develop an interoperable environment that enables collaborative integration of manufacturing applications. To facilitate this proposal, developing a
consolidated mechanism compromising resource virtualisation, service supervision/execution and data/knowledge management is the first and most important task to accomplish. As discussed, neither a data-centric nor a process-centric interoperable approach will alone provide a feasible solution. Thus, hybrid architecture combining both data-centric and process-centric concepts is proposed via the CManufacturing approach. A module-based, neutral data-centric and service-oriented architecture is suggested for the development of an Interoperable Cloud-based Manufacturing System (ICMS). A detailed description of the CManufacturing philosophy is presented in Chapter 3, along with the ICMS framework.
Chapter 3. System Framework

CManufacturing attempts can be summarised as two sectors, i.e. a manufacturing version of Computing Cloud, and a distributed environment that is networked around Manufacturing Cloud. First and foremost, it is necessary to understand existing manufacturing resources and implement them properly in the Cloud. In this chapter, manufacturing resources, abilities and relevant essentials are discussed from a service-oriented perspective. Major work discussed in this section has been published in a paper [156]. The functional requirements of a CManufacturing environment are discussed, along with an interoperable manufacturing system framework. The system framework presented here has also been published as a book chapter in CManufacturing: Distributed Computing Technologies for Global and Sustainable Manufacturing [157]. The latest development has been submitted to the ASME MSEC2013 conference [158].

3.1. Overview

It is identified in Chapter 2 that there is a need to remodel and integrate the manufacturing industry to achieve full CManufacturing. These days, a manufacturing enterprise would not survive without CAx technologies. Deploying CAx software on the Cloud improves performance in terms of flexibility, extendibility, integrity and easy/unlimited data storage. In recent years, research has been carried out worldwide in an attempt to develop an interoperable and collaborative environment with heterogeneous software applications. According to observations from Forrester, Cloud strongly supports online collaborative work over the Internet [159].

From the provider’s point of view, applications can be easily maintained and utilised on the Cloud server. Version updating, maintenance and integration can be remotely finished, avoiding regular service from on-site maintenance specialists as in the past. Without the costs of freight, on-site setup, transportation and so forth, consumer applications are improved by IT experts changing, updating or diagnosing in the Cloud. Thus provider’s costs are reduced by less management and maintenance effort, with a better quality of service.
As mentioned above, CManufacturing provides more opportunities and avenues for business and the ability to customise products easily. It realises on-demand service virtualisation with multi-objective resource access. CManufacturing delivers service packages to fulfil the CU’s original needs without the effort of setting up a whole virtual value chain. It helps enterprises survive, in a competitive market, with fewer resources and personnel. From a user perspective, the cost of expensive applications is spread over multiple CUs thanks to the pay-as-you-go basis of CService. Costly but rarely used software can be priced according to usage. No upfront investment is needed, which further improves cost-efficiency.

CManufacturing is a much more challenging task than purely deploying manufacturing software in the Cloud, which could be a preliminary stage of developing CManufacturing. Unlike software programs and IT infrastructure, physical equipment cannot be easily deployed through the Cloud. Hence there is a need to understand all intermediate processes from raw material to finished products and then to virtualise them in the Cloud. To implement an advanced CManufacturing environment, a Cloud-based manufacturing system framework is proposed in this chapter.

3.2. Advanced Manufacturing: Moving to the Cloud

As shown in Figure 3.1, a typical manufacturing process is organised in a serial chain, to transform raw materials into finished goods. Quality and profitability are affected by the resources and capability of the suppliers, contractors and sub-contractors. When an enterprise forms its business strategy, choice is limited by the scale of partner network, budget and experience. In particular, for SMEs, unaffordable upfront investments and insufficient resources prevent them from expanding or improving their business. It also limits the agility and flexibility of the enterprise.
In the CManufacturing model, suppliers, manufacturers, expertise and relevant personnel are coordinated in the MCloud via the network. Thus, it forms a shared pool of configurable manufacturing resources. Utilisation of Cloud resources could exist from random short-term contracts to strategic long-term cooperation. From the manufacturer’s point of view, a business network is extended by the broad scope of the Cloud network. In the traditional business sharing approach, e.g. Amazon or ThomasNet Product Search [160], products from multiple suppliers are categorised and published in the platform. These methods focused on “final products” and realise a “product as a service” model. In the CManufacturing paradigm, on the other hand, the service arrangement is focused on the resources and capabilities. Manufacturing ability is strengthened by wide support from the whole Cloud at upper levels, which provides a rich collection of devices, equipment and resources. Additionally, costly and rarely-used in-house facilities can be outsourced to the Cloud. From the viewpoint of customers, CManufacturing provides a wide range of products and production abilities. On-demand service can be easily achieved via the interaction between the user and MCloud. Customised production can be evaluated, tested and realised with less effort and cost.

CManufacturing could benefit from the success of CComputing. As mentioned above, there are two types of CManufacturing approach; direct deployment of CComputing in manufacturing, and implementation of manufacturing applications in the MCloud. In fact, the former can be considered part of the latter. As shown in Figure 3.2, the direct practice of Cloud-based manufacturing computing provides Computing as a Service (CaaS) practice in the Cloud. At the software level, CAx, ERP, SCM software tools are deployed in the Cloud.
as service packages. With the harmonised environment in the Cloud, collaborative operations, e.g. design, simulation, process planning and data exchange, is easier in a uniform information system, thus alleviating integration and portability difficulties. Likewise, in an XaaS model the service is extended on a broader scale. Software tools providing the design capability of CAD and the engineering capability of CAE/CAM are leased as design services. CUs are able to utilise these services by providing configuration information and detailed design/engineering specifications, without a full investment in software, resources or expertise.

Figure 3.2 Collective Definition of MaaS (Manufacturing as a Service)

At the PaaS level, the Virtual Manufacturing (VM) platform, experiment environment and monitoring networks can be supplied as services. With centralised resources from the Cloud, VM and experimental tasks can be achieved in the Cloud at a faster speed and at a higher quality [103]. Yet in the CManufacturing approach, these services are further defined as the deployment activity of the tasks. For instance, experiment and storage tasks are achieved based on a service request and input from the user. The CU is served by the whole platform, without the need to own the platform or access the site.
In the IaaS mode, the entire Computer Integrated Manufacturing infrastructure is integrated into a Cloud to manage and control the supply chain. High-level data control and exchange can be achieved, with the help of virtual Local Area Network, Wireless Sensor Networks, or other methods. Another example is the deployment of Cloud-based manufacturing infrastructure. A Cloud-based CNC controller has been developed at ISW, Germany, to deliver higher performance in multi-task computing and workload balancing [161]. In an XaaS model, the capability of the whole supply chain can be packaged as a service and provided to the user, to fulfil a customised production task for the client.

3.3. Manufacturing Capability and Manufacturing Resources

Zhang et al. [68] identified manufacturing ability as a kind of resource. In practice, the main reason for acquiring a manufacturing facility is the functionalities of the equipment, but not the equipment itself. It is therefore necessary to understand and model a Cloud resource, capability and service at different levels. Definitions of resources, capabilities and services in the Cloud background are given below:

- Manufacturing Resource (MResource): material and nonmaterial manufacturing supplies including equipment, machine, device and intelligent properties
- Manufacturing Capability (MCapability): ability to transform one form into another in manufacturing domain, realised via related MResources
- Cloud Manufacturing Service (CMService): a self-contained, configurable and on-demand manufacturing service package to fulfil a user’s stated needs. A CMService can be sporadic, short-term, long-term, or strategic.

The containment relationships of MResource, MCapability and CMService can be summarised as shown in Figure 3.3. MResources are contained within MCapability as one of the essential requirements, since MCapability is realised and implemented via MResource. MCapabilities are repackaged and deployed in the MCloud as CMService, a convenient feature that can be rapidly provisioned and released by a CU.
A CManufacturing system encapsulates and implements MCapability in the Cloud as CMService packages. Manufacturing Capability is composed of Design, Production, Experimentation, Management and Communication Capability:

- Design Capability (DC) refers to domain-specific design knowledge, the expertise of the organisation and past experience from previous design activities
- Production Capability (PC) relies on the speed and quality of creating an output, i.e. product or service, to fulfil a production order
- Experimentation Capability (EC) entails experimentation knowledge and specialists
- Management Capability (MC) includes planning, organising, staffing, leading and controlling an organisation. It relies on the ability of business operational and organisational activities
- Communication Capability (CC) refers to the data exchangeability between applications/devices. It includes data transportation, speed, storage, conversion and QoS.
From a resource perspective, each kind of manufacturing capability requires support from relevant MResource(s). For each type of MCapability, the relevant MResource(s) comes in two forms; soft and hard. Soft resources contain:

- Software: software applications throughout the product lifecycle including design, analysis, simulation, process planning, etc.
- Knowledge: experience and know-how needed to complete a production task, i.e. engineering knowledge, product models, standards, evaluation procedures and results, customer feedback, etc.
- Skill: expertise in performing a specific manufacturing task
- Personnel: human resources engaged in the manufacturing process, i.e. designers, operators, managers, technicians, project teams, customer service, etc.
- Experience: performance, quality, client evaluation, etc.
- Business Network: business relationships and business opportunity networks that exist in an enterprise.

Hard resources include:

- Manufacturing Equipment: facilities needed for completing a manufacturing task, e.g. machine tools, cutters, test and monitoring equipment and other fabrication tools
- Monitoring/Control Resource: devices used to identify and control other manufacturing resources, for instance, Wireless Sensor Networks, virtual managers and remote controllers
- Computational Resource: computing devices to support the production process, e.g. servers, computers, storage media, control devices, etc.
- Materials: inputs and outputs in a production system, e.g. raw material, product-in-progress, finished product, power, water, lubricants, etc.
- Storage: automated storage and retrieval systems, logic controllers, location of warehouses, volume capacity and schedule/optimisation methods
- Transportation: movement of manufacturing inputs/outputs from one location to another. It includes modes of transport, e.g. air, rail, road, water, cable, pipeline and space, relevant price and time taken.

To formulate MCapability, a MCapability Description Model (MCDM) as a 5-tuple formula is proposed,
$MCapability$

$= \{DC(R_{SoftDC}, R_{HardDC}), EC(R_{SoftEC}, R_{HardEC}), PC(R_{SoftPC}, R_{HardPC}), MC(R_{SoftMC}, R_{HardMC}), CC(R_{SoftCC}, R_{HardCC})\}$

(3.1)

where,

$R$ – MResource includes all the resources required to carry out the Task, including hard resources $R_{Hard}$ and soft resources $R_{Soft}$.

High-performance service needs sufficient resources and suitable methodology to exploit it. Therefore, an effective MCapability is contributed by a domain-specific ability and its related resource. MCDM includes the capability of both an individual enterprise and an alliance made up of multiple participants. This means a MCapability meeting a CU’s need could be provided by a single Service Provider (SProvider) or a union of them. A comprehensive Cloud solution is required to take care of all the capabilities and resources mentioned above and provide an optimal solution. Eventually, identified MCapabilities are packaged as CMServices and deployed in the MCloud. During conversion from a current manufacturing status into CManufacturing, existing capabilities and resources should be integrated and utilised in the CManufacturing environment. Thus an interoperable, service-oriented CManufacturing system can be realised.

### 3.4. Capability Virtualisation Philosophy

From distributed MResource to robust CMService, there is a need for identification and virtualisation mechanisms. As shown in Figure 3.4, distributed resources are identified as accessible MResources with the help of the Monitoring/Control Resource mentioned above. After each resource is recognised, the next step is to unify the monitor and control methodology of these resources and wrap them as constituent MCapability modules. Mobile agents and function blocks are capable tools for this application [162]. Details of the modularised mechanism are explained in Chapter 5. These modules are then virtualised with standardised technical specifications that are published and broadcast in the MCloud. In ICMS, virtualization refers to the creation of digitized manufacturing resources in the Cloud environment. The technical specifications document the details of facilities such as feed rate, cutting speed, storage capacity, material stock and so forth. With the help of standardized methodologies, e.g. ISO10303 serials, these technical details can be described and utilized at
the application layer, which support the execution and management of CManufacturing services. Eventually, these specifications are embedded in CMServe packages and delivered to the user. From the CU point of view, when specific service packages are launched, the operational process works in reverse order compared with a system integration procedure. In accordance with the service schedule approved by the CU, relevant MCapability modules are allocated to accomplish a task. With the help of control and monitoring methods, distributed resources are organised to meet the original needs of the customers. Hence, the top-down service integration model provides a bottom-up service-oriented functionality.

To understand and utilise machining resource and capabilities, for example, requires a standardised modelling mechanism to describe and document such manufacturing features. The CMServe modelling structure can be summarised as illustrated in Figure 3.5. STEP, an international standard of data modelling, and STEP-compliant models provide rich representation models for both products and MResource, including hard and soft resources.
Apart from typical ISO 10303 standard serials [26] covering products, machine tools, cutting tools and manufacturing resources, STEP-compliant application modules have also been developed to describe software applications [163], business resources [164], shop floor data archiving [165] and so forth. Thus STEP-based data models can be used to describe the MResources.

After the MResource is well-defined, the next step is to further describe the resource in functionality terms, i.e. MCapability representation. Along with the MResource description, the MCapability data model reflects the functionality and abilities that SProviders offer. Next, the enterprise and its resource are connected and represented in a standard syntax. When a CMService is selected by a CU, the original request information is stated in a request specification document. The Standardised CMService template contains the virtualised MCapability, MResource and SProvider. In this way, a service-centric modelling structure is developed to support resource virtualisation and Cloud functionality.

CManufacturing enables time-distributed resources to be better utilised or outsourced. The elastic feature helps a business utilise MResources better. For example, an enterprise may have an average low workload in a whole year except for a peak period in September, with workloads as high as five times the average load. Additional resources, including hard resources and soft resources, are needed to cope with peak loads. The user is able to access suitable MCapabilities in the Cloud. The pay-as-you-go basis allows users to utilise the capability short-term, without a full investment in the necessary facilities and equipment. Thus the MCapabilities are integrated as a self-contained service package before being delivered to the user.
A global coordinator is required to work between CUs and MCloud. It acts as a neutral entity to interpret a user’s requests and discover suitable solutions. A Smart Cloud Manager (SCM) mechanism is suggested, to play the role of supervision in CManufacturing (Figure 3.6). SCM works in a neutral manner, responsible for coordinating interaction between consumers and MCloud. It is in charge of:

- Request Submission/Interpretation: to read the original request from a CU and map it into a standardised query syntax compliant with the CMService and MCapability description.
- Matching and Execution: to analyse the query syntax and map the specifications to MCapabilities capable of achieving the task.
- Service Input: to collect the essential features before a service is launched, e.g. 3D design, Product Design Specification (PDS) documents, machining parameters, etc. In some cases, SCM organises transport of the input to physical services, e.g. material, moulds, semi-finished products, and so forth.
- Service Registration: to register a new MCapability or delete an existing one. Cloud applications are loosely integrated on a federated basis. Thus adding or detaching an individual capability does not affect the performance of an MCloud.
- Scheduling: to plan a service procedure for CUs, including finding available MCapabilities and queuing for an unavailable resource. For SProviders, SCM is also responsible for balancing workloads and predicting production plans.
- Virtual Service Combination: to combine allocated packages with necessary add-ons as a complete solution collection.
During process execution, SCM is in charge of:

- **Service Execution**: to process virtualised workflow from optimisation results and manage service packages through event-/data-flow
- **Supervision**: to oversee the service process during execution and feed necessary information back to the CU and SProvider.
Moreover, management methods are needed to bridge solid MCapabilities with virtualised CMServices. In the midst of the SProvider domain and MCloud, an interface is necessary to support frequent communication and interaction between providers and CMServices. Smart agents are employed as the kernel of Cloud interface, to facilitate interactions between SProviders and MCloud. Cloud agents are responsible for:

- Resource Detection: detecting a change of MResource, i.e. addition, detachment, availability and modification of applications
- Resource Billing: providing a pricing interface for SProviders to determine the price of a CMService package
- Service Packaging: wrapping MCapabilities as CMServices via controlling the event and data flow of these applications
- Provider Management: providing a management environment for SProviders to publish and maintain CMService
- Monitoring: observing, detecting and recording service performance during the process and providing results to the SProvider or authorised participants.

### 3.5. System Architecture

As mentioned above, Cloud technology provides an opportunity to re-shape manufacturing, in particular in SMEs. Combined with SOA, it is capable of creating new economic growth for customised production or One-of-a-Kind Production (OKP) businesses. The flexibility of Cloud enables elastic task requesting and agile modification. In an OKP environment, the Cloud structure is particularly suitable for maintaining quality and efficiency with customised engineering variations. It provides a larger resource pool to meet user service specifications. Specialised and customised demands can be better served due to the flexible and fast-reactive nature of a Cloud Manufacturing system. Instead of Business-to-Business and Business-to-Consumer models, an Everything-to-Cloud (X2C) model is proposed, and named ICMS. Manufacturing features and elements are defined as MResources and MCapabilities, which are eventually virtualised in terms of CMServices. SCM, CU and SProvider conduct a Cloud-based SOA, which can be summarised as a request-find-provide procedure (Figure 3.7). From a service perspective, SCM plays the role of service broker and processes interaction between CU and SProvider. It takes queries from the user and provides the MCapability in the form of interoperable CMServices. These CMServices are well-defined functionalities that can be reused for different purposes.
Eventually, all the MResources and MCapabilities are integrated as CMServices in the Cloud environment. In this public Cloud infrastructure, CMServices are integrated in the MCloud that is a shared pool maintains modularised, scalable service applications on a pay-as-you-go basis. User Cloud, SCM and MCloud conduct three ICMS layers, i.e. Application Layer, Virtual Service Layer and MCapability Layer (Figure 3.8). The Application Layer contains the domain of CUs connected to the system via network. It provides graphical user interface, and user interaction methods. The Virtual Service Layer includes the central server and the supervision kernel of SCM. As the brain of ICMS, it provides management mechanism and computing capability for ICMS. The MCapability Layer modularises manufacturing applications. Physical MResources are identified by the Cloud agents mentioned above. Virtualised service data is maintained in the database cluster in Storage Cloud (SCloud). The rest of this subsection discusses the details of these three layers.
3.5.1. Application Layer: Customer and Enterprise User

In the Application Layer, CMService consumers are divided into two categories: Customer User (CUser) and Enterprise User (EUser) (Figure 3.9). CUser is defined as a customer or organisation with a request for a self-contained production task, e.g. product/sub-component production, simulation, design, etc. Assisted by the Customer Interface Agent (CIAgent) of SCM, the manufacturing request of a CUser is analysed and located by SCM, and provided by the MCloud. Thus it forms a Request-Find-Provide service chain. Original user requests are taken care of by SCM, which searches for potential solutions and feeds back the results to the user, who is able to optimise a solution based on his or her original needs and finalise the service request.
For organisations which seek a CU solution, the service is virtualised by a Drag-and-Drop interface. An organisation administrator is able to organise service and processes via graphical tools. The results, defined by graphical flow charts, are mapped to the Service Template (ST) and delivered to the control kernel. In this way, an EUser with accessibility is able to query and view the service easily. If there is any modification applied to the service plan, the result alters the ST before it is fed back to the user.

For users, ICMS provides a range of flexible manufacturing options. Customised and original requirements can be realised easily, compared with traditional manufacturing practice. It provides a large pool of opportunities to achieve bigger tasks at a lower cost. For industry it
offers new opportunities, especially for the OKP enterprises and SMEs. These enterprises are loosely integrated in the MCloud as ICMS SProviders, which interacts with consumers as a bigger virtual organisation. MCapabilities and business opportunities are integrated and broadcast in a larger resource pool, which enhances the competitiveness of the entire consortium in a fast-changing market. SMEs are always short of resources and business opportunities. With the help of MCloud, enterprise performance can be improved with fewer additional in-house staff and less cost. Thus more manufacturing objects can be achieved with minimum additional investment and effort.

ICMS takes care of traditional manufacturing tasks for CUsers, as well as collaborative production requests from EUsers. EUser refers to an organisation which owns some MCapability and requests interaction or support from MCloud, for instance additional facilities to achieve a bigger production task, or leasing costly resources/machines to complete a supply chain. In practice, customers occasionally come to a manufacturer requiring a product or capability that an enterprise does not have or only partially owns. With the help of the Enterprise Interface Agent (EIA), an EUser can search for qualified SProviders who are able to “fill in the gap”. The EUser is able to recognise related MResources and allocate a temporary partner(s) for the task. In this case, the original EUser plays the role of the “leading company” in this virtual organisation. The leading company is in charge of interacting with the customer, and collaborating with other participants as a coordinator. From the ICMS perspective, the leading company is considered the EUser, which will be assisted by the SCM module. Workflow and process control logic is partially open to leading companies to meet an enterprise’s management needs. The enterprise administrator has the authority to launch, change and terminate a service or procedure, which provides full flexibility to an EUser.

Part of the management method is open to an EUser, which makes its authority level similar to the PaaS or IaaS structure, to some degree. In this way, CUs are able to accomplish bigger and more demanding production tasks that are otherwise not possible for a single enterprise. In fact, the partner network of a company becomes boundary free. Compared with the Consumer-to-Cloud (C2Cloud) and Business-to-Cloud (B2Cloud) model, an X2C model is further strengthened by the CUser/EUser mechanism.
3.5.2. Virtual Service Layer

As the core of the Virtual Service Layer, the SCM plays the role of coordinator of CMService procedures. Consisting of the Customer Interface Agent, Enterprise Interface Agent, Broker Agent, Supervision Agent and Firewall, the SCM links service providers with users (Figure 3.10). Besides the service virtualisation methods mentioned above, SCM also contains the functions of:

- Human-Machine Interface (HMI): hosting an interaction environment for CUser and EUser, and processing operations with the help of EIA and CIAgent
- Collaboration Mechanism: hosting a collaborative environment when multiple end-users are involved in one CMService
- History Documentation: recording the interactions between CU and ICMS and service operations, to make them traceable for future needs
- Security & Accessibility Control: controlling incoming and outgoing traffic and managing the access and authority rules via Firewall
- Diagnose Report and Record Mechanism: a self-checking mechanism to determine whether the system is correct, and record operation failure where it occurs
- Pricing: determining the price of a specific CMService, and generating a service quote for CUs. Different algorithms can be deployed based on the type of service requested. For instance, the “pay-as-you-go” principle can be applied to the Cloud storage service based on the amount of data user/enterprise archiving, while credit authorisation/advance payment can be requested from the user for reasons such as costly material preparations
- Service Evaluation: collecting evaluation feedback about the service and recording it in the database after the service package is accomplished.
EIA and CIAgent work separately in dealing with different types of CUs. They organise altered HMIs for various user purposes. A service-request interface was developed to handle the service information required by the user. A data converting kernel is implanted in both EIA and CIAgent to organise user service requests in a standardised format compliant with the MResource format in the Module Database, and sent to the Broker Agent later.

Based on this service request document, the Broker Agent analyses user demands with the MCapabilities recorded in the Storage Cloud. The request description is compared with the MCapability description. The Broker Agent assists with the fulfilment of the user’s need and generates a complete ST containing the service/capability details. This forms a “Request-Find-Provide” procedure for the user. Before the ST is sent back to the user, the capability and availability of resources can be verified by the agent. If negotiation is needed (e.g. waiting for machine availability or switching to another resource), the broker feeds the result back to the user and asks for an alternative choice.

After the ST is approved by the user, the Supervision Agent organises the allocated modules in the warehouse and merges them as a “Virtual Service Combination”. Service packages are connected into the final combination, which contains essential MCapabilities and MResources. The service is delivered to the user based on the process list defined in the ST. The Supervision Agent is responsible for launching/shutting down the service by controlling the event flow of service packages. After a user finishes a task on one package, the
Supervision Agent detects the event-out and delivers a module that the user may need next. After being processed by the applications, the latest data is kept in the Storage Cloud with previous versions saved in the Backup Database. Therefore data traceability and reliability are guaranteed.

Applications in Cloud are exposed to harmful malware and its variants, which leads to security concerns. In manufacturing organisations in particular, the leakage of intellectual property, confidential and sensitive design information, and customer details may lead to undesired consequences. It is therefore paramount to guarantee data and device safety. As an important consideration of the system, Firewall Module is utilised for the security of users, providers, and ICMS itself. The functionality of the Firewall Module includes Identity, Protection and Privacy:

- Identity management is needed for both users and providers, by setting up different authorisation levels. Participants have access to different sets of data documented in the Storage Cloud and applications in the warehouse. For instance, the service provider is able to modify and update the software/device configuration implemented in the Cloud, while the end-user can only work with the application in his working domain. When it comes to an EUser, the company is able to manage a privilege regime such as the right to set up the infrastructure partially at the Virtual Service Layer.

- For protection, the Firewall takes care of both data security and the hardware protection. The Firewall agent guarantees the safety of data/software codes without leakage. Meanwhile, it makes sure that the hardware devices can only be accessed by people with the approved identity. For the manufacturing industry, remote access needs to be protected properly. Before a remote service is launched, for instance web-based machining, it has to be confirmed by both the identity management module from the Firewall and an availability message from the on-site provider.

- Privacy refers to both critical information and operation records. ICMS Firewall protects confidential data (e.g. credit card information, contract and personal details) from unauthorised access. Meanwhile, the activity of the user/provider working with the system cannot be collected or utilised by any unauthorised thirdparty.

Through the Firewall, the Virtual Service Combination is delivered to the Interface Agent where the user is able to manipulate the application. As it is controlled by the Firewall, the
user can only retrieve the resource he or she has the right to access, without disturbing others’ working domains. For specific physical devices such as machines and robots, an exclusive agreement applies, which means no other user is able to access this device or make changes while it is occupied.

3.5.3. Manufacturing Capability Layer

MResources are hosted by the SProviders in the Physical Resource domain. As discussed previously, one of the greatest challenges of CManufacturing is to integrate MResources and implement them in the Cloud. The main feature of ICMS is implementing a CManufacturing environment without major changes to the SProviders’ current organisational structure. With the help of virtualisation and standardised interfaces, the MCapabilities residing in SProviders can be fully componentised and implemented without major business restructuring.

In the MCapability Layer, both hard resources (e.g. devices, machines, sensors and materials), and soft resources (e.g. product documents, data, software and knowledge) are wrapped in the Cloud as executable MCapability applications. MResources are packaged as interoperable modules without the need for significant changes. Thus there is a need to find a method to manipulate both types of resource. With the help of mobile agents and unified APIs, the MCapabilities are packaged as self-contained service packages when they are requested. Defined initially, these packages can be launched to complete the task requested by the user. Coordinated by the Supervision Agent, these applications can be easily controlled by manipulating the in-and-out data/event flow. In short, these individual service applications can work autonomously and can be considered as “black boxes”. If there is any update or modification to the capability itself, the algorithm of the wrapper agent can be adjusted accordingly, keeping the service package autonomous. In particular, production machines can also be represented and integrated as service packages. Upon receipt of a user inquiry into machining task details, the ST is sent to the SProvider. Then the provider responds with an appropriate machining progress plan. In this case, the inputs are the product documents (data-in) and materials (event-in), while the outputs are the machined product (event-out) and updated product data.

In the Storage Cloud, the dynamic database group consists of the databases of product, project, knowledge, resource, client, provider and backup:
• Product Database contains product-related data including design, drawing, PDS, manufacturing parameters, process plan and so forth
• Project Database keeps project information about each service case, e.g. price, duration, service status, contract, evaluation document
• Knowledge Database maintains the intelligent properties in the system including experience record, reference material, standards, etc.
• Resource Database organises digitised images of MResources including status, availability, limitations, specifications and so on. With the help of agents, the resource database is updated dynamically to provide a trustworthy reference for SCM’s decision making
• Client Database collects the information of CUs such as identification, authority level and service history. According to the identification and authority setup, CUs are distinguishable and able to access their own applications without interfering with others’. CUs profiles are synchronised with the service project database. Hence the current service project can be resumed and previous ones traced
• Provider Database organises the profile of SProviders including their MCapabilities, MResource, scopes, limitations and service history. Connected with the resource database and project database, an SProvider’s information is harmonised with its resources and service.

To bridge the Service Application Cloud with Storage Cloud, mobile agents are developed to work with SCM directly. Resource Recognition Agent is responsible for identifying the MResource and updating the resource database. With the help of this agent, SProvider is able to connect its resource to the Cloud and broadcast it. Change Detection Agent helps a manufacturing resource shift from a current state to a desired state with dynamic updating in the database. Change stakeholders, normally SProviders, SCM and engaged CUs, are assisted with change acceptance and adaptation. Billing Agent works with SProviders directly to cost the CMService.

To provide an advanced mechanism to feed in the right amount of data for a specific service domain, there is also a need for an interoperable data-exchanging environment to support the Storage Cloud. In addition to product information, manufacturing resource data is also kept in the database. A quantifiable resource description model is used to describe functionalities, so that both the provider and the user are able to choose the best application to finish a specific
process. To model a product and its related manufacturing resources, STEP/STEP-NC is chosen as the main data format. As mentioned above, STEP/STEP-NC provides specific APs for specific applications, activities or environments, and these APs are built based on the same integrated resource. Thus the portability and longevity of the data are guaranteed. In the Storage Cloud, a backup database is in place.

3.6. Recap

CManufacturing requires description technologies to be used to represent various resources and manufacturing abilities. Implementing CAx applications in the Cloud via SaaS is a good start. It is necessary to integrate the current supply chain in the Cloud context. With the definitions of MResource, manufacturing essentials are identified and categorised in the context of CMService. Then MResources are encapsulated as MCapabilities that are ready to be utilised as a CMService package. The virtualisation chain of ICMS can be understood as a so-called “servicelisation” procedure, which identifies and delivers the existing MResource for the sake of service. Manufacturing features are modified and integrated into the Cloud environment. In a flexible and collaborative environment such as ICMS, modularised capacities can be organised and reused based on various requests from different users. Thus it guarantees the flexible service-oriented nature of CManufacturing.

The ICMS framework was proposed to meet the requirements of CManufacturing and CMService components. The Manufacturing Capability Layer provides both physical and intellectual manufacturing capabilities. Both software packages and hardware devices can be integrated as autonomous modules and implemented in the Cloud. With the help of Storage Cloud, digitised manufacturing profiles and specifications are maintained in the Storage Cloud, which includes thorough data support for the CMService process. The use of standardised product/resource formats maximizes the portability of manufacturing documents, and guarantees a smooth data flow. The use of agents also keeps the database up to date, and supports reliable and optimal service scheduling.

In the user domain, Cloud customers are treated as CUser or EUser, to meet different requirements from users. Customers can switch between the roles of CUser and EUser according to the character of the service. Even SProviders can be considered a special type of user, requesting capability outsourcing as a special kind of service. In the ICMS structure,
CUusers, EUsers and SProviders are considered equal. The SCM evaluates service requests in a neutral manner, based on the functions of CMService instead of the SProviders’ control.

Additionally, a central server is placed in the Virtual Service Layer, providing computational capability for the SCM. As an intelligent centre for ICMS, SCM provides intelligence for service execution, supervision and evaluation. SCM coordinates the interaction between CUs and MCloud, providing the “Best-of-the-Best” solution to the CU. Searching and optimisation mechanisms are necessary to match a user inquiry and deliver an effective answer. To achieve service virtualisation and Cloud implementation, it is necessary to understand MResource, MCapability and SProvider in the CMService manner. Comprehensive and interoperable data models are essential to represent the manufacturers and their capabilities. In the next chapter, an SCM-based service consolidation mechanism is proposed in detail, along with the data model development regarding MCapabilities. There is a need for an SCM-based service consolidation mechanism. International standards offer some potential solutions, but more work is needed to fulfil the specific needs of CMService. It is necessary to extend with the description models in international standards to Cloud-based elements. The CManufacturing philosophy requires adoption of more advanced technologies, e.g. remote monitoring and controlling technologies to manage facilities in the Cloud, CComputing technologies in Cloud-based CAx implementations, standardisation of resources and CMService descriptions.
Chapter 4. Smart Cloud Manager: Cloud Service Provision

In CManufacturing, it is necessary to develop a service-oriented mechanism to coordinate manufacturing capabilities and Cloud services. This chapter presents a Smart Cloud Manager mechanism that functions at the Virtual Service Layer of the system. Detailed service procedures and logic flows are then given. The objective is to support manufacturing services in the Cloud paradigm. Standardised data models were developed to represent the CManufacturing Service, and service queries. Major work presented in this section has been reported in paper [156].

4.1. Overview

CManufacturing offers opportunities to improve the performance of the manufacturing industry. A mature CManufacturing system can provide a configurable resource pool containing both hard and soft MResources. MResources are integrated as MCapabilities, which are self-contained mobile function modules. Also, MCapabilities are virtualised as CServices and implemented in the Cloud. Eventually, CMServices can be integrated as a virtual service combination and delivered to Cloud users. There is a need for a control mechanism to be developed to take care of Cloud control, service supervision and execution, optimal allocation, resource sharing, collaborative work and service identification/virtualisation. Additionally, in this resource-capability-service package chain, data models are needed to help understand, describe and virtualise resources as capabilities, so that services can be utilised by remote users directly. In this way, SCM can be developed to act as the neutral supervisor of ICMS. In this chapter, a detailed SCM mechanism is discussed, along with data models supporting resource recognition and service virtualisation.

4.2. SCM Mechanism

In the CComputing context, CUs interact with the Cloud via the network. The storage space, computing power, software and platform services are implemented in a Cloud infrastructure,
which is hosted by multiple servers at remote locations. During the execution of CMService, the service process only exists between CUs and Cloud, without the involvement of any third-party. However, manufacturing is a special type of service which requires more human engagement than computing tasks do. In particular, when it comes to physical resources, interaction between only a CU and Cloud is far from enough. For example, when a CU requests a metal machining service, a virtualised CMService is determined in the Cloud. Then it is necessary to convert the virtual service into materialised capabilities. A physical manufacturer needs to prepare the resources (i.e. material, machine and personnel) to accomplish the task and deliver the finished product to the user. Thus the role of a Cloud participant is quite different from one in a CComputing environment. The stakeholders of CManufacturing are defined as:

- **SProvider**: a Service Provider delivers MCapability as a service. SProvider includes the supplier of a production, warehouse, management, testing and manufacturing-related service retailer, e.g. consulting, data storage, transportation, etc.

- **Cloud Provider (CProvider)**: the differences among CProviders are the major dissimilarity between CComputing and CManufacturing. In CComputing, CProvider is the main body that hosts the service. It is the organisation that builds the infrastructure and creates financial profits. In CManufacturing, CProvider is a third party that is not involved in manufacturing tasks, and only provides the computing ability and a collaborative IT environment. In contrast, SProvider is the manufacturing enterprise or organization creates profits and offers MCapabilities as mentioned above, including the computing capability and Cloud functions to support them. Thus CProviders in ICMS can be treated as a special kind of SProvider who provides the Cloud environment as a service. It plays the role of a neutral IT operator or administrator who is not involved in the task of Cloud consumers, who are normally SProviders and CUs.

- **Cloud User**: a CU is a consumer of CMService, and is identified as a CUser or an EUser.

4.2.1. SCM Involvement Levels

As mentioned above, manufacturing services are different from computing service in multiple scales, in particular at the engagement level of human expertise. In a Cloud-based
manufacturing environment, the involvement of a CU can be summarised as stated in Figure 4.1. In the initial stage of a CManufacturing system, human interaction is unavoidable during the decision making on manufacturing procedure. In addition to sending and receiving queries with SCM, the CMService also needs a CU’s configuration and effort directly. Interaction with SProvider may be required as well, e.g. for consultation, modification queries, design details and so on. The engagement level is relatively high in the preliminary phase of CManufacturing. At the intermediate stage of CManufacturing, SCM takes more responsibility for service coordination with the help of intelligent and optimisation kernels. Configuration queries and data are mainly processed via SCM. The interactions between CMService, SProvider and CU are decreased. Artificial intelligence is fundamentally utilised without the need for human interaction. In an advanced and mature CManufacturing model, CUs have full confidence in the system’s Artificial Intelligence (AI), which is capable of optimising and executing CMServices independently. The interaction between CMService, SProvider and CU is minimised since all events and data flow through SCM. As long as the service specifications (i.e. request, price, time, quality, etc.) are well defined by the CU, SCM is able to locate suitable MResources and accomplish the task within these preference variables. In an ideal CManufacturing environment, manufacturing objects are achieved in time, in high quality and in satisfactory price (3 Ins), without CU’s knowledge about how and where the object is produced. An ideal CManufacturing system would play a role of “black-box”, which takes in the service description and feeds back with service results.

![Figure 4.1 Multiple Engagement Levels of Cloud Users](image-url)
4.2.2. SCM Process Flow

Thus, to fully utilise AI and human expertise/knowledge, a decision-making model is proposed. SCM works in a neutral manner and consists of EIA, CIAgent, Broker Agent (BA), Supervision Agent (SAgent) and Firewall Module. EIA works with EUsers and CIAgent handles requests from CUsers. Although the Graphical User Interfaces (GUIs) and algorithms of EIA and CIAgent are different, the service-oriented working procedures between CU and SCM are almost the same (from the SCM perspective) (Figure 4.2). After a user request is collected by the Interface Agent (IA), BA communicates with the Provider Database and maps the requirement to the available CMServices. As long as the user modifies and confirms the service package, an ICMS ST is generated and delivered to the SCM. Based on ST, Supervision Agent starts up and works with the Service Application Cloud (SACloud). Specific CMServices are organised and launched to meet user expectations. The final service output, which can be product, computing data or technical document, is sent to the user at the conclusion of the service. After the feedback/evaluation document is submitted by the user, the CMService is terminated.

**Figure 4.2 Cloud Service Procedure**

As the supervisor or brain of ICMS, SCM analyses and controls CMServices to meet the user demand. Inside SCM, the interactions among IA, BA and SAgent are summarised in Figure 4.3. IA is developed between a user and ICMS. It provides both a graphical service and a virtual working environment. Firstly, the user is identified by the Firewall and matched with
the profile in User Database. After identity confirmation, the user is asked what kind of service he or she requires. The user’s request is collected by the IA. Service queries can be modelled in quantifiable factors, for example size, shape, mechanical elements of machined products, etc. The details are converted into a standardised service format. Based on these details, an internal request document is generated and sent to BA. According to the request document, the search engine embedded in BA searches in the MResource database for potential solutions. Afterwards, an initial ST file is created and sent back to the user, containing all the possible tracks via matched service provider nodes. A Cloud consumer is able to view all these solutions in human-machine interface along with the suggestions from SCM. Based on factors such as time, cost, quality, functionality, etc., SCM’s recommendation is visible to the user at different levels of detail. Optimisation objectives can be set up based on maximum QoS, minimum cost, functionality/availability, fault tolerance, cost-effectiveness, energy efficiency, process constraints and so on. However, decisions on manufacturing are difficult to make due to the complexity of manufacturing processes, variety of machines/devices and uncertainty of product features. When multiple resources and variables are included, it is even harder to predict a reliable and optimal service solution for the user. Thus, ICMS allows CU to engage in the decision making on a CMService. If a Cloud customer is not satisfied with any of the suggestions provided, he or she is able to modify the ST.
At this stage, the Cloud customer is able to either optimise the ST via BA intelligence or do it manually. If the customer prefers to utilise AI continuously, CU is requested to modify his or her original search request by providing more details, or to modify technical variables. Then the altered request condition is sent back to BA, which will process one more round of analysis and service detection. Improved optimisation results are re-sent to the CU until they are approved. On the other hand, if the Cloud customer chooses to improve the ST manually,
he or she can work on it via GUI and allocate a preferred provider. This way, both expert knowledge and optimisation are utilised in the SCM.

As soon as the user confirms the ST, the specific Cloud Services are launched by the SAgent, which is responsible for monitoring and controlling all the activities of the Cloud service modules. By marking and manipulating the event/data flow of all the application modules, the ST is executed accurately as defined.

As discussed in Chapter 2, the attempt at a neutral data format shows promise in combating interoperability difficulties. It provides full data portability among applications within the manufacturing environment. However, in practice it is difficult to implement an adaptable format among all the manufacturing applications. To help manufacturing systems benefit from the interoperability of neutral syntax and the features of individual applications, a Cloud structure could be a feasible solution. Combining data-centric and process-centric (service-oriented) factors, ICMS provides a solution to bridging heterogeneous applications. ISO standards were chosen as they provide a large number of data models for MResources, for example machine tools [33, 166], controllers [167], cutting tools [161], engineering analysis [55] and product data [54]. However, there is no method for describing CMService or a service-oriented object. To fully utilise the existing models and extend them into the Cloud context, it will be necessary to develop data models dealing with Cloud implementations. There is thus a need for a standardised syntax to model an SProvider, CMService and service queries in the context of Cloud.

4.3. Cloud-based Data Models

Among the modelling languages available, EXPRESS was chosen to describe the CMService and its related resources [168]. EXPRESS is a powerful language for product modelling and many product configurations [169]. It contains solid and robust features, e.g. entity and constraint declarations, dynamic multiplicity and possibility of extensions. Moreover, it is compliant with international standard models (e.g. ISO 10303, ISO 14649, ISO 13399 and ISO 15531), which provide data models throughout the CAD-CAM-CNC chain. Although, the current STEP models do not support particular functionalities in CManufacturing paradigm. The development of data models in EXPRESS provide a flexible mechanism for utilising the existing data models of MResource, and extend them to the Cloud-specific utilisations.
4.3.1. Service Provider Data Model

As shown in Figure 4.4, the enterprise which provides CMServices is defined as an SProvider in EXPRESS-G, the graphical notation of EXPRESS [168]. Therefore, a manufacturing enterprise can be described, in Cloud terminology, as provider profile and service properties. The provider specifications describe information about the organisation, while service specifications present the MCapability in terms of the service that it provides. Note that each company has a unique Enterprise Entity, while its Service entities can be multiple. This means each SProvider has a singular description but provides a number of CMServices. Hence, organisation consistency and service variety are maintained concurrently. Entity Enterprise outlines the properties of CManufacturing via entities Provider_ID, Company_Name, Provider_Size, Provider_Capability, Provider_Location, Provider_Contact, Prior_Experience, Provider_Evaluation and Provider_Description.

![Figure 4.4 Cloud Service Provider Model in EXPRESS-G](image)

Entity Provider_ID provides a unique identifier in the MCloud for an SProvider. It is assigned when an SProvider registers its first CMService, according to the encoding system of ICMS. Based on its Provider_ID, all the CMServices from a provider and its related service history can easily be traced.
Entity Company_Name stores the general name/trademark, which differs from the Provider_ID. While Provider_ID is developed for internal utilisation within the system, Company_Name enables the user to gain the profile and information of an SProvider.

Entity Provider_Size categorises SProviders based on industry, ownership structure, revenue and number of employees. Thus users are informed by the quantifiable scale of an SProvider.

Entity Provider_Capabilities describes the MCapability of an SProvider via sub-entities Design_Capability, Experimentation_Capability, Production_Capability, Management_Capability, Communication_Capability and Manufacturing_Resource, which are compliant with the aforementioned MCDM model. Entity Hard_Resource and Entity Soft_Resource describe the MResources that support a specific MCapability. These entities can be connected to a standardised data model directly, for example ISO14949-201 for machine tools, and ISO10303-45 for material and engineering properties. Hence, the MCapability of an SProvider is described in an explicit and scalable data model.

Entity Provider_Location records the geographical information of an SProvider. Based on the supplier location and destiny position of a user, SCM is able to calculate potential shipping time/cost automatically.

Entity Provider_Contact maintains the general contact information of SProvider, e.g. Staff, Address, Phone and E-mail.

Entity Prior_Experience records the service history of one SProvider, which is visible to the Cloud administrator and the provider itself, but not entirely to the CUs. Entity Provider_Evaluation documents the feedback from these service consumers. Based on these two entities, the performance of the service experience is modelled explicitly.

Entity General_Description provides text-based statements of SProviders. While SCM delivers the potential SProviders to the user, the content of this entity is also visible to the user. In this manner, the unscalable information is contained in the Enterprise Model as well.

### 4.3.2. Cloud Service Template Data Model

In the second category of enterprise attributes, the recognition of a specific CMService is modelled via Entity Cloud_Service_Template and its entities, i.e. Service_ID, Service_Cost, Price, Time, Shipping_Price, Shipping_Time, Service_Status, Service_Document,
Data_Object, Pre_Condition, Availability, Resource, Quality_Evaluation, Technical_SupportCapability, Warranties and Service_Description (Figure 4.5).

Figure 4.5 Cloud Service Model

Entity Service_ID addresses the unique identifier of each CMService. With the help of Service_ID, the entire technical service data and history is traceable.

Entity Service_Cost documents the value of CMService in a monetary form. This entity provides an explicit model of the cost of accomplishing a service object. Service_Cost is only visible for SProviders to understand their MCapability internally, and to quote a reasonable price for external CUs. Service_Cost is described with the help of entities Cost_Per_Unit and Cost_Quantity. The Service_Cost model structure is inherited by four subtypes, which divide the service cost into four categories; Cost_Material, Cost_Machining, Cost_Labour and Cost_Management.

Entity Price maintains the expense of a specific service for a CU. Some of the service applications can be directly quantified, e.g. power, material, freight, etc.. In more complex cases, the SProvider is responsible to evaluate the cost of service, e.g. design, machining and labour, and then provide the initial service quote to the CU. Based on the price factors of the
service described in entity Service_Cost, SCM is able to evaluate the financial value of a
service task. With the help of BiA, SPs are able to update the assessment factors or change
the final price of a service manually, according to the special features of the service itself.
This enables expert experience to be reflected in the billing result before the service proposal
is delivered to the CU.

Entity Time describes duration of a CMService. Based on the service input, BA is able to
predict the approximate time of service. For instance, time of machining can be calculated
beforehand according to the volume of material removal and the feed rate parameter provided
by the resource model mentioned. Combined with MResource availability and schedule, the
length of service can be predicted by SCM reliably. Thus it provides trustworthy data for
service scheduling and process optimisation.

Entity Shipping_Price presents the cost of transportation. In the CManufacturing environment,
it is not uncommon for a CMService to be served by multiple suppliers who are
geographically distributed. Manufacturing essentials, i.e. service input/output, material and
cutting tools, need to be transferred between these SProviders, who are loosely connected by
service tasks. The cost of freighting is stored by this entity. According to the location
information in the aforementioned Provider_Location entity, BA is able to calculate the
shipping price based on transportation method and distance. If there is a restricted shipping
requirement because of special material/device, SProvider is enabled to modify the quote
text before it is delivered to the user.

Entity Shipping_Time contains freight time information. Combined with entity Time, it
enables CUs to predict the duration of a total service, which contains the cost of time
including preparation, processing and transportation.

Entity Service_Status contains information about the running stages after a CMService is
launched. Stages of implementation are described via entities Initial, Ready, Running,
Skipped, Completed, Overload, Paused and Cancelled.

Entity ST_Document keeps the path and version of a ST file. When a CU is working on a ST,
all versions of the ST are recorded by this entity. Thus the service/modification history is
maintained.
Entity Data_Object records the technical document(s) related to the CMService. The optional attribute of this entity is Entity Project, which is compliant with the top level of a neutral data format defined in ISO10303 [170].

Entity Pre_Condition defines the requirements prior to the start of CMService. Limitations or preparations of the service input are recorded and published, e.g. limits of size, material preparation, heat treatment and so forth.

Entity Availability represents the availability and working conditions of a CMService. This entity is dynamically updated by Change Detection Agent. With the help of this agent, the CU is informed of the trustworthiness of availability without major delays. CUs are able to select the available CMService only, or queue on the list, waiting for the preferred package until it is ready to be used. The availability information is further described by its attributes, i.e. Module_Completion, Function_Fitness and Security.

Entity Resource defines the manufacturing resource that is required for a specific service. Its structural attributes (Hard_Resource and Soft_Resource) are compliant with the resource representation of Entity Enterprise. Thus the resource specifications, from both service point of view and enterprise point of view, are shaped and integrated in the Storage Cloud.

Entity Service_Type gives the category into which the service falls. Working with entity Service_Subtype, this entity provides hierarchical classification information about a service. It helps the Cloud administrator to maintain the CMService pool. CUs are able to explore CMService within a specific domain.

Entity Quality_Evaluation maintains assessment feedback from CUs after a service is terminated. At the conclusion of a service, SCM collects the CU’s service evaluation in standard formats. Service performance information is documented and kept in SCloud. CUs can access the quality records of the CMService and select the preferred solution, based on previous background and CU experience.

Entity Technical_Support_Capability describes the range of technical assistance that SProvider provides during and after a service procedure. It contains content such as enquiry, consulting via telephone calls, SMS, online-chat, email, fax, etc.
Entity Warranties presents a guarantee of the reliability of the CMService, with conditions. It is provided when the service is published in the MCloud and set up as visible for potential CUs.

Besides all the technical information modelled by the aforementioned entities, the service can be described in general terms via Entity Service_Description. A user-friendly introduction of a CMService is displayed by texts, figures and tables.

4.3.3. Cloud Service Query Data Model

To describe a user query about a CMService, the Cloud_Service_Request model is used (Figure 4.6). Cloud_Service_Request is compliant with the SProvider and Cloud_Service_Template data structure. As a bridge between a user’s original demand and the CMService in the MCloud, it provides a neutral and standardised methodology to document the query. Via the GUI, the service description is arranged as a structured statement and transmitted to the SCM. Based on this data, SCM is able to suggest a solution from the resource pool, based on the terms and mapping preference. The request data is shaped via entities such as Request_ID, CU, Request_Status, Service_Type, Data_Object, Service_Time, Service_Price, Preference, Request_Description, Service_Document and optional entities Preferred_Resource, Preferred_Provider, Preferred.Region, Quantity_Of_Service_Output and Keywords.
Entity Request_ID gives a unique serial number for a CU’s request. When a new query case is created, a permanent Request_ID is assigned. Users are able to resume, modify and review the request case. Additionally, all the relevant Cloud behaviour and history are traceable, based on a Request_ID.

Entity CU describes the customer information of a service client. When a user launches a new service request, his or her basic profile is attached to the case, including Client_ID, User_Region and Type_Of_User. User_Region describes the regional information of the CU for resource and transportation optimisation. Type_Of_User maintains the classification of CU, which is either a CUser or an EUser.

Entity Request_Status maintains the operational condition of a request. The variables of a process status are Created, Under_Evaluation, Approved, Running, Waiting, Terminated and Finished.

Entity Service_Type is compliant with the entities of the Cloud_Service_Specification model. The user is able to choose the class of service needed, e.g. design, simulating, machining and
so on. With the help of Service_Type and Service_Subtype, a CU request is easily mapped to a CMService in the MCloud.

Entity Data_Object stores the input service information if necessary, for instance PDS, 3D design, technical document, etc. The SCM is able to evaluate a request and provide a solution accordingly.

Entity Service_Time describes the duration requirement of a CU. The SCM predicts the length of each potential reaction. By defining the upper and lower boundary of a CMService, the solution lies within the range of user expectation only.

Entity Service_Price provides a similar definition of cost conditions. Based on the service input, BA quotes and prices all the available applications in the resource pool. The results of searching are filtered by the upper and lower boundary of service expenses.

Entity Preference keeps the optimisation setup when there are multiple options that meet a CU query. The SCM is able to recommend CMService packages based on the rules of maximum QoS, minimum cost, availability, cost effectiveness and energy efficiency.

Entity Request_Description maintains additional query details that are not included in the model. A general syntax is utilised for review by the SProvider.

Entity Service_Document connects a CU demand with the ST file when it is approved. Then the request proceeds to the stage of execution.

Optional Entity Preferred_Resource keeps a user’s predilection for resources and facilities, for example specific machine tools, testing methods and design software. The structure of this entity is compliant with the Hard_Resource and Soft_Resource entities of the SProvider model previously mentioned. Thus a user request can be directly connected to the MCapabilities in the Cloud.

Entity Preferred_Provider describes the preference for SProvider. In practice, users may choose the service provider they are aware of, or the ones they worked with in the past. For an EUser, it is possible to select an “old friend” that has been in its business network before. In this way, Preferred_Provider is optional, to allocate the choice of a favoured service supplier and give it the highest priority in the selection pool.
Entity Preferred Region states the provider location a CU favors. Clients are able to choose a provider that has a strong MResource, or one that is close to the destination.

Entity Quantity of Service Output specifies the explicit amount of service, e.g. amount of product, number of test objects, volume of storage, etc. It gives a scalable size of the service for a specific service task. For a task that is hard to calculate, for example material removal volume for machining and length of simulation, the BA will evaluate the task and provide a price invoice for a CU.

Entity Keywords is an index term for service retrieval. With the help of this descriptor, SCM acts as a search engine and locates the most suitable CMService efficiently.

Thanks to the service request, CMService and SProvider models, the data can be modelled from initialisation to implementation stage in the SCloud. Information packages can be submitted, retrieved and maintained over the Internet regardless of the locations of the central database and server. For data storage queries, a customer’s private data is not maintained in the SCloud directly. In the background, data centers are hosted by third parties that are integrated as SProviders in the MCloud. In this way, a storage task is integrated as one of the CMServices in the virtualised service pools.

For the convenience of web-based data exchange, XML was chosen as the data carrier. XML is a modelling language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It provides a simple, general, and usable textural data format. XML was formally adopted as a part of an international standard to present EXPRESS schema [171]. Note that an XML document transmitted from the client side is compliant with the EXPRESS-G schemas but in a flat data structure (Figure 4.7).
The meta-model was developed for system portability and interoperability. Even though the EXPRESS schema provides a robust and comprehensive structure for objects, it is difficult for users who are not programming experts to work on. Moreover, EXPRESS’s multi-tier structure requires interpreters or add-ons before it can be imported into other software tools. Therefore, a flat meta-model was utilised, to enhance data portability. The meta-model only
keeps the information of entity instance and temporarily suspends attributes and inheritance logics. In this way, CUs are able to view and process such data via general software tools, e.g. web browser, Microsoft Excel, Access and so on. It provides flexibility and reusability on the User’s applications. When the data comes back to the MCloud, it is mapped back to the structured model tree. Then SCM continues to process the service process and searches for all possible solutions.

4.3.4. Cloud-based Data Model Set

Therefore, a set of data models was developed to support Cloud-based manufacturing systems. As shown in Figure 4.8, service-oriented models have been designed to describe essential features for CMServices. As the kernel of the database, Cloud_Service_Template provides methods to represent and utilise MResources as MCapabilities. Via entity Resource, Cloud_Service_Template is connected to the MResource models defined in the SProvider model. The Cloud_Service Request model is connected to an ST as an attribute, also. When a CU is requesting a service package, he or she is able to assign the preferred MResource or SProvider. The former is connected to the resource entity, and the latter has Cloud_Service_Provider as its attribute. There are two subtypes of MResource, i.e. Hard_Resource and Soft_Resource. Hard_Resource is connected to the exiting ISO standards e.g. item_definition entity from ISO 13399 [172] that contains cutting tool data structure, and a resource entity from ISO 15531 [164] that provides manufacturing resource models. For instance, the Hard_Resource entity is directly related to the international standard for drilling cutting tools [166]. As one of the attributes of Entity Limitation, the maximum usable length for a cutting tool is connected with the service’s specifications. When this cutting tool is assigned and packaged as an MCapability, its ability and limits are explicitly reported to the SCM. Thus it provides an interoperable environment to connect physical resources to a Cloud server, and a unified method to integrate MCapabilities into CMServices.

Moreover, Soft_Resource is connected to the data model of nonphysical resources like the Software entity from the ISO 10303-1746 [163] standard describing intelligent products. Additionally, when a service query is made by a consumer, a CU is able to provide an initial product description, e.g. 3D design, manufacturing document, etc. The Data_Object entity is connected with product documents, for example, the Project entity of ISO 14649 and geometry from the ISO 10303-203 STEP document [54]. In this way, CMService-based data models are connected and integrated with international standards, forming an interoperable
data environment in the MCloud. When a CMService is launched, all the relevant resource, product and provider information is integrated and connected in a unified data syntax.
Figure 4.8 EXPRESS-G Models Supporting Cloud Services
4.4. Recap

To recap, ICMS provides a flexible and distributed environment for shared MCapabilities. The “Request-Find-Provide” SOA chain is realised with the help of mobile agents encapsulated within SCM. This enables SCM to collect original requests from CUs, to interpret them into a standardised format compliant with the request model, and to search for possible solutions in the MCapability pool. Matched results are integrated as a CMService combination and documented in the CMService template before they are delivered to the user.

Thanks to the standardised data models mentioned above, MResource, MCapability and CMService requests are unified and integrated into the ST-centric data model segmentation. The data models provide a rich representation method on manufacturing-related features from resource level to service level. The ICMS service modelling method is compliant with ISO standard serials. In this way, it provides an interoperable environment covering operational processes throughout the supply chain. In particular, it offers a number of benefits:

- Data Interoperability: the manufacturing business is commonly troubled by data interoperability issues. CAx applications are widely utilised throughout the production stages. However, these applications are developed by multiple providers using different programme languages and document formats, leading to a heterogeneous data environment. It is difficult to communicate between software tools using different kernels. Data loss and errors often occur during format conversions. By using ICMS, the explicit specifications of software tools are represented as a soft resource in the MCloud. Detailed descriptions, e.g. input and output format requirements, are visible to all the CUs. Interoperable problems can be easily identified and avoided. Users are able to choose SProviders that can smoothly communicate with each other, or alternatively allocate a reliable data conversion service beforehand as one of the CMServices. Therefore, interoperability is achieved even before a CMService is launched.

- Globalisation/Sub-Contracting: with the help of the Internet of Things, manufacturing services/capabilities are virtualised in the MCloud. Compared with web-based manufacturing, ICMS provides a more distributed and flexible environment, which knocks down the boundaries between organisations/enterprises. Powered by the SCM search engine, it is easier to find business partners/sub-contractors based on their
performance of service, regardless of who or where they are. The production ability of an enterprise is strengthened by the shared MResource pool.

- Global Optimisation: since services are broadcast in the Cloud, service solutions can be improved and optimised, based on the virtualised service modules implemented in the Cloud. According to this cognition, SCM predicts the service performance features beforehand, e.g. cost/time resulting from preparation, machining, transporting and packing stages. SCM is able to organise the best solution, based on service requirements and service constraints. So much so, the global solution is optimised based on particular factors or user preferences. Production strategy is predicted and evaluated at a higher level by SCM, providing a “Best-of-the-Best” solution from a wide resource pool.

- Customised Service & Specialised Demand: customisation is becoming more and more important in modern manufacturing, especially for SMEs. In a machine shop, specific cutting/machine tools are required for a particular job. With SCM, it is easy to locate required facilities in the resource pool. Therefore, specialised objects are achieved without additional investment in costly facilities and expertise. Some particular requests can be achieved with minimum cost and effort.

- Cost-Saving: by adopting the CManufacturing concept, manufacturing costs can be reduced. With the shared MCapabilities available in the Cloud, an optimal business solution could be easily found according to optimised results. Since the features of SProviders are virtualised in the SCloud, the user is more likely to find a supplier with better performance, cheaper labour, higher productivity and better geographical location. As a consequence of time-critical or cost-critical optimisation strategies, the performance of the service solution can be predicted and improved from a higher level, in the wider scope of Cloud. For service consumers, a cost-effective combination can be achieved with the help of BA. Besides the cost of the service itself, the cost of strategic decisions is reduced as well. With technical specifications highly integrated in the SCloud, the cost of management, analysis and comparison also decreases.

- Facility Utilisation: resources can be shared in a Cloud. Technical details and availability can be dynamically updated and published in the SCloud. Thus manufacturing resources/capabilities can be better utilised, based on trustworthy knowledge. Production tasks can be easily balanced between high-usage facilities and
low-usage ones. From a user perspective, CUs are able to choose the available qualified providers for urgent jobs, or to wait in the queue for the preferred facility. Therefore, facility utilisation is improved by a widely shared environment and reasonable schedule.

- Better Enterprise Performance: when it comes to cost/time management, ICMS improves not only the experience of CUs but also an enterprise’s performance as a CMService provider. MCapabilities are accessible in the Cloud, bringing more business opportunities. With the help of SCM, an SProvider is able to increase production volume and react rapidly to market changes.

Hence, SCM provides a service organisation and optimisation mechanism in the SOA chain. With the help of service representation methods, CManufacturing service can be described, virtualised, deployed and offered in the MCloud. Solid MResources and virtualised CMServices are connected and integrated at the data and service level, thanks to modelling methods compliant with ISO standards. Subsequently, the challenge is how MResources can be connected to the MCloud at the controlling level. There is a need to develop a mechanism to connect Cloud servers and computers to manufacturing facilities, including soft resources (e.g. software and data) and hard resources (e.g. machine tools and monitors). In the next chapter, a resource integration mechanism is proposed, with modelling methods that are compliant with Cloud service and international standards.
Chapter 5. Service Application Cloud: Virtualised Manufacturing Capability

This chapter describes the manufacturing capability virtualisation mechanism in the Service Manufacturing Cloud that plays the central role in the Manufacturing Capability Layer. Function Blocks and software agent technologies are discussed from the CManufacturing perspective in detail. Next, a novel integration mechanism is proposed, namely the Virtual Function Block. Based on software agents, virtual function blocks are able to manipulate and integrate manufacturing resources via event states and data flows. A standardised data model of the virtual function block was also developed, compliant with the Cloud service model sets. The major work about the Virtual Function Block method was published in the International Journal of Enterprise Information Systems [162].

5.1. Overview

The criteria for a contemporary manufacturing information system can be summarised as portability, interoperability, visibility and longevity. As a requirement for portability, data or applications should be easily moved from one application to another. With interoperability, software applications must be able to run on different hardware and operating systems. Clearly, the concept of interoperability is different from portability. Interoperability refers to the ability of procedures, software, systems and organisations to work together, regardless of the platform upon which they are deployed or built. Portability relies on the ability to move object (e.g. data) from one platform to another. For instance, the Java application [173] can run on different platforms with other applications, while PostScript [174] is a kind of control language that can be understood by many printers. The former is a case of interoperability whereas the latter portability. Data visibility is also important, which means the right type of data is made available to the right people, at the right time, in the right manner. Data longevity refers to data outliving the software and hardware on which it originated, and both data and applications can be extended to take advantage of new and future techniques. In an interoperable manufacturing environment, devices with a “Plug & Play” feature are
advantageous. Furthermore, so-called “what-to-do” data, meaning manufacturing information at the task level, is more easily interchangeable along the CAD/CAM/CNC chain than the traditional “how-to-do” data describing the method.

As mentioned, there is a need to integrate MResources as ready-for-use MCapability modules, to meet the criteria mentioned above. MResources stay in different forms within an enterprise. To effectively bridge the existing resources with the MCloud is a substantial need. MResources, including hard resources (e.g. machine tools and monitors) and soft resources (e.g. software and data), need to be provisioned in the Cloud in terms of service. SACloud is the Cloud infrastructure that hosts MCapabilities, i.e. the image of these MResources at the MCapability Layer of ICMS. An intelligent mechanism is needed to play the role of universal interface between MResource and the SACloud. Moreover, the mechanism is also responsible for the execution of service procedures defined by SCM. The Function Block mechanism offers a control and integration methodology by manipulating input and output variables. Thus, inspired by a function block concept, an integration mechanism is proposed, namely Virtual Function Block (VFB). VFBs are realised via software agents that are in charge of event control and communication. In this chapter, function blocks and software agents are introduced in detail, followed by the VFB methodology.

5.2. Function Blocks

Function Block is an open standard for distributed control and automation, published by the International Electrotechnical Commission (IEC) [175]. The objective of IEC is to promote international co-operation concerning standardisation in the electrical and electronic fields. The IEC collaborates closely with the ISO. The standard for function blocks (IEC 61499-1) was prepared by IEC technical committee 65: industrial-process measurement and control.

5.2.1. History of Function Blocks

In early 1990, Technical Committee 65 (TC65) of the IEC received a New Work Proposal (NWP) to standardise certain aspects of the application of software modules called Function Blocks (FBlocks), in distributed Industrial Process Measurement and Control Systems (IPMCS). IPMCS’s utilisation of the ‘fieldbus’ (IEC 61158) standard and development in Working Group 6 of Subcommittee 65C (SC65C/WG6) were especially emphasised in the NWP. However, FBlocks were also an essential part of the programming language standard IEC 61131-3 for programmable controllers developed in SC65B/WG7. Therefore, TC65
determined that a common model for the use of FBlocks was required, and assigned the new Project 61499 to a new Working Group 6 (TC65/WG6) from the parent committee.

In 1991, TC65 approved a New Work Item (NWI) for the development of the international standard for the use of software objects (i.e. Function Blocks) and in early 2005 IEC published and standardised certain aspects of the application of software modules. It is a common model for the use of FBlocks. The IEC 61499 standard is the result of more than 10 years research, undertaken by leading experts in the software architecture of industrial automation systems, representing the biggest global vendors of automation products and solutions.

The need for this standard came from several studies and research programmes that were started or conducted in the late eighties and early nineties last century [176]. The standard is a response to the new challenges in hardware and software, such as the advent of distributed networking control devices, the modularisation of machinery and component-based software solutions. An increasing portion of the Intellectual Property (IP) related to the domain of industrial automation is reflected in software form. This automation software spans diverse functional domains, such as control, diagnostics, process rendering, modelling, Human-Machine-Interface (HMI) and communication.

IEC 61499 leads to the development of new engineering technologies that are aimed at reducing design effort and enabling fast and easy reconfiguration. The standard is unusual when it is compared with other standards, being used in the domain of control and automation. Literally, it defines ‘reference architecture’ for the software Distributed Process Measurement and Control System. The standard adopts advanced software technology, such as the encapsulation of functionality, component-based design, event-driven execution and distribution.

A function block is a software unit that encapsulates algorithms that can be designed to behave in a similar way to an electronic device or a circuit. This means that a function block can represent a small task in a control plan or it can encapsulate multiple small control units. The unit, designed for a specific purpose, contains the process algorithm and control state a device needs to accomplish a specific task.

The use of FBlocks makes the control device openly programmable and easily reconfigurable. IEC 61499-compliant devices can easily interface with one another, thus
providing seamless distribution of different tasks across different devices [177, 178]. Users may create their own programme using standard FB\textit{Blocks}. Thus the IEC 61499 architecture enables encapsulation, portability, interoperability and configurability.

### 5.2.2. IEC 61499 Standard and Function Blocks

IEC 61499 consists of four parts. As of July 2004, the parts 61499-1 “Architecture” and 61499-2 “Software Tools Requirements” [179] were voted and approved as IEC Standards and published in 2005. Part 61499-3 “Application guidelines” was approved as a Technical Report [180]. A voting draft of 61499-4 “Rules for compliance profiles” [179] was approved in 2005.

There are three standard classes of FB\textit{Block} defined in IEC 61499: (1) basic FB\textit{Blocks}, (2) composite FB\textit{Blocks} and (3) service interface FB\textit{Blocks}. Each FB\textit{Block} has a set of input and output variables. The input variables are read by the internal algorithm when it is executed, while the results from the algorithm are written to the outputs.

The basic block encapsulates algorithms and has an execution control chart that gives this block type the flexibility to model many different components. As an example, defining basic FB\textit{Block} type X2Y2 is shown in Figure 5.1. In basic FB\textit{Blocks} of IEC 61499, a state machine, namely Execution Control Chart (ECC), defines the reaction on input events of each block. The reaction can consist of the execution of algorithms that compute some values as functions of input and internal variables, resulting in the emission of one or several output events. In Figure 5.1 (right side) the ECC and an algorithm are shown. The state REQ has one associated action that consists of calling the algorithm REQ followed by emission of the output event CNF. The algorithm computes $\text{Out} = X^2 - Y^2$.

A composite FB\textit{Block} can encapsulate a network of multiple blocks (both basic and composite), interconnected by external data sources. This is a combination of several basic
FBlocks that look just like a ‘normal’ FBlock from the outside. An example is given in Figure 5.2, where the same $X^2 - Y^2$ function is implemented as a network of three FBlocks, doing addition, subtraction and multiplication respectively. This network can be encapsulated in a composite FBlock with the same interface as the FBlock X2Y2 from Figure 5.1. The possibility of including composite FBlocks with other composite FBlocks enables a hierarchical system description. This is useful for defining multi-layered architecture.

![Figure 5.2 Implementing $X^2 - Y^2$ as a Network of Function Blocks](image)

In this research, the Cloud-based service architecture developed was layered, with different layers that are responsible for service supervision, service execution and data processing. The separation of the functions into layers enables flexibility of the system. The FBlock-based methodology also enables CMService packages to be tightly integrated with the ST, forming a reliable virtual service combination. Before execution, the service can be validated by either SCM or service verification software.

In IEC 61499 architecture, the function performed by the system is specified as an application, which may be implemented in a single device or be distributed among several devices. The application consists of a network of FBlocks connected by data and event connections. The control system is specified as a collection of devices, interconnected and communicating with each other by means of one or more communication networks.

The IEC 61499 FBlock specification is the new standard for meeting requirements based on an explicit event-driven model and also provides data flow and finite-state automata-based control [181]. The algorithms can be written in either high-level programming languages (for example, Java) or in the IEC 61131 languages for programmable controllers (for example, Ladder Diagrams and Structured Text). Previous research on FBlocks proves they can be used as an enabler to encapsulate process plans, integrate with a third-party dynamic scheduling system, monitor a process plan during execution, and control machining jobs.
They are also suitable for machine-level monitoring, shop-floor execution and service control in the CManufacturing context.

An example of an FBlock type conformant to IEC 61499 is shown in Figure 5.3. The example is separated into an upper and a lower part. In the upper part, the event inputs and outputs are shown. The data inputs and outputs are shown in the lower part. In the machining example, in particular, ‘Axis’ is modelled as an input variable and connected to ‘Execute’ like all other data inputs because the FBBlocks of IEC 61499 do not have specific variables.

![Figure 5.3 Example of an FBlock Type Conformant to IEC 61499](image)

In engineering disciplines, especially within time-driven systems, software languages are often based on FBBlocks. Examples are languages for Programmable Logic Controllers (PLC) or simulation environments such as Matlab/Simulink. Many commercial software tools in the control industry also use FBBlocks as programme organisation units. Despite minor differences, the concept of FBBlocks is the same in all the FBBlock-oriented languages considered. One key benefit of FBBlocks is their modular design. Therefore, this research made use of the modularity of FBBlocks. Function blocks have previously been used to model and control other mechanical systems. By combining FBBlocks with mobile agents, IEC 61499 can utilise the modularity and interconnection properties of virtual FBBlocks to allow for rapid integration of MResources in the CManufacturing environment.

A generalised FBBlock consists of input variables, output variables, through variables, internal variables and an internal behaviour description of the FBBlock (Figure 5.4). Input variables can only be written from outside an FBBlock. From inside, they can only be read. Output variables can be read and written from inside an FBBlock and only read from outside. Through
variables are special shared variables. If variables of different FBlock instances are connected, they can all access the variable connected to the first input of the chain. They are often called In-Out-variables. If their data-type matches, output variables can be connected to input variables by a connector. This is similar to a connection of ports with matching protocols.

Unlike simple functions, FBBlocks have internal state information that carry out the execution of FBBlock instances. The internal behaviour can be driven by continuous (as in Simulink) or discrete time (as in PLC), or can be event-driven (IEC 61499). Common to all FBBlocks is that their interface variables (input, output and through) continuously provide data values. Communication with other FBBlocks can only be done by assignment of data values to interface variables. This communication model is opposite to the one of object orientation, where objects communicate by message exchange.

5.3. Software Agent

Agents are welcome in manufacturing because they help to realise important properties, such as autonomy, responsiveness, redundancy, distributedness and openness [182]. Agent-based applications support manufacturing domains, e.g. engineering design, process planning, process control/monitoring and enterprise integration [183-186]. NIST has developed a prototype agent-based platform supporting the integration of manufacturing planning, predictive machining models and manufacturing control [187]. It was proven by Newman and Nassehi that agents allow integration approaches to function in a geographically distributed domain, in particular information exchange among the various CAx systems [38]. Software agents have shown strong capabilities in data exchange, system integration and resource management/sharing [61, 67, 79, 85, 188].

A software agent is a component of software and/or hardware that is capable of acting exactly, in order to accomplish tasks on behalf of its user [189]. The typical characteristics for agents are summarised as autonomous, cooperating and learning ability. Four types of
software agent are categorised, i.e. collaborative agents, collaborative learning agents, interface agents and truly smart agents (Figure 5.5).

An agent is a cluster of programmes that is able to achieve objectives independently. The key notions that distinguish agents from other programmes are reaction to the environment, autonomy, goal-orientation and persistence [190]. In past decades, different types of agent have been developed and implemented in the industry. Intelligent agents are able to complete tasks with artificial intelligence to some degree, e.g. learning and reasoning; autonomous agents are capable of changing or improving the way in which they achieve their assignments; distributed agents can be implemented on physically distinct computer multi-agent systems that contain agents, which can communicate with each other and achieve tasks as a whole; and mobile agents that can relocate their execution process on different devices or applications. Some common characteristics of agents are summarised as follows:

- object-oriented: agents act on the behalf of the designer or user in order to meet a particular task
- autonomous: agents can control their internal state and behaviour in different environments
- interactive: agents are able to interact with the environment and other agents
- adaptive: agents are capable of adjusting themselves to meet a changing environment

There are three typical approaches to developing agents, i.e. transducer, wrapper and rewriter as in Figure 5.6 [191]. A transducer mediates between an existing programme and other agents. The transducer accepts messages from other agents, translates them into the software’s application’s native communication protocol, and passes those messages to the software. It accepts the software programme responses, translates them into requested

Figure 5.5 A Partial View of an Agent Typology [189]
Chapter 5 – Service Application Cloud: Virtualised Manufacturing Capability

formats, and sends the resulting messages to other agents. This approach has the advantage of requiring no knowledge of the programme other than its communication behaviour. It is, therefore, especially useful for situations in which the programme code is unavailable or too delicate to modify, for example in commercial software applications. This approach also works for other types of resource, such as files and people. It is possible to write a programme to read or modify an existing file with a specialised format, e.g. STEP, thereby providing access to that file. Similarly, a GUI can be provided to a user, allowing the user to interact with the system in a specialised graphical language.

Figure 5.6 Three Types of Software Agent

A second approach to dealing with legacy software is to implement a wrapper, i.e. injecting code into software to allow it to communicate in agent-based language. The wrapper can directly examine the data structures of the programme and modify those data structures. This approach has the advantage of greater efficiency than the transduction approach. It also works for cases having no inter-process communication ability in the original programme. However, it requires the source code for the programme to be available.

The third and most drastic approach to dealing with legacy software is to rewrite the original programme. The advantage of this approach is that it may be possible to enhance its efficiency or capability beyond what would be possible in either the transduction or wrapping approaches. The best examples of this agent approach actually come from the manufacturing domain. Many automated design programmes work on completion before communicating with other programmes. For example, the output of a CAD application is passed as input to a CAM application. However, if there is any change or correction that occurs in the manufacturing department, the outputs of CAM cannot be read back by the CAD software, let alone updated. The common solution is to recreate the product design or apply these changes manually, which leads to a huge cost in time and effort. Recent work in concurrent engineering suggests there is much advantage to be gained by writing programmes that communicate partial results in the course of their activity and accept partial results and
feedback from other programmes. Siemens’ Teamcenter provides software packages throughout the CAD-CAM-CNC chain. When there are modifications made in the manufacturing department, the CAD team can be notified by the system and thereby enabled to deploy these modifications directly.

Rewriting the applications based on the same kernel and format provides strong system robustness and reliability to the environment. Implementing a proprietary format is one of the data-centric interoperability solutions discussed in Chapter 2. The deployment of software applications within one organisation may be feasible, but when it comes to multiple tiers of suppliers, contractors and retailers, it is not realistic to force all the stakeholders to use the software tools in the same kernel. The wrapper agent provides attractive advantages, e.g. fast response and efficiency. However, it requires the source code from software programmes to build internal connections between the software and an agent programme, which is against the expectation and intelligent property rules of the software vendors. The transducer agent provides advantages, i.e. flexibility and easy implementation. Hence, it is necessary to develop a transducer-type software agent that partially owns a wrapper agent’s features without interference with the software applications. Therefore, the transducer agents are joined with the concept of FBlocks, forming a new integration terminology called Virtual Function Block.

5.4. Virtual Function Block

The concept of the VFB was developed to meet the integration requirements of CManufacturing. VFB is defined as a programme unit that is able to accomplish a specific task independently. VFBs are conducted by transducer agents that contain internal states and process algorithms. Transducer agents are able to connect software tools by controlling their data and event input/output. Data and event states are therefore supervised by VFBs without the injection of codes into the software resource programme. As mentioned above, MResources are recognised and modelled by standardised models. The resources then need to be integrated into the Cloud as ready-for-use MCapability modules. With the help of VFBs, physical and non-physical resources are integrated by transducer agents that offer partial wrapper functionalities.
5.4.1. Cloud-Module-Agent Structure

ICMS aims to integrate software/hardware applications based on service requests from users. SCM is capable of handling requests from users such as task objects, functionality and input/output requirements, and then organising a series of software/hardware services, which form a “Request-Find-Combine-Provide” loop. As depicted in Figure 5.7, the Service Application Cloud was designed in a three-tier architecture from the implementation viewpoint. Under the Cloud tier at the top, modules realised via VFBs are provided to aid data and service flows. On the bottom tier of the system, MResources are packaged by individual agents acting as plug-ins in ICMS. The packaged resource suites are the basic VFB units in the system fulfilling requests from users.

Figure 5.7 Three-tier Architecture

As soon as the ST is generated by the SCM, the Service Application Cloud carries out one or several tasks, according to the user request. MResources are defined as MCapability Modules such as CAD file converters, CAM software suites and machine tools. Such modules were developed to be self-contained applications that can be executed autonomously. These applications are controlled at event/data level. New tools and systems can be included through newly encapsulated modules. Thus platform extendibility is guaranteed.

5.4.2. Agent-based VFB

In ICMS, agents are in charge of monitoring data/event flow and controlling the activity of modules. Combined with the FBlock concept, software agents are utilised to conduct self-contained function units i.e. VFBs. By providing an explicit event-driven model for data flow and finite state automata-based control, Function Blocks provide several features such as
robustness and modularity. A VFB can be re-used in a wide range of applications and it is also easy to implement.

After MResources are packaged using a VFB (Figure 5.8), individual VFB modules can work autonomously and are considered to be “black boxes”. Usually, there is no need to modify the module after it has been defined for the first time. Mechanisms to trigger the data flows are also integrated into the VFBs. The running of each FBlock is controlled by event_in and event_out variables defined in the architecture. When the output files are saved, an event_out variable is sent to the SCM and the next step is activated. In practice, a module can be opened by the event_in variable and it is available for users through the Human-Machine Interface.

![Figure 5.8 MResource Encapsulated by Virtual Function Block](image)

Inside each Module, the MResource is packaged by the Wrapper Agent, in which an agent core, a pre-processor and a post-processor are encapsulated. Based on the description of ST, different pre-/post-processors are selected and integrated for different purposes. In practice, processors are modules pre-defined or pre-stored in the Cloud before an MResource is encapsulated in the Cloud environment. Thus a processor can be a cluster of programmes, or even another MCapability module. For example, before the machining data is input into the CNC machine, the tool path and setup may need to be validated. In this case, the tool path validation software is assigned by SCM to play the role of pre-processor in this VFB. The software is firstly encapsulated as a validating VFB module, and then integrated into another VFB as the pre-processor. In this case, VFB enables the function of including one VFB within another, forming a composite VFB structure (Figure 5.9).
Deployment of VFB changes the means of interactions between resources and the Cloud. Traditionally, CComputing applications are connected with the central server directly (Figure 5.10). In web-based manufacturing approaches, communications between the server and the resource operator/administrator are also unavoidable. VFB provides an integration method that is a self-controlled programme running near the data/event source. It improves a resource-Cloud paradigm via a better, more efficient and flexible mode of communication. VFBs are responsible for the dynamic monitoring of resource performance, and downloading/uploading service input and output. The VFB enables a smart network with smart messages. VFBs are static entities in the Cloud structure that are able to perform tasks autonomously and synchronously. They can communicate with other agents dynamically.

The service description template can be either simple or complex, and executed by lightweight VFB units. With these agent-based VFBs, customised and distributed services can be provided instantly.
5.4.3. MTConnect to Integrate Hard Resource

When a hard resource (e.g. a milling machine tool) is requested to be integrated within the VFB, the relevant applications can be selected as processors; for instance, process retrieval module i.e. MTConnect [192]. MTConnect [193] is an open standard defining protocol and interchange format to facilitate communications between devices in manufacturing systems. It enables data acquisition capabilities from a shop to multiple locations. Assisted by software agents, MTConnect can provide a plug-and-play environment for hard MResources in the Cloud. MTConnect can be defined as an open protocol and XML-based standard for data integration, which can act as an enabler for high-level standards [194]. Therefore, it can be seen that the integration of international standards (STEP) with MTConnect enables an interoperable approach for accessing and handling machining data across different locations. Figure 5.11 shows an overall picture of how this integration can be achieved, to support manufacturing system interoperability. Its architecture can be easily deployed and retrofitted.
to existing machines, providing particular flexibility and portability functions in the manufacturing environment.

MTConnect is composed of a few basic conceptual parts, which can be divided as [195]:

- **Header**: every MTConnect response must contain Header Protocol-related information as the first element of an MTConnect XML Document before it is sent back to an application.
- **Components**: MTConnect has four distinct components, namely Device, Adapter, Agent, and Client, which collectively act as the backbone of the communication standard. The Device refers to the components (e.g., controllers, sensors and machine tools). It is responsible for providing monitored data. For example, for a three-axis milling machine, a device structure that has a power supply, controller, three linear axes and a spindle can be modeled as illustrated in Figure 5.12.
- **The Agent needs to be capable of delivering data associated with each component to an application**
- **DataItems**: the description of the pieces of information delivered by the Agent is referred to as DataItems. It must specify the type of data being collected, the name of the data item and the category of the item. There will only be one category for each type, but it must be included to aid the application in determining a location for the data.
stream. The data item may specify a source sub-element to provide the native name for the data feed.

Therefore, MTConnect provides powerful and standardised tools to monitor MResources and acquire data in a distributed environment. In this research, it is especially suitable for ongoing machining process. In practice, position, acceleration and vibration data acquirments for milling process has been implemented in the University of Auckland [196]. It is particularly suitable for the CManufacturing paradigm. In the ICMS context, MResources, in particular hard resources, are located globally among suppliers, contractors and other Cloud participants. With the help of MTConnect, remote data captured in real-time or near-real-time (i.e. current speed, position data, temperature data, programme block, etc.) can be utilised by SCM or other CMService applications, e.g. maintenance diagnostic services, management production information services, CAM services, etc. Implemented via agent technology, MTConnect is suitable for agent-based VFB methodology naturally. Encapsulated as a part of
VFB processors, MTConnect provides monitoring and data interoperability to the CMServices.

5.4.4. VFB and CMServices

In practice, SCM makes the optimised choice of a VFB as a processor based on user requests. The agent core monitors and controls the activity of the whole module and reacts to the control statement issued by the SCM. MResources are packaged by VFBs, thus forming MCapability modules. The VFB-based modules are organised according to service specifications in a Virtual Service Combination. As shown in Figure 5.13, the CMService is involved in several stages of process, based on the definition in the ST document. Each process can be treated as a series of events and activities that fulfil user demands from the perspective of integration. With the help of VFBs, event triggers are detected and controlled by wrapper agents. Thus, MResources are integrated within VFBs as function modules.
During software integration, software events can be assisted by a software agent at the command level. As mentioned above, agents can be developed to wrap the software application without affecting the codes, via system commands e.g. launching, shutting down, data transmission, etc. For example, it is assumed that a Cloud user requires work from several machining features. Based on the service query description, Creo Parametric is allocated by SCM, since it provides appropriate CAD and CAM capabilities. The original product document is maintained in a format compatible with the ISO 14649 standard. It cannot be read by Creo directly. Thus a composite VFB is organised to fulfil user request.
The VFB conducts the process with two data translators that are allocated as pre-processor and post-processor. Meanwhile, the Creo application and an agent core are determined to be centre of the module. The data and event flow of this Cloud service is summarised in Figure 5.14. Firstly, the product document (testV1.stp) is remotely retrieved from the Product Database. The translator VFB is then launched as the pre-processor that converts the .stp file (data-in) into a .prt document (data-out) compliant with Creo Parametric. When the test1.prt is output, it triggers the event-out state of the pre-processor, the translator is shut down and the document is delivered to the Creo programme as input data. Likewise, the user is able to work with the 3D features in the Creo environment. After he or she finishes the task, an output testV2.prt is saved on the hard driver, which is stated as the output event. Then the core agent detects this event and terminates the Creo process. This output event triggers the algorithm of the post-processor, which takes it as its event-in. The second translator is launched to convert the testV2.prt file back into .stp format. After the testV2.stp document is output, it triggers the VFB. The post-processor is terminated and the output document is archived back into the Product Database. Then the entire Creo-centric service is completed. The VFB is shut down after an event-out trigger is sent to SCM. SCM then reviews the ST document and processes the service to the next stage.

Figure 5.14 Creo VFB Module
When hard resources are integrated via VFBs, the procedure is similar. A pre-processor can organise essential service input, e.g. process plan, tool path or raw materials. Hard resources can be connected via the network to transmit data and event triggers. MTConnect is proven capable of streaming data from the shop floor to remote databases. In the hard resource context, a number of tools are capable of identifying and integrating resources. Web-based monitoring is a process within a distributed system for collecting and storing state data, to support performance management of the facilities or applications. It provides reliable performance information on equipment, e.g. cutting speed, feed rate, vibration, etc. Working with MTConnect, it enables the integration of physical applications in the Cloud. The wireless sensor network consists of distributed autonomous sensors to monitor physical and environment conditions, and cooperatively pass the data over the network to a main location. In the ICMS environment, it is particularly useful to transform resource information (e.g. location, temperature and status) to the Storage Cloud that is the main data maintenance station. Moreover, RFID offers a non-contact method that uses radio-frequency electromagnetic files to transfer data from a tag attached to an object, for the purposes of automatic identification and tracking. In particular, it is useful in a distributed and network-based manufacturing environment such as ICMS. RFID is capable of product and progress tracking through an assembly line. It also offers remote methods of inventory and transpiration management. In the CManufacturing context, resources and products are distributed globally. RFID provides accurate status information of the products, supporting the supply chain with flexible and trustworthy information.

Therefore, both soft and hard resources can be integrated in the Cloud with the help of VFBs. The next step is to standardise VFB behaviour and maintain these resources in the Cloud. A data model that describes VFB specifications was developed in EXPRESS language.

### 5.5. VFB Data Model

This said, ICMS contains multiple databases in the Storage Cloud. One of them is Service Database, which maintains the available service applications in the Cloud and their VFBs. To describe a VFB and integrate it in the Cloud environment, a VFB data model has been developed in EXPRESS language. When a Cloud service template is generated, the relevant MCapability and MResource are assigned as well. Note that one single service can require multiple MCapabilities integrated across the Cloud. Thus a CMService task can be supported by a number of MCapability modules. Likewise, one VFB module may have multiple
versions in different domains, so much so that both multi-functionality and reusability are guaranteed in ICMS.

As an attribute of MCapability, the VFB module is connected that describes the correct function unit to execute a service task. Hence the MResource, MCapability and its VFBs are integrated around ST to support a specific CManufacturing task. VFB_Module entity has two subtypes, i.e. Hard_VFB and Soft_VFB. Hard_VFB entity describes the application modules supported by a hard resource, and Soft_VFB refers to the modules implemented via soft resources. These two subtypes inherit all the description entities of VFB_Module. The top-level of VFB_Module’s attributes is VFB_View_Definition, which is further represented via entities ID, Name, Provider, Description, Time_Stamp, Security_Setup, VFB_Version, View_Definition, Event_Flow, Data_Flow and Data_Object (Figure 5.15).

**Figure 5.15 VFB Data Model**

Entity ID keeps the unique identifier of a VFB. For different combinations of agents and processors, the identifier is different too.

Entity Provider offers the provider information of the components within a VFB module. It is directly connected to the SProvider data model mentioned.

Entity VFB_Version describes the unique version identifier and version number for each VFB module. As mentioned above, different configurations and arrangements are
documented in different VFB data descriptions. The development or change of VFBs is thus coordinated via version controlling.

Entity Security_Setup provides the safety configurations that cooperate with the system Firewall. It contains the protection rules, access setups and privacy information for each VFB module.

Optional entity Name maintains the general title of a VFB. The Name entity is visible to the user.

Entity Event_Flow describes the event triggers of a VFB. Its attributes event_in and event_out define the events (e.g. production completion, monitor launching, etc.) that initiate or precipitate functional reactions.

Entity Data_Flow defines the input and output data of a VFB module. The data acquired from or achieved in the Storage Cloud is kept here. Via data_in and data_out attributes, the complete data (including format, version and property information) is recorded and saved in the database.

Entity Time_Stamp records the exact time when the VFB was launched and terminated. It is also utilised in the data synchronisation and exchange mechanism explained in the next chapter.

Optional entity Description maintains the functionality statement as well as the modelling entities mentioned above. VFB descriptions can be stated in general terms, e.g. text, figure and attachment.

Thanks to the VFB data model, all the data related to a service package are maintained, along with the product and project data. This said, the development history for each project becomes traceable. The information is integrated in a unified syntax in the CManufacturing paradigm.

5.6. Recap

To implement a CManufacturing system, it is necessary to integrate hard and soft resources to support CMService. An interoperable mechanism is needed to encapsulate these two types of resource. Inspired by IEC 61499 and software agents, an integration terminology was
developed, namely, Virtual Function Block. VFB provides a “wrapping” capability to manage and control manufacturing applications.

For soft resources, e.g. software tools, the resource can be controlled without the need to change the internal codes or programmes. VFB provides a wrapping functionality by controlling the data and event flow of the software without a real injection of a programme into the software. It avoids the risk of infringement of intellectual property rights during system integration and guarantees the interoperability of the system. For hard resources (e.g. machine, monitor and robots), VFB provides portable protocols of signals and data (e.g. MTConnect and STEP), without changing the original arrangements of devices. In summary, the transducer agents provides “wrapping” functionality to control the soft resources without the need of changing the original codes and programmes, while physical FBlock provides portable protocols of signals and data. In this way, both physical and nonphysical resources are integrated as modularised VFBs at the functionality level. The configurations at the operation level are kept as they were before.

Resources can be easily manipulated by their input and output events. Meanwhile, the data_in and data_out mechanism deal with the data flow through a VFB. VFB provides a generic model to merge different manufacturing resources together. After the VFB modules are implemented in the Cloud, it is necessary to develop a mechanism to cope with data exchange issues. As discussed before, a heterogeneous data environment leads to huge cost and effort in the manufacturing industry. Data interoperability needs to be guaranteed within a CManufacturing environment. In the next chapter, a novel data exchange philosophy is revealed, namely, “Data Packet”.
Chapter 6. Storage Cloud: Product Data Exchange Mechanism

This chapter focuses on a data exchange mechanism based on STEP and STEP-NC standards that were developed by the ISO. The first part of the chapter includes the STEP background and objectives. Following that, the extension of STEP called STEP-NC is discussed. Then a novel data exchange mechanism is proposed, based on these data structures. The mechanism aims to provide “the right data to the right people in the right manner” in the Storage Cloud of the ICMS system. The major work in this chapter was presented at the DET2011 conference [197]. Extended development and results were then summarised in paper [198].

6.1. An Overview of the STEP Data Model

The STEP or STandard for the Exchange of Product model data is an International Standard for computer-interpretable representation and exchange of product data, as defined by ISO 10303. The main endeavour undertaken by the international effort in developing this standard was to provide a mechanism capable of describing product data throughout the lifecycle of a product, independent of any particular system. The nature of this description makes it suitable not only for neutral exchange, but also as a basis for implementing and sharing product databases, and archiving [26].

6.1.1. STEP – ISO 10303

The first major release of proposed STEP documents occurred in 1988, when a large set of models was assembled into a single Integrated Product Information Model (IPIM). The documents from the IPIM were adopted as initial drafts of ISO standards at a Sub-Committee 4 (SC4) meeting in Tokyo in late 1988. By 1989, STEP had focused on the concept of AP, a sub-set of STEP that would be necessary for a specific industrial use and could be implemented and subjected to conformance testing. The first version of STEP became an ISO standard in 1994 and companies such as General Electric, Boeing and General Motors began announcing commitments to use STEP in 1995. In 1994/95, the ISO published the initial
release of STEP as International Standards (IS) with ISO 10303 Parts 1, 11, 21, 31, 41, 42, 43, 44, 46, 101, AP-201 and AP-203.

In the second phase, the capabilities of STEP became widely extended, primarily for the design of products in the aerospace, automotive, electrical, electronic and other industries. This phase ended in 2002 with a second major release, including ISO 10303 APs such as AP 202, AP 209, AP 210, AP 212, AP 214, AP 224, AP 225, AP 227 and AP 232. As of June 2008, the SC4 website (TC184/SC4, 2008) listed twenty-three application protocols that had become international standards. New editions of the previous monolithic APs on a modular basis have been developed, which consist of only AP 203, AP 209 and AP 210.

In early July 2010, the development of a new Automotive and Aerospace Application Protocol of the future, denoted AP 242, was initiated. In cooperation with various international bodies, as well as representatives of industry, the AP 203, AP 214 and other APs in mechanical design were combined in a compatible way under the designation STEP AP 242. The first edition of AP 242 was expected to be technically complete in 2012, and will contain major updates in the area of Geometric dimensioning and tolerancing, and kinematics. According to recent ISO development updates, the AP 242 data was implemented to describe machine tool performance, including positional and kinematic data of a 3-axis Mazak machine in KTH, Sweden.

A scenario of how STEP can be implemented in today’s manufacturing environment can be illustrated using the following example: an automobile engine designer working with a commercially available CAD system designs an engine block. The AP 203 representation is saved in a STEP data file using Part 21 of STEP. The engine block design is sent to a manufacturing plant by sending the STEP Part 21 file for the design. At the manufacturing plant, a manufacturing engineer using a CAD system from a different vendor tells the CAD system to read the STEP file. This is possible because the second CAD vendor has also implemented STEP AP 203. The system has a module that can read the STEP file and build a representation of the design in the second CAD system’s native format. With the design now residents in the CAD system, the manufacturing engineer discovers how to manufacture the engine block. If the manufacturing engineer wants to suggest a change in the design (and the second CAD system includes a STEP output module), he or she can have the CAD system write a STEP AP 203 Part 21 file and send it back to the designer. It is also possible to use
STEP to communicate design information at the feature level (AP 224) and manufacturing information at the operation level (AP 238) [199].

The data exchange, from design to manufacturing, enabled by STEP is illustrated in Figure 6.1. STEP AP 203 represents the 3-D model of a CAD drawing that contains the geometric representation data, which is first translated into manufacturing features in a process plan defined by STEP AP224. The machining feature definitions are used as inputs to macro process planning applications defined by AP 240 for machining, AP 223 for casting and AP 229 for forging. Subsequently, micro process planning for machining, AP 238, and inspection, AP 219, will be carried out for the abovementioned application processes. As a result, the need for data conversion is eliminated [200].

Figure 6.1 Design-Manufacturing Data Exchange Enabled by STEP
6.1.2. Objects of STEP

The role of the STEP standard (ISO 10303) is generally to act as a data backbone for product information. The main aim is to specify a form for unambiguous representation and exchange of computer-interpretable product data throughout the life of a product, in which the form is independent from any particular computer system. This form enables consistent implementation across multiple applications and systems. This international standard permits different implementation methods, not only for exchange purposes, but also for storing, accessing, transferring and archiving product data [26].

The descriptions of these attributes are paraphrased as follows [201]:

- Product Data Exchange: transfer of product data between a pair of applications. STEP defines the form of the product data that is to be transferred between a pair of applications. Each application holds its own copy of the product data in its own preferred form. The data conforming to STEP is transitory and defined only for the purpose of exchange.

- Product Data Sharing: access to and operation of a single copy of the same product data by more than one application, potentially and simultaneously. STEP is designed to support the interfaces between a single copy of the product data and the applications that share it. The applications do not hold the data in their own preferred forms. The architectural elements of STEP may be used to support the realisation of the shared product data itself. The product data of prime interest in this case is the integrated product data, not the portions used by particular product data applications.

- Product Data Archiving: storage of product data, usually long term. STEP is suitable to support the interface with the archive. As in product data sharing, the architectural elements of STEP may be used to support the development of the archived product data itself. Archiving requires that the data conforming to STEP for exchange purposes be kept for use at some other time. This subsequent use may be through either product data exchange or product data sharing.

6.1.3. STEP Architecture

The STEP standard is separated into many series of parts, which covers implementation architectures, conformance testing, resource information models and application protocols. Each series has a unique function and may have one or more interrelated parts. These parts
are called Description Methods, Implementation Methods, Application Protocols, Information Models and Conformance Tools. Figure 6.2 illustrates the structure of the STEP standard.

![Figure 6.2 Structure of STEP standard [26]](image)

The parts of STEP may be grouped according to a series. The parts are numbered so that all parts of the same series fall in the same number range. The series and the numbering scheme are listed as follows:

- Overview and Fundamental Principles – Part 1: this is a single document giving an overview of STEP and an exposition of its fundamental principles
- Description Methods – Parts 11 to 19: these cover the information modelling language EXPRESS and its graphical form, EXPRESS-G, that were introduced in Chapter 4.
- Implementation Methods – Parts 21 to 29: these cover methods of representing data that have been modelled in EXPRESS
- Conformance Testing Methodology and Framework – Parts 31 to 39: these give the general concept of conformance testing as well as actual test methods and requirements on testing labs and clients
• Application Protocols – Parts 201 to 299: these are the Parts intended for implementation in industry. As described in more detail below, each application protocol includes several documents
• Integrated Generic Resources – Parts 41 to 59: these are EXPRESS information models of widely useful specific subject domains, such as geometry, topology and tolerances
• Integrated Application Resources – Parts 101 to 199: these are EXPRESS information models of more narrowly focused specific subject domains

An AP can be built by including a large number of application modules. Using application modules is a more recent architectural approach than using Application Interpreted Constructs, and may replace application-interpreted constructs.

6.1.4. Application Protocols

The STEP standard is divided into many Application Protocols belonging to the ISO 10303 family of standards. Each protocol defines a data exchange standard for a defined family of products at a defined stage in its lifecycle. An application protocol is focused on a particular application domain. When the AP concept was first introduced in STEP, an AP had three parts:

• Application Activity Model (AAM) — a model of the activities and data flows of the application
• Application Reference Model (ARM) — a model of the data needed for a particular application
• Application Interpreted Model (AIM) — an encoding of the ARM in terms of the STEP integrated resources which is the model intended for implementation in systems that use STEP.

The AAM is provided as an aid in understanding the scope and information requirements defined in an application protocol. In other words, AAMs are presented as a set of figures that contain the activity diagrams and a set of definitions of the activities and their data. AAMs are built using the Integration Definition for Function Modelling (IDEF0), which is a graphical method of modelling activities and data flows. Activities are represented as boxes, while data, actors and constraints are represented by arrows. An example of the first page of an AAM is shown in Figure 6.3.
In the IDEF0 approach, an aggregated model is first built, to show the big picture with three to six activities. Then one or two rounds of refinement are performed with each activity at an upper level that is expanded into an entire page at the next level down. Once the AAM stage is completed and an ARM has been built, the AAM plays no further role [202].

The application reference model of an application protocol is a model of the data needed for a particular application. The model is given using the terminology of the application so that the model can be understood by participants of the application (who are involved in the development of the model). The process of building an ARM usually includes workshops, at which domain experts decide what entities should be defined and what their attributes should be. ARMs may be written in EXPRESS, EXPRESS-G or IDEF1X. The modelling language is less important than the content.

The application-interpreted model of an application protocol is an EXPRESS model of the information in an ARM, and is encoded in terms of the STEP integrated resources [203]. The data structure coded using mapping tables, the format of which is formally defined and is uniformed across STEP. The typical example of this is STEP AP-203. For most APs, however, the encoding is mostly done using Part entities. The AIM that results from the
encoding is usually very complicated and not user-friendly and can only be understood by a
STEP expert.

6.1.5. Part 21 Implementation Methods

Each implementation method included in ISO 10303 is specified by a mapping from the
EXPRESS language into the formal language used for the method. The mapping is
independent of the application protocol. The EXPRESS language does not define any
implementation method. Therefore, additional implementation methods are defined to
describe STEP instances for building product exchange models, for example ISO 10303 AP-
238 models. There are several implementation technologies available:

- a product model specific file format called Part 21 physical file [170]
- the variety of programming language bindings that allow an application programmer
to open a data set and access values in its entity instances. Bindings have been
developed for C, C++ and Java [29-32]
- three methods for mapping the EXPRESS defined data into XML, described by Part
28 Edition 1 [171]
- the XML Schema-governed representation of EXPRESS, described by Part 28 Edition
2 [204].

Given that EXPRESS language does not define any implementation method for building
product exchange models, STEP part 21 is the first implementation method that defines the
basic rules for storing EXPRESS/STEP data in a character-based physical file. Its aim is to
provide a method so it is possible to write EXPRESS/STEP entities and transmit those
entities using normal networking and communication protocols, such as File Transfer
Protocol (FTP), e-mail and Hyper Text Transfer Protocol (HTTP). A Part 21 file does not
have any EXPRESS schemas included. It only defines the relationships between entities that
are defined by external EXPRESS schemas.

An example of the beginning part of a Part 21 file is shown in Figure 6.4. Each entity
instance in a Part 21 file begins with a unique Entity ID and terminates with a semicolon
(“;”). The Entity ID is a hash symbol “#” followed by an integer and has to be unique within
the data exchange file. The Entity ID is followed by an equal symbol (“=”)) and the name of
the entity that defines the instance. The names are always capitalised because EXPRESS is
case insensitive. The name of the instance is then followed by the values of the attributes listed between parentheses and separated by commas.

```plaintext
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('ISO 14649-11 EXAMPLE 2', 'COMPLEX PRORGRAM WITH VARIOUS MANUFACTURING FEATURES'), '1');
FILE_NAME('EXAMPLE2.STF', '2012-10-02', ('JOCHEN WOLF', 'YONG TAK HYUN', 'C.SAKAMOTO' 'XIN
Vincent Wang'), ('WZL, RWTH-AACHEN', 'KOMATSU'), $, 'ISO 14649', $);
FILE_SCHEMA(('combined_schema'));
ENDSEC;
DATA;
#1= PROJECT('EXECUTE EXAMPLE2', #2, (#7, $, $, $);
#2= WORKPLAN('MAIN WORKPLAN', (#4, #5, #6), $, #14, $);
#4= WORKPLAN('WORKPLAN ROUGHING', (#17, #18, #19, #20, #21), $, $, $);
#5= WORKPLAN('WORKPLAN DRILLING', (#23, #24, #25, #26, #27, #28, #29, #30, #31, #32, #33, #34, #35 , #36, #37), $, $, $);
#6= WORKPLAN('WORKPLAN FINISHING', (#39, #40, #41, #42, #43, #44, #45, #46, #47, #48, #49, #50, #51, #52, #53), $, $, $);
#7= WORKPIECE('PART 2', #13, 0.01, $, $, $, (#9, #10, #11, #12));
#9= CARTESIAN_POINT('CLAMPING_POSITION1', (25., 25., -20.));
#10= CARTESIAN_POINT('CLAMPING_POSITION2', (205., 25., -20.));
#11= CARTESIAN_POINT('CLAMPING_POSITION3', (25., 155., -20.));
#12= CARTESIAN_POINT('CLAMPING_POSITION4', (205., 155., -20.));
```

**Figure 6.4 Example of Part 21 File**

It is expected that a system that reads the file will find values for these attributes, based on the data that is provided in the file. Comments in the header indicate what each header item means. The special token "$" is used to represent an object whose value is not omitted. The special token "*" is similar to ":$" except that the value can be derived from other values, according to rules given in the EXPRESS schema.

### 6.2. STEP-NC

STEP-NC was developed to provide a data model for a new breed of intelligent CNC controllers. The STEP-NC data model provides standard data requirements for machining processes associated with CNC machining. The STEP-NC standard is an extension of STEP and allows for connections between STEP-based CAx and CNC. The ARM of STEP-NC, that is ISO 14649, is made up of several Parts. In general, STEP-NC is the application of
STEP methods to numerically controlled machines, representing a common standard specifically aimed at part programming, making the goal of a standardised CNC controller and a part programme code-generation facility a reality.

The first set of Parts of ISO 14649 became an international standard in 2004. Several Parts of ISO 14649 were also adopted as conceptual models by the ISO team, to develop the AIM of STEP-NC, which was ISO 10303-238 (or STEP AP-238) in the early 2000s, and AP-238 was published in 2007.

There are two different subcommittees within the ISO: ISO TC 184/SC1 and ISO TC 184/SC4. They have been actively working on the development of STEP-NC standards. SC1 focuses on the control of machines, while SC4 focuses on industrial data. Since numerical control programmes for machining a product are product data, there is a natural overlap between SC1 and SC4.

ISO TC184/SC1 has the intent that STEP data representation methods are used with ISO 14649, since the Parts of ISO 14649 include examples using STEP Part 21 files from them. File exchange for industrial use can be accomplished quite well by using Part 21 files based on ARM type models. ISO 14649 is thought of as modelling information for process planning at the micro level, hence the intention to replace G-code, which is traditionally and still extensively used to programme NC machine tools. It is therefore worth mentioning ISO 10303-240 Process plans for machined products [205], which can model process planning information at the macro-level.

The ISO TC184/SC4 manufacturing working group, on the other hand, has adopted the ARM models built in SC1 as the ARMs for AP 238. The model is then mapped to STEP integrated resources to obtain an implementation model.

A set of ISO 14649 constituting the STEP-NC standards is listed as follows:

- ISO 14649-1: Overview and fundamental principles [33]
- ISO 14649-10: General process data [206]
- ISO 14649-11: Process data for milling [166]
- ISO 14649-12: Process data for turning [207]
- ISO 14649-111: Tools for milling [167]
- ISO 14649-121: Tools for turning [208]
These Parts are arranged hierarchically, in that Part 11 uses Part 10 and Part 111, while Part 12 uses Part 10 and Part 121. Part 10 provides a set of basic capabilities for process planning for machined parts. As shown in Figure 6.5, Parts 11 and 12 represent the technological capabilities for milling and turning, respectively. It can also be seen that the development of the standards still continues to cater for other applications such as grinding, rapid prototyping and data modelling for machine tools.

Figure 6.5 CAx Chain Enabled by STEP-NC

A multitude of benefits from using STEP-NC in comparison with G-code have been recognised (Figure 6.6):

- STEP-NC provides a complete and structured data model, linked with geometrical and technological information, so that no information is lost between the different stages of the product development process
- its data elements are adequate enough to describe task-oriented NC data
- the data model is extendable to further technologies and scalable (with Conformance Classes) to match the abilities of a specific CAM, Shop-Floor Programming (SFP) or NC system
- machining time for small to medium-sized job can be reduced because intelligent optimisation can be built into the STEP-NC controllers
- post-processor mechanisms will be eliminated, as the interface does not require machine-specific information
• machine tools are more adaptable because STEP-NC is independent from machine tool vendors
• modification at the shop-floor level can be saved and fed back to the design department. Hence, bi-directional information flow from CAD/CAM to CNC machines can be achieved.

Figure 6.6 Comparison of G-code and STEP-NC Data [209]

6.2.1. STEP-NC Content and Structure

The fundamental principle of a STEP-NC data model is the object-oriented view in programming. In terms of manufacturing features, its terminology is different from direct coding of axis motions and tool functions, as defined in ISO 6983 (G-code). As a consequence, STEP-NC can effectively define a data input standard for CNC systems. As STEP-NC is an extension of STEP in handling NC processes, it strictly follows the STEP standard. Like other STEP applications, STEP-NC files also conform to ISO 10303-21.

Figure 6.7 represents the structure of a STEP-NC data model. The first section of the part programme is the section called HEADER. In this HEADER, some general information and comments concerning the part programme are included. These can be, for example, filename,
author, date and organisation. The second and main section of the programme file is the data section, namely DATA. This section contains all the information about manufacturing tasks and geometries. This section also includes a Project entity that is an explicit reference for the starting point of the manufacturing tasks. The Project entity contains a main “Workplan” that contains sequenced executable manufacturing tasks called “Workingsteps”.

Executable objects initiate actions on a machine and are arranged in a predefined but changeable order. The data section of a STEP-NC file has three sub-sections: a Project entity, Workplan/executables and a Geometric description. The data in a STEP-NC file consists of instances of entities. One of the instances is called Workplan, which contains sequenced subsets of executable manufacturing tasks or commands and may also include information from the workpiece that is to be machined [166, 206]. The executables can be of three different types: Workingsteps, NC functions and programme Structures.

Machining_workingsteps are the essential elements of executables in a STEP-NC physical file, which are defined based on two5D_manufacturing _feature and 3D (region) of machining_features. They therefore specify the connection between a distinct manufacturing feature and a machining operation to be performed on that feature. Each Workingstep also

---

**Figure 6.7 Structure of the STEP-NC Data Model**

Executable objects initiate actions on a machine and are arranged in a predefined but changeable order. The data section of a STEP-NC file has three sub-sections: a Project entity, Workplan/executables and a Geometric description. The data in a STEP-NC file consists of instances of entities. One of the instances is called Workplan, which contains sequenced subsets of executable manufacturing tasks or commands and may also include information from the workpiece that is to be machined [166, 206]. The executables can be of three different types: Workingsteps, NC functions and programme Structures.

Machining_workingsteps are the essential elements of executables in a STEP-NC physical file, which are defined based on two5D_manufacturing _feature and 3D (region) of machining_features. They therefore specify the connection between a distinct manufacturing feature and a machining operation to be performed on that feature. Each Workingstep also
includes further sub-features such as planar_face, pocket, step, slot and round_hole, in addition to cutting condition information. The Workplan combines several executables in a linear order that depends on given conditions if conditional controls are used. Some geometrical data information for workpiece, set-ups, manufacturing features, machining strategy, tooling and the rest is also included in a STEP-NC physical file.

In essence, STEP-NC describes “what to do”, while G-code describes “how to do” it. STEP-NC describes tasks (pre-drilling, drilling, roughing, finishing) that are based on the machining features, so that the part programme supplies the shopfloor with higher-level information, which is information about machining tasks and technological data on top of pure geometrical and topological information. As a result, modifications on the shop floor can be saved and transferred back to the planning department, enabling a better exchange and preservation of experience and knowledge.

6.2.2. Machining Features

Since the research reported herein includes a feature machining service via Cloud Service, it is necessary to analyse the taxonomy of the STEP-NC features. In STEP-NC, a machining feature is placed in an object-oriented data structure as shown in Figure 6.8. Machining_features, as the supertype of all features and the subtype of two5D_machining_feature, are defined as having close resemblance to ISO 10303-224 [210]. They are planar_face, pocket, slot, step and the rest. Their definition, entity and attributes are explained in the ISO 14649-10:2003 standard. A feature is described as a form of machining contour that consists of a set of parameters. For instance, planar_face is described as machining of the outer of a workpiece. Two attributes used to define planar_face include course of travel and removal boundary. Course of travel denotes a straight-line with magnitude and direction, whereas removal boundary denotes a line with direction and magnitude, which swept along a path defines the area on a workpiece for volume removal. The geometry of the planar_face in z-axis is given through the depth. The depth expresses the bottom of the material that needs to be removed from the workpiece to achieve the final shape of the feature.
Unlike planar_face, step feature has only one defined attribute, i.e. open boundary. Open boundary denotes the outline or the shape that forms the upper edge of the step. When travelling along the curve as defined by its sense, the material is to the left of the curve. However, both planar_face and step inherit depth as an elementary surface that can be inclined or orthogonal to the feature’s local z-axis.

Pocket is a supertype of closed_pocket and open_pocket. A closed_pocket is a pocket with its feature boundary given by its contour on the outer face that is enclosed by the workpiece and its depth. Unlike closed_pocket, open_pocket is a pocket with its feature boundary given by an open_boundary and a wall_contour. Open_boundary denotes the outline or shape that forms the upper edge of the open_pocket, and wall_contour denotes the outline or shape that forms the side-edge of the open_pocket. The contour is defined implicitly by the selected tool and the fillet options inherited from machining_feature.

Similarly to a pocket, an entity Slot also has two types – slot with radiused end and slot with two open ends. Generally speaking, a slot is a special type of pocket. Entity slot consists of only one attribute, i.e. course_of_travel. Course_of_travel denotes the location and extension of the slot. Entity slot is typically machined by a single remove, whose shape is given by the tool diameter. However, when a slot has a greater width than the tool diameter, more than one cut will have to be made. The complete description of planar_face, step, closed_pocket, open_pocket and slot can be seen in Figure 6.9 (a), (b), (c), (d) and (e), respectively.
6.2.3. Machining Operation

In addition to machining feature information, a STEP-NC data model also captures information about machining operations. For example, process data for milling is represented in ISO 14649-11:2003, specifying the technology – specific data element needed for defining milling processes. Among various types of data, milling_type_operation is utilised in this research and is therefore discussed in this section.

Milling_type_operation is inherited from entity machining_operation, defined in ISO 14649-10 as describing machining technology and strategy. Meanwhile, milling_type_operation is also a supertype of freeform_operation and two5D_milling_operation. A few two5D_milling_strategies are used in the proposed systems, i.e. unidirectional, bidirectional, contour_parallel, bidirectional_contour, contour_bidirectional, contour_spiral, center_milling and explicit_strategy. Optional information about overlap is required for machining strategies in the proposed system. The overlap is the path between two neighbouring cutting
movements as a percentage of the tool diameter. In a latter section, the overlap is categorised as either full-immersion or part-immersion.

6.3. Data Packet Mechanism

Although using STEP/STEP-NC is a possible solution to achieving system interoperability, at the same time some drawbacks have been observed. It is not a trivial task to consolidate a large number of heterogeneous applications. Semantic issues still exist. In many data-centric systems, software applications are encapsulated in a highly closed environment, which goes against the modern business practice that is becoming more and more decentralised. Furthermore, there are still synchronisation and confidentiality issues with this data-centric approach.

Since the key concept of STEP is to provide an integrated information resource, it may lead to risks for synchronisation and confidentiality. In modern industry, manufacturing is often conducted cooperatively. As illustrated in Figure 6.10, product data may experience a parallel or serial flow through the system. When a product is being manufactured by different suppliers or contractors at the same time, a parallel data flow occurs. Communication between suppliers is usually minimal if any; so is data synchronisation.

Moreover, when a product is manufactured by different suppliers serially, a serial data flow forms. The requirement of passing on a complete data model from one supplier to another may infringe on the supplier’s confidentiality requirement. This means that use of data-

Figure 6.10 Synchronisation and Confidentiality Issues for Data-centric Approaches
centric approaches makes it hard to deal with intellectual property matters. In the ICMS storage Cloud, the product data is saved in the STEP and STEP-NC data format for archiving purposes, along with the native data formats of CMService Modules. As mentioned above, despite the advantages of the STEP/STEP-NC data format, intelligent property protection issues still exist. The key aim of STEP-based data integration is to maintain data in one unified syntax and document. However, it may lead to a large STEP file when distributed assemblies and kinematics information are included. It was determined by the ISO working group recently that the STEP data of big manufacturing projects, e.g. complete airframes and ships, is too large to manage [211]. In particular, for the CManufacturing paradigm, it is neither logical nor safe to expose the data to all the participants involved. Thus a collaborative product data exchange mechanism was needed and developed.

6.3.1. Collaborative Product Data Exchange

When a number of suppliers, contractors and sub-contractors are working on the same project, it is important to meet the requirements of data synchronisation and confidentiality. Even in one enterprise, it is neither necessary nor safe to expose all the data to every department or employee. As shown in Figure 6.11, a STEP-NC document contains information across multiple layers of a manufacturing project. For a project manager, he or she may only work with the general data at high level, e.g. main work plan, selection of work piece and status of the project. The shop floor manager concentrates more on detailed data, for example the setups of sub work plans and working steps status. For machine operators, their concern is domain-specific data that is only related to the machining process he or she is in charge of. For instance, a CNC operator may only need the information for the setup of a machining feature, requirements of cutting tools, etc. There is no need to provide the information at higher levels to irrelevant participants. In particular, for CManufacturing, a service package may be achieved by multiple participants at different operation levels. Thus, it is very important to ‘provide the right amount of data to the right people in the right manner’.
In ICMS, the concept of Data Packet (DPacket) was conceived. A DPacket is defined as a set of self-contained mobile clusters of data. Once the Service List allocates the data subset needed, the Data-Localisation Mechanism extracts and generates a stand-alone file according to the data domain defined in the list, before it is packaged in the Cloud Service Combination (Figure 6.12). After the DPacket is processed by the user, the modified information needs to be updated. In this mechanism, a DPacket can be “stitched” back to the data source, which is called the Data-Integration Mechanism. In general, the goal is to develop an algorithm to identify logical connections amongst different data subsets across different levels. According to these connections, a stand-alone file containing an assigned data subset is generated and delivered to the user. In practice, since STEP/STEP-NC describes product information from an object-oriented perspective, it is possible to identify and extract data in a specific scope. This provides the user with an interoperable environment to work on the appropriate subset of data. In this way, data synchronisation is guaranteed and confidentiality maintained.
Unlike the Working Set method of ST-Developer, the DPacket mechanism is able to identify a specific range of data and generate a processable file for the user. Meta-data models were developed to record and recover the logic connections within and around DPackets. Even though the DPacket may be changed by a user during service processing, with the help of these meta-models the mechanism is able to re-connect the DPacket back to the resource, based on the data structure, without affecting the information in other working domains.

6.3.2. Data-Localisation Mechanism

The concept of a DPacket was realised by pre-processors and post-processors, both being encapsulated in VFB modules. After a service list is generated by the SCM, the pre-processor searches in the database and locates the top-level information requested by the task. After the DPacket is located, all relevant information is extracted and transformed into a self-contained file for the target application.

To realise the DPacket concept, it is necessary to extract and process a DPacket from a STEP physical file [170]. Although the instance of entities follows the data structure defined by the corresponding EXPRESS schema, the internal logic relationships amongst entities stay implicit, as in a text-based Part 21 file. It is difficult to process and recover this kind of relationship from the file directly. Therefore, it is necessary to re-describe the product information in the STEP files. Because STEP/STEP-NC describes the information in a task-oriented way using ‘entities’ and ‘reference relationships’, such logics can be described in a tree-structure. Thus a meta-data model was defined to re-represent the product information from a Part 21 file. In this data model, the instances of entities stored in a STEP file are denoted as ‘nodes’ and the relationships between nodes denoted as ‘edges’. For one tree node, all the information defined in the original data source is kept. As illustrated in Figure 6.13, the entity-attribute structure is mapped to Parent-Child logics in the DPacket meta model. The entity instance at a higher level is defined as Parent Node, and the ones at a lower
level as Child Node. The specifications of the attributes are kept in edges including attribute type, optionality, etc.

![Parent-Child Logic](image)

The process of DPacket extraction process is summarised in Figure 6.14. When the top-level entity of a DPacket is assigned, this entity is defined as the first ‘Parent Node’. Then the attributes of the Parent Node are traversed. All the information defined in the entity is transformed and kept as edges of the Parent Node. The types of attribute are filtered by the Boolean logic of DPacket. When the attribute is pointing to another node, a temporary edge is formed and attached to the Parent Node. After Parent Node’s analysis is done, the mechanism rechecks the Service Template. If the DPacket has enough information according to the ST, the system will withdraw unused temporary edges and generate an executable STEP file based on the DPacket extracted. If the last Parent Node has not reached the end of DPacket, which means the user requires more information around the top node, the system will allocate the Child Node attached on the edge as the new Parent Node, and run the analysis process again until all the information needed is collected. By defining parent-child logics, this mechanism enables the user to access the right amount of data, fulfilling user demand across various data levels. For instance, a DPacket generator is able to provide general information to the user, such as the name and owner of a project, while the technical details, dimensions and parameter information of a specific task can be packaged in a DPacket as well.
In one line of a Part 21 manufacturing document, for example the main work plan, the original code of the main work plan is shown at the top of Figure 6.15. The semantics maintained in the Part 21 file are interpreted into the DPacket meta-model. A node structure consists of slot, list and pointer. Starting with the hash symbol (#), a new node is created. The data is filled into slots based on the meanings of the symbols. During conversion, when a hash symbol is detected within an entity instance, it means this entity points to other instances. A pointer space is then created that directs the Parent node to the child, along with a new edge structure connecting them. In this case, the definition of “main workplan” is
interpreted and all its elements are found and attached, i.e. sub work plan entity instance #4, #5, #6. After all the requested data is collected, it is converted to Part 21 format again, thus forming a DPacket document that contains all the data needed.

6.3.3. Data-Integration Mechanism

After a CMService is completed on the DPacket, VFB detects the results and the post-processor is initialised before DPackets are reassembled to the data source. Since STEP/STEP-NC stores the product data in an object-oriented way, it helps maintain the integrity of the data and reconnect the broken links between the DPacket and the original data source. In more complicated situations, when multiple changes are made to the same data set by different users, the integration mechanism synchronises the changes and makes a backup version for each of them in the Product Database. The history of the product data is maintained in this way.

With Data-Integration mechanism, synchronisation is also catered for. The DPacket philosophy provides a specific data working domain for various users. Since users are limited to processing the subset of data in the working domain, data changes can be updated one by one without affecting the information outside the scope of DPacket, which takes care of synchronisation difficulties and data conflicts. Before each update is deployed, a backup version is saved in the database in order to enable traceability and recoverability of the information.

However, when multiple users are working on a task at the same time, data updating can be a problem. As illustrated in Figure 6.16, when various users extract DPackets from the Database regarding the same working set, there is a risk of data loss during the updating
procedure. In a more complicated situation, as shown on the right-hand side, it may lead to bigger data loss by updating the whole DPacket purely based on a timestamp. When processing periods overlap, it is difficult to pinpoint which version should be kept as the latest in the database. Therefore an advanced data synchronisation mechanism is needed. To synchronise and harmonise the DPacket changes, a DPacket agent is proposed to take charge of the updating procedure. Timestamps are stored each time a DPacket is generated and integrated. Before the DPacket is updated back to the database, the agent checks whether there is any time overlap with another DPacket in the same data domain. If there is, a negotiation agent is launched, which enables the user to choose which version of data will be kept, through negotiation with the previous version. For instance, when DPacket 1.3 is synchronised, it is advised by the system to negotiate over the update results with DPacket 1.2, and then a timestamp notice will be marked for the next update procedure (DPacket 1.1 update).

When it comes to some special type of data, for instance a sensitive or important data segment, an exclusive agreement can be deployed to the meta-data entities. When a DPacket related to this subset is generated, exclusive rules are activated, which means no other user is able to view, extract or change the information in this scope until the current task is finished. In this way, the data file synchronisation and version control is guaranteed.

In order to realise the DPacket concept and model the related information, data models compliant with STEP and STEP-NC standards were developed using the EXPRESS-G diagram (Figure 6.17). The Data Packet model is directly connected to the service definition in the S-Module. Therefore, when a service is launched by the SCM, related DPacket is
assigned and extracted from the database. The DPacket specification is defined via entity DPacket_definition, which has the attributes its_id, defined_version, its_timestamp, parent_entity, depth_of_data, its_connection, security_setup, etc. Defined in the structure that is compliant with the service model for the control, internal data portability within ICMS is assured.

Entity ID stores a unique id for every single DPacket so that all data subsets are easily identifiable. Since all the DPacket information described by this model is saved in the database, the history of a DPacket can be traced, thanks to the Entity ID. Entity DPacket_version describes version information about a DPacket during a working procedure. When multiple users are working on the same DPacket, different versions can be identified by the version details, along with the ID information. Entity Time_stamp records the time when a DPacket was created and integrated. With the time_in and time_out attributes, i.e. the date_and_time entity defined in ISO 10303-41 [212], the synchronisation mechanism mentioned above can be enabled, based on the chronological data. Entity Parent_entity allocates the top level of a DPacket. As discussed in 3.2.1, a Parent node is allocated, based on the service description from a user. Set up as the key of a DPacket, information related to the Parent_entity is found and extracted from the database for a task.

Entity Data_depth describes the amount of data that is made available to the user. Based on this entity, the range of a DPacket can be defined to meet a specific user need. Scopd by
Data_depth, DPacket is able to provide an appropriate amount of product data, across various data levels from the data source. Entity Connection stores the information about entities that are connected to the DPacket. When the data integration mechanism is collecting child nodes for a DPacket, the entities connected but not included in this task are recorded in Connection entities. Thanks to the object-oriented nature of the STEP data model, these external connections with DPacket can be rebuilt after the DPacket is processed by the user, via the integration mechanism. Entity Security_setup defines the security information about a DPacket. Along with the Entity Owner, the user information of a DPacket user is stored and security management will be operated based on the authorisation level of such data. System Firewall is responsible for defining the access level of a specific user regarding a specific data subset, for instance accessing, viewing only, or changing/deleting permission. As optional entities, attributes description, name and additional characterisations enable the system to add more customised information to a DPacket. Comments, records and working notes can be attached to a DPacket for future use. To maintain consistent data semantics and syntax, a validation mechanism is in place.

As part of the post-processor, the validation mechanism detects any unreasonable parameters, such as too large/small dimensions for a manufacturing feature and ill-defined manufacturing information (e.g. a minus diameter for a driller). Furthermore, harmonisation between the DPacket and the source data will be validated as well. For instance, if the depth of a pocket is changed in the DPacket while the tool-path depth remains unchanged in the data source, the validation system will detect the conflict and send a warning message to the user. At completion of the validation process, the post-processor shuts down and a service-ending message is delivered to the S-Module before a new service is launched. The object-oriented nature of STEP makes it possible to develop a validation mechanism based on the concept of DPacket.

6.4. Recap

At present, manufacturing organisations are facing interoperation difficulties because of their heterogeneous data environment and the software vendors they are dealing with. Meanwhile, product lifecycle management is challenged by the conflict of synchronisation and confidentiality issues. Using neutral product data formats (e.g. STEP/STEP-NC) is a practical solution to archive data interoperability, and to relieve the issues mentioned here. However, the problems cannot be overcome by introducing these data models directly. This paper
introduces a novel Data-Packet concept to represent product semantics via the STEP/STEP-NC data model. Inspired by previous research, DPacket has benefitted through the data integrity of STEP. Meanwhile, data confidentiality and traceability is maintained. In this research, milling process is undertaken to evaluate DPacket’s capability of processing manufacturing data. Detailed examples and implantations can be found in Case Study 2, Chapter 7.

Implemented in ICMS, the Data Localisation/Integration mechanism proves it is practicable to process a data subset containing a reasonable amount of information. These methods enable users to work with product components across different levels of abstraction, using different representations, without affecting the working domains of other users. Furthermore, participants are able to synchronise the parallel modifications after the data subsets are processed in a distributed environment. This research work neutralises the conflicts between data integration and confidentiality/update requirements, and creates new application possibilities using the object-oriented (STEP) data structure.
Chapter 7. System Implementation and Case Studies

This chapter introduces the proposed Cloud-based Interoperable Manufacturing System, its methodologies and the algorithms developed. It describes the implementation of this system framework, service and resource database and the prototype software supporting the Data Packet philosophy. The system was developed in the NetBeans Java environment, with the utilisation of multiple software tools, such as ST-Developer, Microsoft Access, Creo Parametric, etc. Three case studies were carried out to test the proposed concept and system. The software functions are introduced, together with the case studies.

7.1. Software Development Environment and Tools

The ICMS is Windows-based, and is built using an object-oriented modelling technique. The principal language is Java, due to its platform neutrality and portability. ST-Developer is used for processing STEP/STEP-NC data. Microsoft Access™ is utilised to build and maintain the Cloud database.

7.1.1. NetBeans IDE

NetBeans Integrated Development Environment (NetBeans IDE) is an open-source IDE that supports the development of all Java application types. It helps develop the software applications together with Java Development Kit (JDK) and Java Runtime Environment (JRE). The NetBeans platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform can be extended by third party developers as plug-ins. It is used as a platform for Windows 7 to create software applications with Java. JDK is a set of Java development tools and JRE is a software framework containing a programming execution environment and libraries. Both of them are provided by Oracle Cooperation. Coupled with NetBeans IDE, they provide an integrated development environment for ICMS modules and agents.
7.1.2. ST-Developer™

ST-Developer, provided by STEPTools Inc [139], is a set of software development tools used to build, operate and maintain STEP, Industry Foundation Classes (IFC), CIMsteel Integrated Standard (CIS/2) and EXPRESS-defined tools, translators and databases. Since STEP was chosen as the neutral data format for product and resource maintenance, ST-Developer was utilised to support programming bindings for Java application tools used for testing data sets against verification rules and constraints, browsing through the contents of data sets and building information models. It also provides libraries for reading, writing, processing and checking STEP data Part 21 formats.

7.1.3. Microsoft® Access

Microsoft Access is a database management system that combines the relational Microsoft Jet Database Engine with software-development tools. It supports online services and a web-based database functionality, e.g. retrieving records and displaying them on the Internet browser. In this research, Access is utilised as the intermediate methodology to bridge standardised resource data format and web-based functionalities.

7.1.4. Creo™ Parametric

Creo is a suite of software applications supporting product design, engineering and manufacturing and is provided by PTC. Creo Parametric is its 3D CAD/CAM feature for solid modelling. As a typical manufacturing application, it is a commercial software tool that does not support neutral data formats like STEP-NC. Therefore it was chosen as a target application that needed to be integrated in the Manufacturing Cloud, to evaluate the interoperability and extendibility of ICMS.

7.2. Case Study 1: Cloud Service Provision

At the service level of ICMS, two examples are provided to evaluate the concept of ICMS. The first study example shows its ability to integrate CMServices to fulfil a Customer User request for a series of service packages. The second example demonstrates how a CManufacturing environment can assist an Enterprise User with multiple solutions and improve its enterprise performance at the same time.
7.2.1. Cloud Service for a Customer User

In the first example, a customer launches a customer interface remotely. The interface is developed in HTML format via the Adobe Dreamweaver package. It can be viewed within web browsers, e.g. IE, Safari and Firefox (Figure 7.1). CUs can select CManufacturing Services from predefined categories and specify service details. After logging on to the system, Users are able to view their service history and the status of his or her services in-process. In this case, the user requires a series of services including product design, simulation, CNC milling, precision welding and grinding.

![Figure 7.1 ICMS Interface](image-url)

Service query descriptions and specifications are input via a GUI. CIAgent collects these data and transmits the data into an XML format to enable Internet communication (Figure 7.2). As mentioned in Chapter 4 the XML document, which is transmitted from the client side, is compliant with the EXPRESS-G schemas but in a flat data structure. This means that the meta-model only keeps the information of entity instance and temporarily suspends attributes and inheritance logics. The flat meta-model is utilised to enhance data portability. CUs are
able to view and process these data via popular software tools, e.g. web browser, Microsoft Excel, Access and so on. When the data comes back to the MCloud, it is mapped back to the structured model tree. Then SCM continues to process the service process and searches for all possible solutions.

![XML Representation of a Service Request](image)

As a result, the SProviders which are able to meet a user’s requirements are summarised in Figure 7.3. The service is constructed of five service phases that are mapped to the five stages of production service. The service is illustrated as nodes and the transmitting methods as edges. Each node indicates an SProvider that is able to accomplish the service task in this phase. Multiple service providers are matched and stated by the SCM. For instance, in Phase 1, there are three SProviders that can provide the design service based on the query specification. Taking into account different route sequences, the potential service procedures are presented in the flowchart.
To optimise the solutions and provide a customised result, the SCM analyses the results from
the Storage Cloud and evaluates the selections based on user preference (e.g. cost-efficient
and time-efficient rules). The total cost includes the cost of service, and freight between
different providers. After SProviders are found, all possible transmitting services are detected
as attachment services. The actual total cost includes the Cost of service at phase \(i\) (Cost(\(i\)),
and the Shipping Cost that follows phase \(i\) (SC(\(i\))).

\[
\text{Cost} = \text{Cost}(i) + \text{SC}(i) = \sum_{i=0}^{n} C(i+1) + \sum_{i=0}^{n+3} SC_{i,(i+1)}
\]  
(7.1)

\[
\text{Cost}_{\text{opt}} = \text{Minimise}[\text{Cost}(PD, S, CM, PW, G) + \text{Cost}(SC_{jk}, SC_{kl}, SC_{lm}, SC_{mp})]
\]  
(7.2)

where,

- PD – Product Design;
- S – Simulation;
- CM – CNC Milling;
- PW – Precision Welding;
- G – Grinding.

In this example, the SCM computes all possible paths through the service nodes and their
edges. The sequential relationships between nodes can be represented as shown in (Table 7.1).
Value “1” indicates there is an operational constraint from node X to node Y. The lowest cost
combination is found and suggested to the user. Since there are 36 combinations in this case,
an iterative algorithm is deployed. For more complex business matters, the neural network
and a genetic algorithm can be used to improve computational efficiency.
Computation under the time-efficient rule is similar. Besides the period of service, results of transitions between SProviders cannot be ignored. SCM traverses all possible paths and finds the most sufficient solution in terms of time.

\[
\text{Time} = \sum_{i=0}^{n}(T_{(i+1)} + ST_{(i+1)}) \quad (7.3)
\]

where,

- \(T\) – Time of Service;
- \(ST\) – Time of Shipping.

\[
\text{Time}_{opt} = \min\{T(PD_j, S_k, CM_{l}, PW_{m}, G_p) + T(SC_{jk}, SC_{kl}, SC_{lm}, SC_{mp})\} \quad (7.4)
\]

Eventually, the entire solution pool is detected and provided to the user. The results are mapped into a general chart visible to the user via the User Interface (Table 7.2). The optimised solution is marked for the user (black bullet points). At this stage, the user is able to follow the SCM recommendation or customise the selection.
To recap, manufacturing services, from design to production, can be virtualised and integrated within public CManufacturing architecture, without changing the existing organisational structures of a manufacturing enterprise or company. CMService participants are able to improve their enterprise performances while maintaining their autonomy and competitiveness. The participants are part of the integrated environment via contributing their valuable resources and services for mutual benefits.

### 7.2.2. Cloud Service for Enterprise User

When manufacturing enterprises are brought into the MCloud, the first step is to understand the MCapability of an enterprise, which is then offered as a CMService. The second example concerns a manufacturing SProvider named BE, which has factories around the world. An example product was selected to demonstrate how an industrial service is virtualised and implemented in the MCloud. The example product, ID T14859 Driver, is a plate-shaped part with multiple holes at different angles (Figure 7.4). An order of thirty parts was processed at the BE Australia branch.

<table>
<thead>
<tr>
<th>Suggestion (Cost)</th>
<th>Suggestion (Time)</th>
<th>Service (Activity)</th>
<th>Provider</th>
<th>Service Serial No.</th>
<th>Activity ID</th>
<th>Price/each</th>
<th>Quantity</th>
<th>Time (week)</th>
<th>Capability</th>
<th>Price (total)</th>
<th>Time (total)</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Design</strong></td>
<td>CAMEX</td>
<td>5023</td>
<td>C001</td>
<td>1100</td>
<td>1</td>
<td>2</td>
<td>***</td>
<td>1100</td>
<td>2</td>
<td>North Shore, NZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product Design</strong></td>
<td>EagleBurgm  ann</td>
<td>5001</td>
<td>E006</td>
<td>900</td>
<td>1</td>
<td>3</td>
<td>***</td>
<td>900</td>
<td>3</td>
<td>North Shore, NZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product Design</strong></td>
<td>UOA Workshop</td>
<td>5003</td>
<td>U011</td>
<td>800</td>
<td>1</td>
<td>2</td>
<td>***</td>
<td>800</td>
<td>2</td>
<td>Auckland, NZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>Data Trans</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Simulation</strong></td>
<td>PTC Creo</td>
<td>5009</td>
<td>P001</td>
<td>1000</td>
<td>1</td>
<td>2</td>
<td>***</td>
<td>1000</td>
<td>2</td>
<td>PTC, JS</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Simulation</strong></td>
<td>UOA Workshop</td>
<td>5008</td>
<td>U013</td>
<td>700</td>
<td>1</td>
<td>1</td>
<td>**</td>
<td>700</td>
<td>1</td>
<td>Auckland, NZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>CAMEX -&gt;CAMEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NZPOST</td>
<td>S101</td>
<td>N101</td>
<td>500</td>
<td>1</td>
<td>0.5</td>
<td>***</td>
<td>500</td>
<td>0.5</td>
<td>CAMEX -&gt;BENZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NZPOST</td>
<td>S102</td>
<td>N102</td>
<td>500</td>
<td>1</td>
<td>0.5</td>
<td>***</td>
<td>500</td>
<td>0.5</td>
<td>BENV -&gt;CAMEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>BENV -&gt;BENV</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NZPOST</td>
<td>S104</td>
<td>N104</td>
<td>600</td>
<td>1</td>
<td>0.5</td>
<td>***</td>
<td>600</td>
<td>0.5</td>
<td>UOA -&gt;CAMEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Precision Welding</strong></td>
<td>CAMEX</td>
<td>5012</td>
<td>C004</td>
<td>60</td>
<td>1</td>
<td>100</td>
<td>2</td>
<td>600</td>
<td>2</td>
<td>North Shore, NZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Precision Welding</strong></td>
<td>BENZ</td>
<td>5013</td>
<td>E003</td>
<td>40</td>
<td>1</td>
<td>100</td>
<td>1</td>
<td>400</td>
<td>1</td>
<td>North Shore, NZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NZPOST</td>
<td>S101</td>
<td>N101</td>
<td>500</td>
<td>1</td>
<td>0.5</td>
<td>***</td>
<td>500</td>
<td>0.5</td>
<td>CAMEX -&gt;BENZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freight</strong></td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>BENV -&gt;BENV</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Grinding</strong></td>
<td>BENZ</td>
<td>5010</td>
<td>E001</td>
<td>70</td>
<td>1</td>
<td>100</td>
<td>2</td>
<td>700</td>
<td>2</td>
<td>North Shore, NZ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 7.2 Cloud Service Mapping Results from Smart Cloud Manager*
In fact, BE has sites in Australia (BE-AU) and New Zealand (BE-NZ), both of which are SProviders and capable of machining this part. Thus it was necessary to virtualise their capability in the Cloud, evaluate the task and decide which branch should process it. Firstly, the MCapabilities of the two sites were identified (Figure 7.5) in the Storage Cloud.
Chapter 7 – System Implementation and Case Studies

Historically, product data and process plans were documented in different data semantics. Due to the geographical and currency differences, it was hard to evaluate machining tasks in the two sites. With the help of ICMS, the technical details are mapped from the flat chart to the modularised Cloud_Service_Specification model previously mentioned. The decision maker of BE is able to compare and assess the cost, time, resource and status of both site with the help of SCM. SCM provides a standardised and explicit methodology to describe the capabilities, without the need of previous knowledge or experience with the candidate providers.

There are stronger Hard Resources at BE-NZ compared with BE-AU, i.e. a 9-axis multitasking CNC machining centre, with Sandvik T–Max® U Trepanner cutters and Sandvik saw blade cutters. With the help of the 9-axis machining centre and its bigger cutter magazine at BE-NZ, the clamping and setup time can be drastically decreased. Additionally, the service input is a 316 stainless steel round bar of 300 millimetres diameter. Originally, the removed materials were in the form of chips that could only be recycled. Thanks to the Trepanner cutting tools, the result of the machining process was a hollow product with a cylindrical bar (diameter 130 millimetres) that could be utilised directly as a service input for other processes at BE-NZ. Thus the actual material cost was drastically reduced. The cost margins were
mapped to the subtypes of Service_Cost. The parameters and features are described by the Entity Hard_Resource. Based on the data in this section, the actual service cost was summarised and provided. Subsequently, BE-NZ was chosen to carry out the production task.

In this case study, ICMS served as a Community Cloud for the organisation. BE played the role of SProvider and EUser at the same time. The Cloud-based virtualisation mechanism provided explicit and integrated presentations of its service/products. It provided clear knowledge of the operational processes. Business decisions can be made without the need of previous history or knowledge of the candidate solution. From the perspective of a provider, BE was able to understand available services and review/optimise them as required. As a CU, an enterprise is able to choose the most suitable service provider, which could be a partner/contractor connected by the MCloud or the enterprise itself.

7.3. Case Study 2: Data Packet Implementation

As mentioned above, STEP provides data models throughout the CAx chain. It supports production activities by various APs facing specific domains. As shown in Figure 7.6, a typical STEP-based supply chain involves multiple APs in describing product and manufacturing data in specific domains. With the help of the DPacket mechanism, a reasonable amount of data can be provided to support a specific task. Data objects can be extracted out of the reference tree, and output as a self-contained document.

![Figure 7.6 Typical STEP-based Production Procedure](image)

---

161
7.3.1. Converting EXPRESS Schema into Java Class

The EXPRESS language was used to model the product and resource data structure. However, it is not a data processing or management language. In order to utilise the data model defined in EXPRESS, it was necessary to compile/convert the EXPRESS schema into another programming language that is the actual system development language, i.e. Java. The compiler takes the EXPRESS schema as the input, and compiles each entity within the EXPRESS schema into Java classes. The Java classes are effectively a set of Java source codes defining each entity consisting of .java files. Figure 7.7 shows the procedure of how the EXPRESS to Java Converter is utilised in converting EXPRESS definitions (ICMS test schema 01.exp) into a set of Java class files (i.e. Project.java, Profile.java, Workpiece.java, etc.).

![Figure 7.7 Converting EXPRESS Schema into Java Classes](image)

7.3.2. Implementation of the ST-Developer Interfacing with Java

Before the output classes are utilised in the NetBeans IDE environment, environment variables need to be set up. This is done via a so-called “library” file in the ST-Developer installation directory, for example “C:\ST-Developer14\lib\java”. ST-Developer for the Java programming environment consists of an EXPRESS Java Converter that generates Java classes as mentioned, and a set of foundation classes (e.g. stdev.jar package) that provides...
services such as reading and writing instances to STEP Part 21 exchange files. All the .jar files are located in the installation lib/java directory. Along with other libraries and .java classes, the ST-Developer application is integrated with the Java programming environment (Figure 7.8).

![Figure 7.8 Importing ST-Developer Library into Java](image)

7.3.3. Data Packet Generator

The test product information is stored in a STEP-NC file (ISO14649-11 example 2 [166]). As illustrated in Figure 7.9, the first section of the part programme is the header section marked by the keyword “HEADER”. In this section, some general information and comments concerning the project are given, e.g. filename, author, date and organisation. The second and main section of the programme file is the data section marked by the keyword “DATA”. This section contains all the information about manufacturing tasks and geometries.
Chapter 7 – System Implementation and Case Studies

To develop the system, the JAVA language was chosen for its interoperability amongst different platforms. The system was developed using JDK packages with JRE. First of all, the STEP-NC physical file is read by the system, and interpreted into the meta-data model. Then the meta-model is represented in the tree-structure. In this way, the text-based STEP-NC information is transformed into programmable classes that are capable of being analysed and processed. To implement the DPacket concept and evaluate the algorithm, a GUI (Figure 7.10) is developed. After the product document is loaded into the programme, the data structures are visible in the tree view interface, including entity type, contents and attribute details.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(ISO 14649-11 EXAMPLE Z, 'COMPLEX PROGRAM WITH VARIOUS MANUFACTURING FEATURES'), "1",
FILE_NAME(EXAMPLEZ.STP);
...
FILE_SCHEMA('MACHINING_SCHEMA', 'MILLING_SCHEMA');
ENDSEC;

DATA;
#1=PROJECT('EXECUTE EXAMPLEZ', #2(#7), $, $);
#2=WORKPLAN('MAIN WORKPLAN', (#4, #5, #6), $, #14, $);
...
#7=WORKPIECE('PART Z', #13, 0.01, $, ($9, #10, #11, #12));
...
#78=BORING($, $, BORING_HOLEZ', 20, $, #266, #270, #230, $, 15, 15, $, $, 'T', 1, $);
...
#101=COMPOUND FEATURE('COMPOUND FEATURE HOLEZ', (#7), (#108, #109, #110));
#108=ROUND_HOLE('HOLEZ', #78, #369, #545, #546, $, #214);
...
ENDSEC;
END-ISO-10303-21;
```

Figure 7.9 STEP-NC Data Structure
After receiving the control message from the S-Module, the system searches in the structure tree and launches the DPacket process. Assume that a user requires a view of the detailed information about a MACHINING_WORKINGSTEP and its corresponding feature called PLANAR_FACE. According to the Service Template based on a customer request, the system firstly locates the top level of the DPacket and then scans its entity classes. The user is able to choose different methodologies to locate the key entity instance, for instance ID of entity, name or entity type (e.g. work plan, hole, pocket, etc.) (Figure 7.11).
Chapter 7 – System Implementation and Case Studies

After the top level of DPacket is determined, the DPacket system converts STEP-NC classes into java Vector classes (Figure 7.7). The Vector class implements an expandable array of objects. It contains components that can be accessed using an integer index. The size of a Vector can grow or shrink as needed, to accommodate addition and removal of items after the Vector has been created. It is particularly suitable for playing the role of intermediate data model when STEP-NC entity instances are added or detached to the DPacket structure. Thus, all the information of the top entity is delivered to a new DPacket tree structure and the connections between this entity and others are tagged and recorded in the system.

```java
final Vector<Object> RootNode =
    new TreeNodeVector<Object>(testIn.getLocalDomain().getName());
AddAttributes(testIn, RootNode);
```

Figure 7.12 Converting Entity Instances into Java Vectors

After the top level of data is collected, the process of extracting the next level of product information starts. The PLANAR_FACE feature “PLANAR FACE1” is set as the new Parent Node and transmitted to the DPacket. Then the system continues the loop of data-collection (Figure 7.13).

```java
public class GetADatapacketV2All {
    public static int j = 1;
    public static void addEntity(EntityInstance in, Model mod) {
        PopulationBase pop = mod.getPopulation();
        pop.addInstance(in);
        EntityIDTable order = EntityIDTable.forNameModel(mod);
        try {
            order.setId(BigInteger.valueOf(j), in);
            j++;
        } catch (DuplicateIDException ex) {
            Logger.getLogger(GetADatapacketV2All.class.getName()).log(Level.SEVERE, null, ex);
        } catch (DuplicateInstanceException ex) {
            Logger.getLogger(GetADatapacketV2All.class.getName()).log(Level.SEVERE, null, ex);
        }
        Attribute sd[] = in.getLocalDomain().getAllAttributes();
        //EntityInstanceSet set = (EntityInstanceSet) mw.get_datapacket();
    }
```

Figure 7.13 DPacket Collection Method

When all the required information is extracted, the reference relationships between the DPacket and the rest of the file are saved in the Connection Entity for future updating. Meanwhile, the DPacket data model is written into a new Part21 file and saved. In this way, the user is able to work with the requested data. As illustrated in Figure 7.14, the output file contains the exact amount of information requested by the user, and all the logic relationships between entity instances are kept intact. Different users are able to process the data regarding to the specific machining working step, namely “working step finish planar face”. The
domain specific information is processed in different data flows without affecting other parallel procedures or product project. After the users finished the work on the DPacket document, the modifications are synchronized in the source database.

To recap, the concept of DPacket was implemented and evaluated in this case study. With the help of Data Localisation/Integration Mechanisms, the CU is able to access a specific scope of data without affecting the rest. It provides more flexibility to a cooperative manufacturing environment and more possibilities of utilizing a standardized ICT protocol, i.e. STEP/STEP-NC.

7.4. Case Study 3: VFB-based Service Integration

To evaluate VFB and its service integration mechanism, a case study was carried out to test the VFB capability to integrate STEP-NC and Creo Parametric software. It is assumed that a CU has requested the use of a Creo Parametric application for mechanical analysis,
simulation and NC code generation. The product and NC code is maintained in the STEP-NC models in the SCloud. Since Creo Parametric does not support this format naturally, it is necessary to develop VFBs that transfer the features and manufacturing information into and out of Creo.

### 7.4.1. VFB Development

In a case where it is requested that the product model be viewed and edited in the Creo Parametric environment, it is necessary to map STEP manufacturing features to those of Creo and allocate parameters from the STEP-NC model to corresponding dimensions into the Creo part file. Three VFBs, therefore, need to be developed to integrate the STEP-NC data format with Creo Parametric. A data exchange mechanism is also built between Storage Cloud and Creo. Additionally, the CU requested to utilise the NC module of Creo Paramedic. Hence, the process data in the STEP-NC model also needs to be imported into Creo. A series of Cloud Services are organised that include a STEP-NC/Creo interpreter, Creo Parametric and a Creo/STEP-NC interpreter (Figure 7.15).

![Figure 7.15 VFB Case Study](image)

VFBs wrap these applications by manipulating the event/data flow. For example, when the user finishes with the STEP-NC/Creo processor and saves the file (Data-Out), this action is defined as “Even-Out” for this VFB and triggers the VFB algorithm. The Wrapper Agent inside VFB shuts down the application and feeds back the service progress to the SCM. After the SCM sends back the next command line, based on the service document, the next VFB, the Creo module, is launched, and the agents keep listening to the user activity until the next
trigger event is detected (Table 7.3). In this way, applications are easily controlled and integrated in the Cloud.

<table>
<thead>
<tr>
<th>Task</th>
<th>Event_In Trigger</th>
<th>Event_out Trigger</th>
<th>Application Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Work</td>
<td>Interface Message Displayed</td>
<td>File Loaded</td>
<td></td>
</tr>
<tr>
<td>STEP-NC file=&gt; Creo Compliant Classes</td>
<td>STEP-NC file loaded</td>
<td>Creo Compliant Classes created</td>
<td>STEP-NC/Creo Interpreter</td>
</tr>
<tr>
<td>Original Creo Classes =&gt; Updated Creo Classes</td>
<td>Creo Compliant Classes loaded</td>
<td>Updated Creo Compliant Classes created</td>
<td>Creo Parametric</td>
</tr>
<tr>
<td>Creo Compliant Classes =&gt; STEP-NC file</td>
<td>Creo Compliant Classes loaded</td>
<td>STEP-NC file created</td>
<td>Creo/STEP-NC Interpreter</td>
</tr>
<tr>
<td>Arrived at Destination</td>
<td>STEP-NC file archived in Storage Cloud</td>
<td>Interface Message displayed</td>
<td></td>
</tr>
</tbody>
</table>

The listening and control methods are implemented via Java command methods, e.g. `exec()`, `addShutDownHook()`, etc. After the VFB is registered in the Database, the SCM is able to manipulate the service executions via the standardised command messages. The software agent receives these commands via Internet and maps them to local programme languages. Then the service package is processed as requested by the SCM (Figure 7.16).

```java
public static void EventAdder(HashMap<String, String> eventlist) {
    eventlist.put("step", "d:\1.step");
    eventlist.put("notepad", "C:\\WINDOWS\\system32\\notepad.exe");
    eventlist.put("5", "rundll32 SHELL32.DLL,ShellExec_RunDLL  
    " + "Explorer.exe /select," + "d:\1.step");
}

public static String EventFinder(String in) {
    String out = null;
    HashMap<String, String> HM = null;
    EventAdder(HM);
    out = HM.get(in);
    return out;
}
```

Figure 7.16 Application Registration and Command
When a VFB is initially defined in the SACloud, a description document is generated in the Service Database, based on the Cloud_Service_Template model previously mentioned. By using a data structure compliant with the STEP standard, data portability between VFB resources and STs is realised. Thus the SCM is capable of searching for and choosing appropriate VFB solutions by mapping the user request to the Service Database.

In this way, Creo Parametric and related processers can be packed and integrated as a Virtual Service Combination. Based on the Service Template delivered by the SCM, the Preprocesser, Creo and Post-processer are launched for the user in sequence, to fulfil his or her request. Using the Java programme, a remote communication mechanism is built, based on socket-server protocol. The data flow among distributed devices is implemented in terms of document, command and message transmission (Figure 7.17).

In addition, a Web-based communication structure was built for ICMS (Figure 7.19). A socket-server connection bridges a user’s device and the Central Server. A user with accessibility is able to retrieve documents from the SCloud remotely over the Internet and archive them afterwards. Transmission speed was affected by network traffic, which reached 2MB/s during the implementation test. Thanks to the mobility of the Java language, the system can be easily implemented in different operating environments.
In addition, a real-time text-based message system was developed. For communication between users and providers, or collaborative work amongst participants, the system provides a messenger programme enabling users to send simplified data or messages over the Internet. Users are able to build point-to-point real-time communication through the interface with the ICMS infrastructure. In the future, more communication methodologies can be introduced to the system, e.g. Skype™, Google Chat, etc..

7.4.2. Interpreter Development

This said, the product document is defined in the STEP-NC neutral data format, which cannot be input into Creo Parametric directly. Hence, interpreters were also developed to evaluate the interoperability of the STEP standard in the Cloud environment. This tests the mobility of applications using VFBs and the re-usability of CMServices.
The interpreters run as an auxiliary application on Creo. The STEP-NC/Creo interpreter is in charge of importing product design and machining information into Creo, and the Creo/STEP-NC interpreter is responsible for exporting the results back to a STEP file. These interpreters were developed via J-Link interfaces of PTC. The interfaces are integrated with the Creo Parametric programme when they are registered in the Service Cloud. The VFB functions are visible in the ribbon user-interface of Creo Parametric (Figure 7.20).

7.4.2.1. Data Importing

The STEP-NC/Creo Interpreter allows the user to load a STEP Part 21 file and transfer it to Creo. Firstly, a new Creo design part without any design or manufacturing information is created by the user. After the interpreter function is launched, a dialogue box appears, asking for the STEP-NC file the user wishes to import. The Creo part file is then populated with features mapped from the STEP-NC file by the VFB programme. The interpreter VFB follows the original definition of the product. The new Creo design part inherits the feature constraints and the coordinate system. As shown in Figure 7.21, the programme first sets the default coordinate system of the part as the “machining zero” point. It then proceeds to create the Setup coordinate system, the Workpiece coordinate system and the workpiece solid that are defined sequentially in the STEP-NC file.

Figure 7.21 Mapping Coordinate Systems
7.4.2.2. Workpiece Importing

It needs to be noted that in a STEP-NC data format, the geometry of the workpiece is not regarded as essential information. However, in Creo Parametric, a solid block needs to be provided before the machining features are placed, since the features need to be represented via material removal from, or addition to, the workpiece block. Therefore, if workpiece geometry is not provided in the original document, the VFB programme prompts the user for information regarding the “base” model. The user is able to either load another solid model to act as the block, or enter the placement and dimensions to allow a block workpiece to be generated by the programme. Figure 7.22 shows the user-interface when workpiece geometry is not detected in the STEP-NC file.

![Figure 7.22 Configurations for Workpiece Information](image)

The next step involves populating the base solid in Creo with features from the STEP-NC model. The programme iterates all the machining working steps contained in the STEP-NC data structure, including the project’s main work plan and sub-work plans, to process their associated machining features. The result of an imported STEP-NC model is shown in Figure 7.23. The product model from ISO 14649-11 example 1 was successfully imported into Creo via VFB. The design model tree containing all the features is shown on the left with the 3D view on the right.
7.4.2.3. Building STEP-NC Features in Creo

At this phase of development, it is important to clarify the differences between Creo and STEP-NC data models. Creo Parametric is a feature history-based CAD application where the order of building features is essential for the system. The Creo model follows the “model tree” structure, where features are generated procedurally. In history-based CAD software such as Creo, standard practice in constructing a part involves identification of the design intent. Certain features, by necessity, precede other more dependent features in the design process [213]. Those dependent features heavily rely on the features previously defined, which play the role of dimensional and geometric references. This is known as a parent-child relationship. The dependency is important, in particular when changes are processed across the model. After a parent feature in a part is changed, all child features are dynamically altered to reflect the modifications in the parent feature, to maintain design consistency.

In a STEP-NC data model, however, there is no essential recognition of feature constraints. The features are not heavily dependent on one another. Manufacturing features (e.g. pockets, holes, slots, etc.) make up the geometry of the 3D model. The locations of these features are defined by the attribute Feature_Placement, which dictates the location and orientation of the feature local coordinate system, relative to the Workpiece coordinate system. The profile shape, depth and other relevant dimensions of a particular feature are then defined by attributes belonging to specific subtypes as discussed in Chapter 6. This data structure results in a model where features normally reference the Workpiece, instead of relying on one another. The model can also be compared to that of a history-free CAD model where the order of features generated is not required.
7.4.2.4. Mapping Manufacturing Features to UDFs

In the interpreter VFB, the manufacturing features are converted into Creo User-Defined-Features (UDFs). UDF is the functionality that allows a user to place a template feature group provided by Creo. A UDF can consist of one or more features grouped together. An example of a UDF implementation for a closed pocket can be seen in Figure 7.24. The name of the feature group “POCKET1” follows the original identifier in the STEP-NC document.

![Figure 7.24 Contents of a Closed Pocket UDF](image)

In the interpreter VFB, two5d manufacturing features are converted into UDFs. Since two5d features are prismatic parts, the majority of the features can be represented in Creo using the extrude feature. However, extrusions in Creo must be done on a reference plane, which can either be a surface or a datum plan. Therefore, it is necessary to build a reference system additionally to the STEP-NC model. The parent feature (CS2) of the UDF feature group maintains its local coordinate system. Consequently, only the workpiece coordinates need to be referred to when placing the UDF. The datum plane DTM9 is a plane where Sketch1 is created. The sketch represents the profile of the closed pocket, which in this case is defined in the STEP-NC entity as a rectangular closed profile. Extrude 2 is an extrusion of the profile, described in Sketch 1. Finally, round corners are utilised to map the profile radius. In this way, the gaps caused by feature constraints are bridged by the interpreter. The interpreter generates the features and reference elements needed in Creo.

The model definitions are maintained in the UDF counterparts. This is done via necessary manipulation of the STEP-NC attributes mapped to various types of Creo features. An angled
taper at the end of a round hole, for example, is constructed in Creo using a sketch and the revolving feature on it, about the vertical axis. In STEP-NC, however, the geometry of the hole is defined by its attributes, i.e. diameter, depth and angle. Figure 7.25 shows the structure of the tapped feature definition of the angle taper feature in ISO standards.

![Figure 7.25 Diagram for Angle Taper](image)

Figure 7.25 Diagram for Angle Taper

Figure 7.26 shows a Creo sketch that is mapped from the STEP-NC definition. When the taper feature is imported, the angle and depth attributes from the feature entity are directly mapped to their corresponding dimensions in the Creo sketch. The diameter attribute is halved to match the radius dimension. Minimal manipulation of attributes is required as this simplifies the process of importing and exporting.

![Figure 7.26 Sketch Used in a UDF of a Round Hole with a Tapered End](image)

Figure 7.26 Sketch Used in a UDF of a Round Hole with a Tapered End

### 7.4.2.5. Compound Features and Replicate Features

A compound feature is a feature consisting of two or more features in the STEP-NC data model. The individual features reference the group’s (i.e. compound feature’s) local coordinate system rather than the workpiece’s. A replicate feature, on the other hand, is
assembled from a number of similar features (e.g. a group of holes). The machining features are built based on the compound feature so that they will not come into another category by mistake. However, there is no compound or replicate data structure in the UDF definitions. With the help of the interpreter VFB, they are represented as grouped feature elements.

Figure 7.27 shows the product example 2 from ISO14649-11. When a compound feature is processed by the VFB, the programme first places its local coordinate system, CS4, for “Compound_Feature_Hole1”. Then the elements of the compound feature are cycled through and mapped into the UDF definitions. The sub-elements (e.g. straight holes and their tapered ends) are generated, based on the compound feature coordinate systems. Lastly, the VFB groups the feature’s local coordinate system as well as the elements. So they are compliant with the reference logic of Compound_Feature_Hole1 in the original STEP-NC document.

Figure 7.27 Creo Compound Feature Mapped to Creo Feature Group (Appendix I.B)

An example of replicate features is shown in Figure 7.28. A general pattern is created based on UDF, namely HOLES_4_5_6_AND_7. Like the compound feature, the parent feature is the local coordinate system (CS3). The replicated features of this pattern are Compound_Holes which contain four sub-elements in the same shape. The feature group
HOLES_4_5_6_AND_7_ELEMENT0 contains the first group of replicate instances, while HOLES_4_5_6_AND_7_ELEMENT1 is the second group. Each group of instances has its own placement location and a copy of the replicated feature, which matches a set containing a hole, a tapered end and their setups. The pattern’s local coordinate system and all of its replicate feature instances are then bonded to form a UDF feature group.

Figure 7.28 Replicate Feature Example

7.4.2.6. Data Exporting

The STEP-NC data is mapped and imported into Creo Parametric. In order to realise the ability to export a Creo part file back to the STEP-NC data format, all the information of the STEP model must be retained. This is done by storing a copy of the entire Part 21 file in the Creo temporary format. Since Creo allows the storage of external information in the form of integers, doubles or strings, the interpreter integrates all the data into the Creo format.

The purpose of the export function is to allow changes made to the STEP model in Creo to be translatable back to the STEP format. As described before, a copy of the original STEP file is stored in the Creo part file when the model is first imported. This copy is read and its STEP model is loaded by the programme when the export option is selected. The programme then
iterates through the Creo feature groups in the model tree, and maps the dimensions of the placed UDFs in Creo, back to the original features in the STEP model. In this way, all the changes made in the Creo are reflected back to the STEP-NC format without any data conflict. The STEP model is then written into the chosen Part 21 file, forming a bi-directional data-flow between CMService and SCloud.

7.4.2.7. Machining Process

A method of transferring manufacturing feature information has now been developed. The VFB is also able to process machining operations. The operations include plane rough/finishing milling, side rough/finish milling, bottom and side rough/finish milling, and multistep drilling. When a STEP-NC model is imported into Creo, the interpreter first generates the product geometry and features. Then it processes the “main work plan” and extracts the operations from the machining working steps contained in the work plans. This is achieved by mapping the machining process to the UDF annotations that hold Creo machining templates. Since Creo’s NC module does not support UDFs directly, the user is able to create a new Creo manufacturing document and import the templates that maintain the STEP-NC machining information as a reference model. Then all the information from the reference model can be imported into the Creo native model, with the help of the Creo process manager (Figure 7.29).

Figure 7.29 Extracting Manufacturing Operations from a Reference Model
After the operation template is extracted, its parameters can be updated by using the Import Process Data function as illustrated in Figure 7.20. The tool path can be generated based on the data source from the STEP-NC document. Since there is a major difference between STEP-NC machining strategies and Creo strategies, the prototype system interprets the basic machining setups, e.g. retract offset distance and safe planes. Detailed parameters can be realised in a future development. However, another limitation of the interpreter is that the cutting tool interface is not supported by Creo Parametric. Currently, default cutting tools are utilised in the manufacturing process. In the future, it will be possible to develop a cutting tool library, based on Creo Parametric, when it is integrated in the Cloud. When Creo Parametric is requested by a user, its cutting tool library that provides appropriate tools for machining operations can be loaded.

![Figure 7.30 Tool Path for a Plane Rough Milling Operation](image)

7.5. Recap

This chapter describes the key features of the ICMS prototype that have been developed. The system was developed in a NetBeans IDE 7.1 environment, using Java programming language. Multiple software development tools and packages were utilised and integrated into the system. Web-based interfaces were developed via Dreamweaver to interact with Cloud Users. Standardised databases were built to maintain the Cloud Service and resource data. ST-Developer was used to convert EXPRESS schema into Java classes, which were then integrated into the CMService and VFBs. The Data Packet mechanism was realised via Java programmes. To evaluate the interoperability of the VFB-based Cloud Service, Creo Parametric was chosen as the test application, integrated in the Manufacturing Cloud. VFB
functions were developed in the JDK and JRE environment, along with a Java-based interpreter coupled with Creo J-link interfaces. Three case studies were carried out to test the proposed CManufacturing model and related integration methodologies (Case Study 1). The manufacturing services were integrated as CMServices, which maintain data-interoperability (STEP-NC-based Data Packet) and process-interoperability (Cloud-based service integration via VFBs).
This chapter concludes the work presented in this thesis. In the first section a recap of the study is presented. Next is the author’s vision of the main contributions to the domain field. In addition, the chapter addresses future research opportunities.

8.1. Conclusions

The work presented in this current study deals with an interoperable manufacturing model, in conjunction with Cloud technology, from a new perspective, i.e. CManufacturing. The main focus is on an Interoperable Cloud-based Manufacturing System that provides manufacturing resources in terms of service.

8.1.1. Summary of the Research

The modern manufacturing industry calls for a new generation of integration models that are more interoperable, intelligent, adaptable and distributed, so that manufacturers are able to respond quickly and effectively to frequent changes in the global market. Manufacturing enterprises will be able to co-operate with each other to seek stronger competiveness and better resources.

However, collaboration and interaction between business partners are limited by interoperability issues caused by the heterogeneous environment in manufacturing industry. The supply chain is conducted by multiple tiers of suppliers/contractors, who deploy software/hardware applications from different vendors. These applications were developed using different platforms and programme languages, which leads to a variety of communication methodologies, i.e. different product documents, process data, protocols, standards, etc. Interaction between business partners is difficult, in particular when multiple manufacturing enterprises are involved in the same production task.
This research was motivated by industry requirements for both process interoperability and data interoperability. There is a strong need for integration and interaction methodologies in the background of the globalised manufacturing business. This is evidenced by a comprehensive number of systems, which have been developed over less than two decades. A thorough review of interoperable manufacturing systems was first carried out. It was found that most research had focused on either promoting conversional data-exchange methods to communicate domain-specific applications (data-centric), or organising production procedures that are “strung out” by the production processes (process-centric). There is still a need to deliver feasible solutions to realise both data and process interoperability. To develop a manufacturing solution that bridges the gaps between manufacturing participants and their manufacturing facilities, this research investigated a different manufacturing model that aggregated Cloud technology, international standards and interoperable manufacturing systems.

The work presented in this study provides a manufacturing model that organises resources and capabilities at higher level, i.e. the Cloud Service level. It was noted that in recent research trends, Cloud technology changed the way of computing and service processing. In the manufacturing context, at the same time, there is a recognised need to integrate heterogeneous manufacturing facilities. The significance of the ICMS infrastructure lies in the fact that it can organise operational processes ahead of the process integration stage. The system is able to consider trustworthy strategies to achieve better service performance and avoid interoperability problems in advance.

In ICMS, distributed manufacturing resources are recognised as MCapabilities and their specifications are digitized in the MCloud in terms of Cloud Service Modules. With shared capacities, knowledge and competencies, it is possible for an enterprise to take on more substantial projects that have otherwise been impossible for a single company to do. It is particularly valuable for countries like New Zealand, in which most manufacturing enterprises are SMEs. In contrast with conventional interoperable manufacturing approaches, ICMS integrates manufacturing facilities with full knowledge and scalable specifications. The resources are understood and modelled from the Cloud Service viewpoint. When a service package is scheduled, the capabilities and limitations are analysed by the supervision kernel (SCM). Interoperability issues are foreseen, thanks to the comprehensive database, and avoided via arrangements for additional services, bridging different resources or alternative
modules. Process interoperability is achieved through the coordination of the intelligent SCM.

In the ICMS environment, users can submit production requests in terms of Cloud Service queries remotely, via the network. With the help of the SCM mechanism, the manufacturing objectives are mapped to scalable service modules encapsulated in the Manufacturing Cloud. The users do not need to recognise a service’s carrier or whereabouts. Interaction between users and Service Providers is minimised. Production goals can be achieved without on-site visits being necessary. ICMS leads to the next generation of “Design Anywhere, Manufacture Anywhere”.

At the data level, the emergence of STEP, STEP-NC, MTConnect standards and the Cloud opens up new possibilities that could resolve the problem of manufacturing data integration. They can maintain complete product and manufacturing information, as well as data exchange methods across the design-machining chain. These information methodologies supersede the limitations of legacy manufacturing approaches that only provide descriptions of domain-specific data, instead of allowing more intelligent and interoperable interactions to be conducted. Therefore, standardised neutral data models can extend the scope of information management, thus guaranteeing the data interoperability of ICMS.

From this research, eight papers have been produced, consisting of four journal papers, one book chapter and three conference papers. The first and second papers introduced the proposed research framework. The first was presented at the 2010 international conference on Flexible Automation and Intelligent Manufacturing, Oakland, California, US, and the second was published in the *Journal of Enterprise Information Systems*. To implement the integration methodology and promote the integration environment to a higher level, Cloud technology was merged into the system. The results of the study were published as a book chapter in *Cloud Manufacturing: Distributed Computing Technologies for Global and Sustainable Manufacturing* by Springer. The concept of the Data Packet was reported at the international conference on Digital Enterprise, Technology 2011, and extended as a journal paper that was submitted to the *International Journal of Computer Integrated Manufacturing* as recommended. Comprehensive review work was conducted to identify research gaps and motivations. Based on the review work and the latest CManufacturing research proposals, a journal paper was prepared for the *International Journal of Production Research*. Following this, the results of the Cloud-based SCM and the VFB mechanism were described in a paper
Chapter 8 – Conclusions and Future Work


8.1.2. Intellectual Contributions

One of the major objectives of this research has been the formulation and implementation of ICMS, which integrates existing manufacturing capabilities and resources. The following major intellectual contributions concerning the achievements obtained in realisation of the above objectives are outlined here.

**Contribution 1: The Interoperable Manufacturing Model based on Cloud**

The development of Cloud technology has considerably impacted the IT market. It provides new service models in the computing paradigm. Inspired by Cloud, this research adopts the Cloud technology in the manufacturing context, thus forming a novel Everything as a Service (XaaS) manufacturing model, namely the Interoperable Cloud-based Manufacturing System. ICMS provides an integration methodology that supports distributed and collaborative manufacturing, i.e. CManufacturing. Unlike B2Cloud or C2Cloud, it offers an X2C model, in which manufacturing capabilities and resources are shared in a large pool of resources that are provisioned in terms of scalable Cloud Manufacturing Services. From the manufacturers’ perspective, the CManufacturing model strengthens the competitiveness of enterprises through an extended business network. For customers, their manufacturing tasks are accomplished via a larger scale of higher quality service solutions.

**Contribution 2: Classification of Interoperable Systems and Cloud Components**

The development of interoperable manufacturing systems is a multi-disciplinary act and the literature shows that there are many types of approaches, interpretations, terminologies and scopes involved. Therefore, current research takes various synergistically encompassed aspects of each approach used, by developing a classification to present the interoperable environment in either a data-centric or process-centric category. In this manner, one can easily understand and correctly analyse interactive systems more comprehensively and effectively.

In the CManufacturing context, different Cloud participants were also categorised and defined in this research. The role of a Cloud stakeholder is clearly explained in comparison with those in a Cloud Computing environment. Additionally, manufacturing resources and
facilities are described as a different level of Cloud elements: MResource, MCapability and CMService in the CManufacturing perspective. The recognition of production features provides a clear classification when new resources are integrated in the MCloud. Moreover, Cloud Users are divided into two categories, i.e. Enterprise User and Customer User. Cloud consumers are treated appropriately with different service mechanisms.

**Contribution 3: Intelligent Cloud Supervision Mechanism**

In recent years, the concepts and frameworks of CManufacturing systems have been proposed world-wide. However, there is no coordinating mechanism that guides and manages Cloud Manufacturing Services. In this research, the service packages are scheduled and processed by the SCM mechanism at an upper level, in which process interoperability is guaranteed. Intelligent and optimised decisions can be made via the SCM. It offers Cloud users a wide field of candidate solutions. For Service Providers, the SCM arranges the Manufacturing Resources based on their status, which is dynamically updated. Hence the SCM increases the utilisation rate and improves enterprise performance globally. Additionally, the workload of different facilities is also balanced.

**Contribution 4: VFB Mechanism Supporting Resource Integration**

In ICMS, manufacturing facilities and materials are defined as MResources. To integrate MResources as MCapabilities in the Manufacturing Cloud, Function Block and a software agent are merged, forming Virtual Function Blocks. The VFB mechanism provides the methodology to integrate both soft resources and hard resources. The soft resources are packaged by the software agents within VFBs, while the hard resources are integrated with the help of event-driven Function Block devices and MTConnect. The VFB mechanism provides a rich methodology to integrate the manufacturing facilities. It is especially suitable for a distributed environment like CManufacturing and offers a flexible mechanism that integrates manufacturing facilities in the Cloud. The VFB provides a universal methodology for Cloud Service integration. As an example of a commercial software package, Creo Parametric was integrated as a Cloud Service with the help of VFB, along with pre-processor and post-processor VFBs, to evaluate the mechanism.
Chapter 8 – Conclusions and Future Work

Contribution 5: Extended STEP and STEP-NC Data Models Supporting CManufacturing and Data Interoperability

The utilisation of various products/syntaxes leads to big interoperability issues. In ICMS, STEP/STEP-NC data models are utilised to describe the information along the supply chain. International standards offer a comprehensive data-description mechanism from design to manufacture. Moreover, the current international standards are extended to the CManufacturing paradigm, e.g. Cloud Service, Service Provider, Service Quarry, VFB, etc. These models offer an explicit vision of Cloud Service and related functionalities. Thus the CManufacturing environment is integrated with international standards. A unified data environment has been developed that covers all production stages from raw material to final product, from low-level process data to high-level Cloud Service strategy. Attempts to implement international standards in the Cloud have realised interoperability between the Manufacturing Cloud and domain-specific applications. In addition, the Creo/STEP-NC interpreter was also developed to bridge the gap between commercial software tools and the standardised Cloud environment. The design and manufacturing data are ported between Creo and the STEP-NC data format, forming a bi-directional communication flow. This is evidence of the feasibility of a STEP-compliant Cloud environment. Moreover, integrating Creo Parametric in the MCloud has created the opportunity to invite more manufacturing resources into Cloud, for example, PTC Windchill, Arbortext, NC machines, etc.

Contribution 6: Advanced Data Sharing Mechanism

Even though the utilisation of STEP/STEP-NC standards brings data interoperability to the Cloud, the drawbacks of an integrated data format can still be observed. Large project documents and related data-management difficulties have been reported in recent years. Hence the Data Packet mechanism was developed, based on the object-oriented nature of STEP. Thanks to the Data Packet sharing and integrating mechanism, Cloud Service stakeholders can be supported by the right amount of data, without the risk of confidentiality and synchronisation issues. It assists the data management of big production objectives. Thus smooth communication and information exchange can be realised with the DPacket mechanism. Working with VFB, the DPacket mechanism plays the role of pre-/post-processor of Cloud Service applications. It enables a flexible data-exchange environment in the MCloud. Additionally, a reasonable size and number of documents is also essential, especially for a web-based data transmission environment like CManufacturing. The DPacket
mechanism offers an opportunity to minimise data size, which is particularly suitable for communications and interactions over the network.

### 8.2. Recommended Future Work

In recent research trends, CManufacturing is a relatively new area that needs to be researched. There is a recognised need to incorporate manufacturing processes within supplier/contractor networks. This research has developed a fundamental system infrastructure along with key technologies, e.g. facility integration, data exchange, and service management. The prototype system and essential algorithms have been implemented to evaluate the proposals. However, there remains a long road to realising a complete Cloud-based supply chain. In order to further test the comprehensiveness of the system, industry implementation is indispensable. Therefore, further extended research opportunities could be carried out in the following areas:

1. More types of manufacturing facilities will be integrated in the Cloud. The current research encapsulates typical manufacturing processes (e.g. Creo, CAD and CAM modules, milling, welding and grinding processes) as test manufacturing services. Other software and hardware facilities could be integrated by VFBs to include more manufacturing phases of production, for instance, turning process, inspections, additive manufacturing, etc.

2. The existing SCM mechanism utilises the iterative method to explore the most cost-effective or time-effective service combination. However, such techniques would be computationally intensive in time and space when there are large numbers of candidate applications in the resource pool. Other techniques, such as artificial neural networks and genetic algorithms, could be implemented in the SCM intelligence to improve the computational performance.

3. The Cloud management performance can be improved with the help of the ERP. From a wide viewpoint, Manufacturing Cloud could be considered a virtual organisation capable of providing manufacturing-related services to fulfil user requests. ERP systems have been studied for years to support information management across the entire organisation. Combined with the SCM supervision kernel, it will be possible to improve the service quality and management efficiency of ICMS.

4. The Cloud resource pool could be further extended. The current Cloud Service and Service Provider databases have been built based on the existing research and partnership network of this research project. In future, more Cloud participants,
especially international stakeholders, could be involved to extend the scope of ICMS and evaluate the DAMA concept of CManufacturing.

5. Manufacturing Applications on mobile devices have gained more weight in the manufacturing industry these years. On-site tasks, e.g. measurement, product tracking, warehouse management, etc., could be aided by the wireless network connected to the ICMS. Mobile devices such PDA, RFID, GPS and smart phones could be integrated to the Cloud to further strengthen the flexibility and efficiency of manufacturing systems.
References


References

194


Appendix I

ICMS Product Files
A. Updated STEP Part 21 File for Example 1 of ISO 14649 - Part 11
(This example document is utilized in the Section 7.2 and 7.3)

ISO-10303-21;

HEADER;
FILE_DESCRIPTION(('ISO 14649-11 EXAMPLE 1',
'SIMPLE PROGRAM WITH A PLANAR_FACE, A POCKET, AND A ROUND_HOLE'),
'1');
FILE_NAME('EXAMPLE1.STP',
'2012-05-02',
('Xi Vincent Wang', 'YONG TAK HYUN', 'JOCHEN WOLF'),
('WZL, RWTH-AACHEN'),
'S',
'ISO 14649',
'S);

FILE_SCHEMA(('COMBINED_SCHEMA'));
ENDSEC;
DATA;
#1= PROJECT('ICMS EXECUTE EXAMPLE1',#2,(#4),$,$,$);
#2= WORKPLAN('MAIN WORKPLAN',(#10,#11,#12,#13,#14),$,#8,$);
#4= WORKPIECE('SIMPLE WORKPIECE',#6,0.010,$,$,$,(#66,#67,#68,#69));
#6 = MATERIAL('ST-50', 'STEEL', (#7));
#7 = PROPERTY_PARAMETER('E=200000N/M2');
#8 = SETUP('SETUP1', #71, #62, (#9));
#9 = WORKPIECE_SETUP(#4, #74, $, $, $);
#10 = MACHINING_WORKINGSTEP('WS FINISH PLANAR FACE1', #62, #16, #19, $);
#11 = MACHINING_WORKINGSTEP('WS DRILL HOLE1', #62, #17, #20, $);
#12 = MACHINING_WORKINGSTEP('WS REAM HOLE1', #62, #17, #21, $);
#13 = MACHINING_WORKINGSTEP('WS ROUGH POCKET1', #62, #18, #22, $);
#14 = MACHINING_WORKINGSTEP('WS FINISH POCKET1', #62, #18, #23, $);
#15 = PLANAR_FACE('PLANAR FACE1', #4, (#19), #77, #63, #24, #25, $);
#16 = ROUND_HOLE('HOLE1 D=22MM', #4, (#20, #21), #81, #64, #58, $);
#17 = CLOSED_POCKET('POCKET1', #4, #22, #23, #84, #65, $, #27, #35, #37, #28);
#18 = PLANE_FINISH_MILLING($, $, 'FINISH PLANAR FACE1', 10.000, $, #39, #40, #41, #60, #61, #42, 2.500, $);
#19 = DRILLING($, $, 'DRILL HOLE1', 10.000, $, #44, #45, #41, $, $, $, $, $, #46);
#20 = REAMING($, $, 'REAM HOLE1', 10.000, $, #47, #48, $, $, $, $, #49, .T., $);
#21 = BOTTOM_AND_SIDEROUGH_MILLING($, $, 'ROUGH POCKET1', 15.000, $, #39, #50, #41, $, $, $, $, $, 51.2, #50, 5.000, 1.000, 0.500);
#22 = BOTTOM_AND_SIDE_FINISH_MILLING($, $, 'FINISH POCKET1', 15.000, $, #39, #52, #41, $, $, $, $, $, $, $, #53, 2.000, 10.000, $, $);
#23 = LINEAR_PATH($, #54, #55);
#24 = LINEAR_PROFILE($, #57);
#25 = THROUGH_BOTTOM_CONDITION();
#26 = PLANAR_POCKETBOTTOM_CONDITION();
#27 = GENERAL_CLOSED_PROFILE($, #59);
#28 = TAPERED_ENDMILL(#30, 4, $, $, $, $);
#29 = MILLING_TOOL_DIMENSION(20.000, $, $, $, 1.500, $, $);
#30 = TWIST_DRILL(#32, 2, .RIGHT., .F., 0.840);
#31 = MILLING_TOOL_DIMENSION(20.000, 31.000, 0.100, 45.000, 2.000, 5.000, 8.000);
#32 = TAPERED_REAMER(#34, 6, $, $, $, $);
#33 = MILLING_TOOL_DIMENSION(22.000, $, $, $, $, $, $);
#34 = TOLERANCED_LENGTH_MEASURE(1.000, #36);
#35 = PLUS_MINUS_VALUE(0.100, 0.100, 3);
#36 = TOLERANCED_LENGTH_MEASURE(10.000, #38);
#37 = PLUS_MINUS_VALUE(0.100, 0.100, 3);
#38 = MILLING_CUTTING_TOOL('MILL 20MM', #29, (#125), 80.000, $, $);
#39 = MILLING_TECHNOLOGY(0.040, .TCP., #12, 12.000, $.F., $.F., $.F., $);
#40 = MILLING_MACHINE_FUNCTIONS(.T., $, $, .T., $, $, $, $);
#41 = BIDIRECTIONAL_MILLING(5.000, .T., #43, .LEFT., $);
#42 = DIRECTION('STRATEGY PLANAR FACE1: 1.DIRECTION', 0.000, 1.000, 0.000);
#43 = MILLING_CUTTING_TOOL('SPIRAL_DRILL_20MM', #31, (#126), 90.000, $, $);
#44 = MILLING_TECHNOLOGY(0.030, .TCP., 16.000, $.F., $.F., $.F., $);
#45 = DRILLING_TYPE_STRATEGY(75.000, 50.000, 2.000, 50.000, 75.000, 8.000);
#47= MILLING_CUTTING_TOOL('REAMER_22MM',#33,(#127),100.000,$,$);
#48= MILLING_TECHNOLOGY(0.030,.TCP.,$18.000,$.,F.,F.,F.,$);
#49= DRILLING_TYPE_STRATEGY($,$,$,$,$,$);
#50= MILLING_TECHNOLOGY($,.TCP.,$20.000,$.,F.,F.,F.,$);
#51= CONTOUR_BIDIRECTIONAL($,$,$,$,$,$);
#52= MILLING_TECHNOLOGY($,.TCP.,$20.000,$.,F.,F.,F.,$);
#53= CONTOUR_PARALLEL(5.000.,T.,CW.,CONVENTIONAL.);
#54= TOLERANCED_LENGTH_MEASURE(120.000,#56);
#55= DIRECTION('COURSE OF TRAVEL DIRECTION',(0.000,1.000,0.000));
#56= PLUS_MINUS_VALUE(0.300,0.300,3);
#57= NUMERIC_PARAMETER('PROFILE LENGTH',100.000,'MM');
#58= TOLERANCED_LENGTH_MEASURE(22.000,#56);
#59= POLYLINE('CONTOUR OF POCKET1',(#121,#122,#123,#124,#121));
#60= PLUNGE_RAMP($,45.000);
#61= PLUNGE_RAMP($,45.000);
#62= ELEMENTARY_SURFACE('SECURITY PLANE',#73);
#63= ELEMENTARY_SURFACE('PLANAR FACE1-DEPTH PLANE',#80);
#64= ELEMENTARY_SURFACE('DEPTH SURFACE FOR ROUND HOLE1',#83);
#65= ELEMENTARY_SURFACE('DEPTH SURFACE FOR POCKET1',#94);
#66= CARTESIAN_POINT('CLAMPING_POSITION1',(0.000,20.000,25.000));
#67= CARTESIAN_POINT('CLAMPING_POSITION2',100.000,20.000,25.000));
#68= CARTESIAN_POINT('CLAMPING_POSITION3',0.000,100.000,25.000));
#69= CARTESIAN_POINT('CLAMPING_POSITION4',100.000,100.000,25.000));
#71= AXIS2_PLACEMENT_3D('SETUP1',#95,#96,#97);
#73= AXIS2_PLACEMENT_3D('PLANE1',#98,#99,#100);
#74= AXIS2_PLACEMENT_3D('WORKPIECE',#101,#102,#103);
#77= AXIS2_PLACEMENT_3D('PLANAR FACE1',#104,#105,#106);
#80= AXIS2_PLACEMENT_3D('PLANAR FACE1',#107,#108,#109);
#81= AXIS2_PLACEMENT_3D('HOLE1',#110,#111,$);
#83= AXIS2_PLACEMENT_3D('HOLE1',#112,#113,#114);
#84= AXIS2_PLACEMENT_3D('POCKET1',#115,#116,#117);
#94= AXIS2_PLACEMENT_3D('POCKET1',#118,#119,#120);
#95= CARTESIAN_POINT('SETUP1: LOCATION ',(150.000,90.000,40.000));
#96= DIRECTION(‘ AXIS ’,(0.000,0.000,1.000));
#97= DIRECTION(‘ REF_DIRECTION’,(1.000,0.000,0.000));
#98= CARTESIAN_POINT('SECPLANE1: LOCATION ',(0.000,0.000,30.000));
#99= DIRECTION(‘ AXIS ’,(0.000,0.000,1.000));
#100= DIRECTION(‘ REF_DIRECTION’,(1.000,0.000,0.000));
#101= CARTESIAN_POINT('WORKPIECE1:LOCATION ',(0.000,0.000,0.000));
#102= DIRECTION(‘ AXIS ’,(0.000,0.000,1.000));
#103= DIRECTION(‘ REF_DIRECTION’,(1.000,0.000,0.000));
#104= CARTESIAN_POINT('PLANAR FACE1:LOCATION ',(0.000,0.000,5.000));
#105= DIRECTION(‘ AXIS ’,(0.000,0.000,1.000));
#106= DIRECTION(' REF_DIRECTION', (1.000,0.000,0.000));
#107= CARTESIAN_POINT('PLANAR FACE1:DEPTH', (0.000,0.000,-5.000));
#108= DIRECTION(' AXIS ', (0.000,0.000,1.000));
#109= DIRECTION(' REF_DIRECTION', (1.000,0.000,0.000));
#110= CARTESIAN_POINT('HOLE1: LOCATION ', (20.000,60.000,0.000));
#111= DIRECTION(' AXIS ', (0.000,0.000,1.000));
#112= CARTESIAN_POINT('HOLE1: DEPTH ', (0.000,0.000,-30.000));
#113= DIRECTION(' AXIS ', (0.000,0.000,1.000));
#114= DIRECTION(' REF_DIRECTION', (1.000,0.000,0.000));
#115= CARTESIAN_POINT('POCKET1: LOCATION ', (45.000,110.000,0.000));
#116= DIRECTION(' AXIS ', (0.000,0.000,1.000));
#117= DIRECTION(' REF_DIRECTION', (-1.000,0.000,0.000));
#118= CARTESIAN_POINT('POCKET1: DEPTH ', (0.000,0.000,-30.000));
#119= DIRECTION(' AXIS ', (0.000,0.000,1.000));
#120= DIRECTION(' REF_DIRECTION', (1.000,0.000,0.000));
#121= CARTESIAN_POINT('P1', (0.000,0.000,0.000));
#122= CARTESIAN_POINT('P2', (0.000,80.000,0.000));
#123= CARTESIAN_POINT('P3', (-50.000,80.000,0.000));
#124= CARTESIAN_POINT('P4', (-50.000,0.000,0.000));
#125= CUTTING_COMPONENT(80.000,$,$,$,$);
#126= CUTTING_COMPONENT(90.000,$,$,$,$);
#127= CUTTING_COMPONENT(100.000,$,$,$,$);
ENDSEC;
END-ISO-10303-21;
B. Updated STEP Part 21 File for Example 2 of ISO14649 - Part 11

(This example document is utilized in the Section 7.3)
#5 = WORKPLAN('WORKPLAN
   DRILLING', (#23, #24, #25, #26, #27, #28, #29, #30, #31, #32, #33, #34, #35, #36, #37), $, $, $);
#6 = WORKPLAN('WORKPLAN
   FINISHING', (#39, #40, #41, #42, #43, #44, #45, #46, #47, #48, #49, #50, #51, #52, #53), $, $, $);
#7 = WORKPIECE('PART 2', #13, 0.01, $, $, $, (#9, #10, #11, #12));
#9 = CARTESIAN_POINT('CLAMPING_POSITION1', (25., 25., -20.));
#10 = CARTESIAN_POINT('CLAMPING_POSITION2', (205., 25., -20.));
#11 = CARTESIAN_POINT('CLAMPING_POSITION3', (25., 155., -20.));
#12 = CARTESIAN_POINT('CLAMPING_POSITION4', (205., 155., -20.));
#13 = MATERIAL('FC200', 'CAST IRON', $);
#14 = SETUP('SETUP OF WORKPIECE', #342, #333, (#16));
#16 = WORKPIECE_SETUP(#7, #344, $, $, $, (#9, #10, #11, #12));
#17 = MACHINING_WORKINGSTEP('ROUGHING HEAD1', #333, #60, #61, $);
#18 = MACHINING_WORKINGSTEP('ROUGHING SIDE1', #333, #62, #63, $);
#19 = MACHINING_WORKINGSTEP('ROUGHING STEP', #333, #64, #65, $);
#20 = MACHINING_WORKINGSTEP('ROUGHING POCKET1', #333, #66, #67, $);
#21 = MACHINING_WORKINGSTEP('FINISH POCKET1', #333, #66, #68, $);
#23 = MACHINING_WORKINGSTEP('BORING1 HOLE1', #333, #100, #73, $);
#24 = MACHINING_WORKINGSTEP('DRILLING HOLE2', #333, #101, #76, $);
#25 = MACHINING_WORKINGSTEP('DRILLING HOLE3', #333, #102, #77, $);
#26 = MACHINING_WORKINGSTEP('BORING HOLE2', #333, #101, #78, $);
#27 = MACHINING_WORKINGSTEP('BORING HOLE3', #333, #102, #79, $);
#28 = MACHINING_WORKINGSTEP('COUNTER SINKING HOLE2', #333, #101, #80, $);
#29 = MACHINING_WORKINGSTEP('COUNTER SINKING HOLE3', #333, #102, #81, $);
#30 = MACHINING_WORKINGSTEP('DRILLING HOLE4', #333, #103, #82, $);
#31 = MACHINING_WORKINGSTEP('DRILLING HOLES5', #333, #104, #83, $);
#32 = MACHINING_WORKINGSTEP('TAPPING HOLES5', #333, #104, #84, $);
#33 = MACHINING_WORKINGSTEP('COUNTER SINKING HOLE4', #333, #103, #85, $);
#34 = MACHINING_WORKINGSTEP('COUNTER SINKING HOLE5', #333, #104, #86, $);
#35 = MACHINING_WORKINGSTEP('DRILLING HOLE6', #333, #105, #87, $);
#36 = MACHINING_WORKINGSTEP('REAMING HOLE6', #333, #105, #88, $);
#37 = MACHINING_WORKINGSTEP('COUNTER SINKING HOLE6', #333, #105, #89, $);
#39 = MACHINING_WORKINGSTEP('FINISH SIDE-SLOT', '#333, #123, #90, $);
#40 = MACHINING_WORKINGSTEP('BORING HOLE1', '#333, #100, #74, $);
#41 = MACHINING_WORKINGSTEP('COUNTER SINKING HOLE1', '#333, #100, #75, $);
#42 = MACHINING_WORKINGSTEP('ROUGHING1 T-SLOT', '#333, #124, #91, $);
#43 = MACHINING_WORKINGSTEP('ROUGHING2 T-SLOT', '#333, #124, #92, $);
#44 = MACHINING_WORKINGSTEP('FINISH T-SLOT', '#333, #124, #93, $);
#45 = MACHINING_WORKINGSTEP('FINISH2 T-SLOT', '#333, #124, #94, $);
#46 = MACHINING_WORKINGSTEP('ROUGHING SLOT1', '#333, #125, #95, $);
#47 = MACHINING_WORKINGSTEP('SLOT2 PLUNGE DRILLING', '#333, #126, #96, $);
#48 = MACHINING_WORKINGSTEP('ROUGHING SLOT2', '#333, #126, #97, $);
#49 = MACHINING_WORKINGSTEP('FINISH SLOT1', '#333, #125, #98, $);
Appendix I – ICMS Product Files

#50= MACHINING_WORKINGSTEP('FINISH SLOT2', #333, #126, #99, $);
#51= MACHINING_WORKINGSTEP('FINISH STEP',  #333, #64, #72, $);
#52= MACHINING_WORKINGSTEP('FINISH HEAD1',  #333, #60, #69, $);
#53= MACHINING_WORKINGSTEP('FINISH SIDE1',  #333, #62, #70, $);
#60= PLANAR_FACE('HEAD1', #7, (#61, #69), #348, #334, #178, #179, $, $);
#61= PLANE_ROUGH_MILLING($, $, 'ROUGH HEAD1',
    #10, $, #226, #229, #230, $, $, #184, $, $);
#62= GENERAL_OUTSIDE_PROFILE('SIDE1',
    #7, (#63, #70), #352, #335, #185, #186);
#63= SIDE_ROUGH_MILLING($, $, 'ROUGH SIDE1',
    #10, $, #231, #234, #235, $, $, #191, #4, #15, #0.5);
#64= STEP('STEP1', #7, (#71, #72), #355, #336, #192, $, ());
#65= BOTTOM_AND_SIDE_ROUGH_MILLING($, $, 'ROUGH STEP1',
    #65, $, #231, #234, #235, $, #196, #197, #198, #10, #18, #1, #5.2, $);
#66= OPEN_POCKET('POCKET1', #7, (#67, #68), #359, #337, (#199), $, #202, $, $, #203, $);
#67= BOTTOM_AND_SIDE_ROUGH_MILLING($, $, 'ROUGH POCKET1', #70, #237, #241, #235, $, $, #205, #10, #0.5, #0.5);
#68= BOTTOM_AND_SIDE_FINISH_MILLING($, $, 'FINISH POCKET1',
    #10, $, #231, #234, #235, $, $, #206, #10, #5, #0.5);
#69= PLANE_FINISH_MILLING($, $, 'FINISH HEAD1',
    #10, $, #226, #229, #230, $, #207, #208, #209, #2.5, $);
#70= SIDE_FINISH_MILLING($, $, 'FINISH SIDE1',
    #10, $, #231, #234, #235, $, $, #211, #1.5, #1, $);
#71= BOTTOM_AND_SIDE_ROUGH_MILLING($, $, 'ROUGH STEP1',
    #65, $, #231, #234, #235, $, #196, #197, #198, #10, #18, #1, #5.2, $);
#72= BOTTOM_AND_SIDE_FINISH_MILLING($, $, 'FINISH STEP1',
    #65, $, #231, #234, #235, $, #196, #197, #198, #5, #12.5, #5, $);
#73= MULTISTEP_DRILLING($, $, 'BORING1 HOLE1',
    #20, $, #242, #246, #230, #48, #0, #5, $, #15, #10, #1, $);
#74= BORING($, $, 'BORING2 HOLE1', #20, $, #252, #256, #230, #45, #36, #5, #0, #T, #1, #0.5);
#75= COUNTER_SINKING($, $, 'COUNTER SINKING HOLE1',  #30, $, #257, #261, #230, #1, #40, #5, $);
#76= DRILLING($, $, 'DRILLING HOLE2', #20, $, #262, #251, #230, $, #0, #5, $);
#77= DRILLING($, $, 'DRILLING HOLE3', #20, $, #262, #251, #230, $, #0, #5, $);
#78= BORING($, $, 'BORING HOLE2', #20, $, #266, #270, #230, #15, #15, #5, #0, #T, #1, #0.5);
#79= BORING($, $, 'BORING HOLE3', #20, $, #266, #270, #230, #15, #15, #5, #0, #T, #1, #0.5);
#80= COUNTER_SINKING($, $, 'COUNTER SINKING HOLE2',  #30, $, #271, #261, #230, #1, #25, #5, $);
#81= COUNTER_SINKING($, $, 'COUNTER SINKING HOLE3',  #30, $, #271, #261, #230, #1, #25, #5, $);
#82= DRILLING($, $, 'DRILLING HOLE4', #20, $, #275, #251, #230, $, #0, #5, $);
#83= MULTISTEP_DRILLING($, $, 'DRILLING HOLE5',
    #20, $, #247, #251, #230, $, #0, #5, #5, #10, #10, #0.5);
#84= TAPPING($, $, 'TAPPING HOLES', #10, $, #279, #283, #230, #15, #8.3, #5, $, #T, #0.5);
#85= COUNTER_SINKING($, $, 'COUNTER SINKING HOLE4',  #30, $, #284, #261, #230, #1, #10, #5, $);
#86= COUNTER_SINKING($, $, 'COUNTER SINKING HOLE5',  #30, $, #284, #261, #230, #1, #10, #5, $);
#87= DRILLING($, $, 'DRILLING HOLE6', #10, $, #288, #251, #230, #22, #0, #5, $);
#88= REAMING($,$,'REAMING HOLE6',10.,$,#292,#296,#230,$,21.,9.,$,$,$,.F.,$,$);
#89= COUNTER_SINKING($,$,'COUNTER SINKING HOLE6',30.,$,#284,#261,#230,$,1.,10.,$,$,$);
#90= BOTTOM_AND_SIDE_FINISH_MILLING($,$,'SIDE SLOT',10.,$,#302,#306,#230,$,#219,#220,#311,$,$,$);
#91= BOTTOM_AND_SIDEROUGH_MILLING($,$,'ROUGHING1 T-SLOT',10.,$,#307,#311,#230,$,$,$,11.,$,1.,0.5);
#92= BOTTOM_AND_SIDEROUGH_MILLING($,$,'ROUGHING2 T-SLOT',10.,$,#307,#311,#230,$,$,$,11.,$,1.,0.5);
#93= BOTTOM_AND_SIDE_FINISH_MILLING($,$,'FINISHING1 T-SLOT',10.,$,#312,#236,#230,$,$,$,$,$);
#94= BOTTOM_AND_SIDE_FINISH_MILLING($,$,'FINISHING2 T-SLOT',10.,$,#313,#317,#230,$,$,$,$,$);
#95= BOTTOM_AND_SIDEROUGH_MILLING($,$,'ROUGHING SLOT1',10.,$,#318,#322,#230,$,$,$,1.,0.5);
#96= DRILLING($,$,'DRILLING SLOT2',10.,$,#288,#251,#230,$,5.,0.,$,$,$);
#97= BOTTOM_AND_SIDEROUGH_MILLING($,$,'ROUGHING SLOT2',10.,$,#318,#322,#230,$,$,$,1.,0.5);
#98= BOTTOM_AND_SIDE_FINISH_MILLING($,$,'FINISHING SLOT1',10.,$,#323,#327,#230,$,$,$,196,$,197,5.,5.,$,$);
#99= BOTTOM_AND_SIDE_FINISH_MILLING($,$,'FINISH SLOT2',10.,$,#323,#327,#230,$,$,$,$,$);
#100= COMPOUND_FEATURE('COMPOUND FEATURE HOLE1',#7,(),#363,(#106,#107));
#101= COMPOUND_FEATURE('COMPOUND FEATURE HOLE2',#7,(),#363,(#108,#109,#110));
#102= COMPOUND_FEATURE('COMPOUND FEATURE HOLE3',#7,(),#370,(#111,#112,#110));
#103= GENERAL_PATTERN('PATTERN_HOLE4',#7,(),#374,#113,(#377,#376));
#104= GENERAL_PATTERN('PATTERN_HOLES',#7,(),#380,#116,(#383,#382));
#105= GENERAL_PATTERN('PATTERN_HOLES6',#7,(),#386,#120,(#389,#388));
#106= ROUND_HOLE('HOLE1 STRAIGHT',#7,(),#73,#74,#368,#1000,#539,$,#215);
#107= ROUND_HOLE('HOLE1 TAPERED',#7,(),#75,#368,#1010,#542,#212,#215);
#108= ROUND_HOLE('HOLE2 FLAT BOTTOM',#7,(),#78,#369,#1020,#546,$,#214);
#109= ROUND_HOLE('HOLE2 CONICAL BOTTOM',#7,(),#76,#369,#1030,#549,$,#213);
#110= ROUND_HOLE('HOLE2 TAPERED',#7,(),#80,#369,#1040,#552,#212,#214);
#111= ROUND_HOLE('HOLE3 FLAT BOTTOM',#7,(),#79,#373,#1050,#556,$,#214);
#112= ROUND_HOLE('HOLE2 THROUGH BOTTOM',#7,(),#77,#373,#1060,#559,$,#215);
#113= COMPOUND_FEATURE('COMPOUND FEATURE HOLE4',#7,(),#379,#114,#115);
#114= ROUND_HOLE('HOLE4 CONICAL BOTTOM',#7,(),#82,#379,#1070,#567,$,#213);
#115= ROUND_HOLE('HOLE4 TAPERED',#7,(),#85,#379,#1080,#570,#212,#214);
#116= COMPOUND_FEATURE('COMPOUND_FEATURE_HOLE5',#7,(),#385,#117,#118,#119);
#117= ROUND_HOLE('HOLE5 CONICAL BOTTOM',#7,(),#83,#385,#1090,#576,$,#213);
#118= ROUND_HOLE('TAPM10',#7,(),#84,#385,#1100,#579,$,#214);
#119= ROUND_HOLE('HOLE5 TAPERED',#7,(),#86,#385,#1110,#583,#212,#214);
#120= COMPOUND_FEATURE('COMPOUND_FEATURE_HOLE6',#7,(),#391,#121,#122);
#121= ROUND_HOLE('HOLE6 CONICAL BOTTOM',#7,(),#87,#88,#391,#1120,#589,$,#213);
#122= ROUND_HOLE('HOLE6 TAPERED',#7,(#89),#391,#1130,#592,#212,#214);
#123= SLOT('SIDE-SLOT1',#7,(#90),#395,#338,#127,#132,(#218));
#124= SLOT('TEE_SLOT1',#7,(#91,#92,#93,#94),#394,#339,#137,#143,(#216,#217));
#125= SLOT('SLOT1',#7,(#95,#68),#392,#340,#156,#162,(#216,#217));
#126= SLOT('SLOT2',#7,(#96,#97,#99),#393,#341,#167,#173,(#217,#217));
#127= COMPLETE_CIRCULAR_PATH(#128,#130);
#128= AXIS2_PLACEMENT_3D('SIDE1:COURSE OF TRAVEL',#129,$,$);
#129= CARTESIAN_POINT('SIDE1:PLACEMENT',(0.,0.,-20.));
#130= TOLERANCED_LENGTH_MEASURE(44.,#131);
#131= PLUS_MINUS_VALUE(0.1,0.1,3);
#132= SQUARE_U_PROFILE(#400,#133,#135,0.,#135,0.);
#133= TOLERANCED_LENGTH_MEASURE(3.5,#134);
#134= PLUS_MINUS_VALUE(0.1,0.1,3);
#135= TOLERANCED_LENGTH_MEASURE(0.1,#136);
#136= PLUS_MINUS_VALUE(0.02,0.02,3);
#137= GENERAL_PATH(#138,#140);
#138= AXIS2_PLACEMENT_3D('TEE1:COURSE OF TRAVEL',#139,$,$);
#139= CARTESIAN_POINT('TEE1:PLACEMENT',(0.,0.,-10.));
#140= POLYLINE('TEE_SLOT1',(#141,#142));
#141= CARTESIAN_POINT('TEE_SLOT1',(0.,0.,0.));
#142= CARTESIAN_POINT('TEE_SLOT1',(50.,0.,0.));
#143= TEE_PROFILE($,60.,60.,#144,#146,#148,#150,#152,#154);
#144= TOLERANCED_LENGTH_MEASURE(32.,#145);
#145= PLUS_MINUS_VALUE(0.1,0.1,3);
#146= TOLERANCED_LENGTH_MEASURE(20.,#147);
#147= PLUS_MINUS_VALUE(0.1,0.1,3);
#148= TOLERANCED_LENGTH_MEASURE(0.2,#149);
#149= PLUS_MINUS_VALUE(0.01,0.01,3);
#150= TOLERANCED_LENGTH_MEASURE(20.,#151);
#151= PLUS_MINUS_VALUE(0.1,0.1,3);
#152= TOLERANCED_LENGTH_MEASURE(0.1,#153);
#153= PLUS_MINUS_VALUE(0.02,0.02,3);
#154= TOLERANCED_LENGTH_MEASURE(0.1,#155);
#155= PLUS_MINUS_VALUE(0.02,0.02,3);
#156= GENERAL_PATH(#157,#159);
#157= AXIS2_PLACEMENT_3D('SLOT1:COURSE OF TRAVEL',#158,$,$);
#158= CARTESIAN_POINT('SLOT1:PLACEMENT',(0.,0.,0.));
#159= POLYLINE('SLOT1',(#160,#161));
#160= CARTESIAN_POINT('SLOT1',(0.,0.,0.));
#161= CARTESIAN_POINT('SLOT1',(-20.,0.,0.));
#162= SQUARE_U_PROFILE($,#163,#165,0.,#165,0.);
#163= TOLERANCED_LENGTH_MEASURE(10.,#164);
#164= PLUS_MINUS_VALUE(0.1,0.1,3);
#165= TOLERANCED_LENGTH_MEASURE(0.1,#166);
#166= PLUS_MINUS_VALUE(0.02,0.02,3);
#167= GENERAL_PATH(#168,#170);
#168= AXIS2_PLACEMENT_3D('SLOT2: COURSE OF TRAVEL',#169,$,$);
#169= CARTESIAN_POINT('SLOT2: PLACEMENT', (0.,0.,0.));
#170= POLYLINE('SLOT2',(#171,#172));
#171= CARTESIAN_POINT('SLOT2', (0.,0.,0.));
#172= CARTESIAN_POINT('SLOT2',(25.,0.,0.));
#173= SQUARE_U_PROFILE($,#174,#176,0.,#176,0.);
#174= TOLERANCED_LENGTH_MEASURE(10.,#134);
#175= PLUS_MINUS_VALUE(0.1,0.1,3);
#176= TOLERANCED_LENGTH_MEASURE(0.1,#136);
#177= PLUS_MINUS_VALUE(0.02,0.02,3);
#178= LINEAR_PATH($,#180,#181);
#179= LINEAR_PROFILE($,#183);
#180= TOLERANCED_LENGTH_MEASURE(180.,#182);
#181= DIRECTION('COURSE OF TRAVEL DIRECTION', (0.,1.,0.));
#182= PLUS_MINUS_VALUE(0.3,0.3,3);
#183= NUMERIC_PARAMETER('PROFILE LENGTH',260.,'MM');
#184= UNIDIRECTIONAL(0.2,.T.,#483.,CONVENTIONAL.);
#185= LINEAR_PATH($,#187,#188);
#186= LINEAR_PROFILE($,#190);
#187= TOLERANCED_LENGTH_MEASURE(-50.,#189);
#188= DIRECTION('COURSE OF TRAVEL DIRECTION', (0.,0.,-1.));
#189= PLUS_MINUS_VALUE(0.3,0.3,3);
#190= NUMERIC_PARAMETER('PROFILE LENGTH',230.,'MM');
#191= CONTOUR_PARALLEL($,.T.,.CW.,.CLIMB.);
#192= LINEAR_PATH($,#193,#194);
#193= TOLERANCED_LENGTH_MEASURE(100.,#195);
#194= DIRECTION('COURSE OF TRAVEL DIRECTION', (0.,1.,0.));
#195= PLUS_MINUS_VALUE(0.3,0.3,3);
#196= PLUNGE_TOOLAXIS($);
#197= PLUNGE_TOOLAXIS($);
#198= BIDIRECTIONAL(5.,.T.,$,$,$);
#199= BOSS('POCKET1-BOSS1',#7,(),#359,#337,#200,$);
#200= GENERAL_CLOSED_PROFILE($,#201);
#201= COMPOSITE_CURVE('BOSS BOUNDARY POCKET1',(#487,#497,#505,#513),.F.);
#202= PLANAR_POCKET_BOTTOM_CONDITION();
#203= GENERAL_PROFILE($,#204);
#204= COMPOSITE_CURVE('OPEN BOUNDARY POCKET1',(#521,#529),.F.);
#205= BIDIRECTIONAL_CONTOUR($,.T.,$,$,$,CONVENTIONAL.);
#206= BIDIRECTIONAL_CONTOUR($,.T.,$,$,$,CONVENTIONAL.);
#207= PLUNGE_RAMP($,45.);
#208= PLUNGE_RAMP($,45.);
#209= BIDIRECTIONAL(0.2,.T.,#210,$,$);
#210= DIRECTION('OPEN POCKET:FEED DIRECTION',(0.,1.,0.));
#211= CONTOUR_PARALLEL($,.T.,.CW.,.CLIMB.);
#212= ANGLE_TAPER(45.);
#213= CONICAL_HOLE_BOTTOM(30.,$);
#214= FLAT_HOLE_BOTTOM();
#215= THROUGH_BOTTOM_CONDITION();
#216= OPEN_SLOT_END_TYPE();
#217= RADIUSED_SLOT_END_TYPE();
#218= LOOP_SLOT_END_TYPE();
#219= PLUNGEHELIX($,25.,45.);
#220= PLUNGEHELIX($,25.,45.);
#221= CONTOUR_PARALLEL($,$,$,.CLIMB.);
#222= CENTER_MILLING(5.,$);
#223= CENTER_MILLING(5.,$);
#224= CENTER_MILLING(4.,$);
#225= CENTER_MILLING(4.,$);
#226= MILLING_CUTTING_TOOL('MILL 36MM','#228,(#227),75.,#482,32.);
#227= CUTTING_COMPONENT(20.,#328,$,1230.,.#229);
#228= FACEMILL(#329,8,.RIGHT.,.F.,$);
#229= MILLING_TECHNOLOGY(0.5,.TCP.,14.,S,0.5.,.T.,.F.,.F.,$);
#230= MILLING_MACHINE_FUNCTIONS(F,.$,$,$,F,$());
#231= MILLING_CUTTING_TOOL('ENDMILL 25',#233,(#232),72.,#482,20.);
#232= CUTTING_COMPONENT(30.,#330,$,1200.,.#236);
#233= ENDMILL(#331,4.,.RIGHT.,.F.,$);
#234= MILLING_TECHNOLOGY($,CCP.,5,5,40.,S,.T.,.F.,.F.,$);
#235= MILLING_MACHINE_FUNCTIONS(F,.$,$,$,F,$());
#236= MILLING_TECHNOLOGY($,TCP.,S,50.,0.2,.T.,.F.,.F.,$);
#237= MILLING_CUTTING_TOOL('ENDMILL 12',#239,(#238),54.,#482,8.);
#238= CUTTING_COMPONENT(30.,#332,$,1000.,.#241);
#239= ENDMILL(#240,3.,.RIGHT.,.F.,$);
#240= MILLING_TOOL_DIMENSION(12.,S,$,S,$,S,$,S,$);
#241= MILLING_TECHNOLOGY(0.024,..TCP.,S,120.,.T.,.F.,.F.,$);
#242= MILLING_CUTTING_TOOL('BORING TOOL 36','#244,(#243),200.,#482,32.);
#243= CUTTING_COMPONENT(30.,#332,$,1200.,.#246);
#244= BORING_TOOL(#245,3.,.RIGHT.,.T.,.S.,.F.);
#245= MILLING_TOOL_DIMENSION(36.,11.,0.,4.3,0.3,$,$);
#246= MILLING_TECHNOLOGY($,TCP.,2.3,$,0.1,.T.,.F.,.F.,$);
#247= MILLING_CUTTING_TOOL('DRILL 8','#249,(#248),70.,#482,8.);
#248= CUTTING_COMPONENT(30.,#332,$,1230.,.#251);
#249= TWIST_DRILL(#250,8.,.RIGHT.,$,$);
#250= MILLING_TOOL_DIMENSION(8.,31.,$,$,$,$,$);
#251 = MILLING_TECHNOLOGY(0.08, TCP, 1.6, S,S, T,F,F,S);
#252 = MILLING_CUTTING_TOOL('SPADE DRILL 40', #254, (#253), 160, #482, 32);
#253 = CUTTING_COMPONENT(30, #328, S, 1200, #256);
#254 = SPADE_DRILL(#255, 4, RIGHT, F,S);
#255 = MILLING_TOOL_DIMENSION(40, 20, 0, 13.5, 0.8, S,S);
#256 = MILLING_TECHNOLOGY(0.005, TCP, 2.3, S,S, T,F,F,S);
#257 = MILLING_CUTTING_TOOL('COUNTERSINK 40', #259, (#258), 70, #482, 20);
#258 = CUTTING_COMPONENT(30, #332, S, 1400, #261);
#259 = COUNTERSINK(#260, 2, RIGHT, F,S, 2.7);
#260 = MILLING_TOOL_DIMENSION(45, 45, S, 28, S,S,S);
#261 = MILLING_TECHNOLOGY(0.09, TCP, 1.2, S,S, T,F,F,S);
#262 = MILLING_CUTTING_TOOL('DRILL 15', #264, (#263), 165, S, 15);
#263 = CUTTING_COMPONENT(30, #332, S, 1350, #251);
#264 = TWIST_DRILL(#265, 2, LEFT, S,S);
#265 = MILLING_TOOL_DIMENSION(15, 31, S,S,S,S, S);
#266 = MILLING_CUTTING_TOOL('BORING TOOL 25', #268, (#267), 180, #482, 25);
#267 = CUTTING_COMPONENT(30, #328, S, 1000, #270);
#268 = BORING_TOOL(#269, 2, RIGHT, T,F, S,F);
#269 = MILLING_TOOL_DIMENSION(25, 7, S, 4.5, 0.35, S,S);
#270 = MILLING_TECHNOLOGY(0.005, TCP, 2.1, S,S, T,F,F,S);
#271 = MILLING_CUTTING_TOOL('COUNTERSINK 25', #273, (#272), 60, #482, 20);
#272 = CUTTING_COMPONENT(30, #332, S, 1400, #261);
#273 = COUNTERSINK(#274, 2, RIGHT, F,S, 2.27);
#274 = MILLING_TOOL_DIMENSION(30, 45, S, 18, S,S,S);
#275 = MILLING_CUTTING_TOOL('DRILL 10', #277, (#276), 90, #482, 10);
#276 = CUTTING_COMPONENT(30, #332, S, 1280, #251);
#277 = TWIST_DRILL(#278, 2, RIGHT, S,S);
#278 = MILLING_TOOL_DIMENSION(10, 31, S,S,S,S, S);
#279 = MILLING_CUTTING_TOOL('TAP M10', #281, (#280), 63, #482, 7);
#280 = CUTTING_COMPONENT(30, #332, S, 200, #283);
#281 = TAP(#282, S, RIGHT, S,S);
#282 = MILLING_TOOL_DIMENSION(10, S,S,S,S,S,S);
#283 = MILLING_TECHNOLOGY(0.035809, TCP, 0.9, S,S, T,F,F,S);
#284 = MILLING_CUTTING_TOOL('COUNTERSINK 10', #286, (#285), 50, #482, 10);
#285 = CUTTING_COMPONENT(30, #332, S, 1400, #261);
#286 = COUNTERSINK(#287, 2, RIGHT, F,S, 1.84);
#287 = MILLING_TOOL_DIMENSION(15, 45, S, 8, S,S,S);
#288 = MILLING_CUTTING_TOOL('DRILL 9', #290, (#289), 80, #482, 9);
#289 = CUTTING_COMPONENT(30, #332, S, 1230, #251);
#290 = TWIST_DRILL(#291, 2, RIGHT, S,S);
#291 = MILLING_TOOL_DIMENSION(9, 31, S,S,S,S,S);
#292 = MILLING_CUTTING_TOOL('REAMER 10', #294, (#293), 100, #482, 8);
#293 = CUTTING_COMPONENT(30, #328, S, 980, #296);
215

#294= REAMER(#295,6,.LEFT.,$,$);
#295= MILLING_TOOL_DIMENSION(10.,20.,0.3,70.,0.5,1.5,3.6);
#296= MILLING_TECHNOLOGY(0.08,.TCP.,1.2,S.,T.,F.,F.,$);
#297= MILLING_CUTTING_TOOL('T-SLOT MILL',#299(#298),62.,#482,15.);
#298= CUTTING_COMPONENT(30.,#328,$,1120.,#301);
#299= T_SLOT_MILL(#300,4,.RIGHT.,F.,S,3.5);
#300= MILLING_TOOL_DIMENSION(35.,S.,S.,S.,S.,S.,$);
#301= MILLING_TECHNOLOGY(0.0165,.TCP.,5.5,S.,T.,F.,F.,$);
#302= MILLING_CUTTING_TOOL('ENDMILL 16',#304(#303),54.,#482,10.);
#303= CUTTING_COMPONENT(30.,#328,$,1110.,#306);
#304= ENDMILL(#305,3,.RIGHT.,F.,$);
#305= MILLING_TOOL_DIMENSION(16.,S.,S.,S.,S.,S.,$);
#306= MILLING_TECHNOLOGY(0.024,.TCP.,S.,120.,S.,T.,F.,F.,$);
#307= MILLING_CUTTING_TOOL('T-SLOT MILL 25',#309(#308),63.,#482,10.);
#308= CUTTING_COMPONENT(30.,#328,$,1230.,#311);
#309= T_SLOT_MILL(#310,4,.RIGHT.,F.,S,18.);
#310= MILLING_TOOL_DIMENSION(25.,S.,S.,S.,S.,S.,$);
#311= MILLING_TECHNOLOGY(S.,TCP.,S.,70.,0.18.,T.,F.,F.,$);
#312= MILLING_CUTTING_TOOL('ENDMILL 18',#314(#313),64.,#482,15.);
#313= MILLING_CUTTING_TOOL('T-SLOT MILL 32MM',#315(#314),72.,#482,10.);
#314= CUTTING_COMPONENT(30.,S.,S,1120.,#317);
#315= T_SLOT_MILL(#316,4,.NEUTRAL.,F.,S,20.);
#316= MILLING_TOOL_DIMENSION(32.,S.,S.,S.,S.,S.,$);
#317= MILLING_TECHNOLOGY(0.03,.TCP.,S.,70.,S.,T.,F.,F.,$);
#318= MILLING_CUTTING_TOOL('ENDMILL 8',#320(#319),54.,#482,8.);
#319= CUTTING_COMPONENT(30.,S.,S,1000.,#322);
#320= ENDMILL(#321,3,.RIGHT.,F.,$);
#321= MILLING_TOOL_DIMENSION(8.,S.,S.,S.,S.,S.,$);
#322= MILLING_TECHNOLOGY(S.,TCP.,3.,S.,0.1.,T.,F.,F.,$);
#323= MILLING_CUTTING_TOOL('ENDMILL 10',#325(#324),54.,#482,8.);
#324= CUTTING_COMPONENT(30.,S.,S,1000.,#327);
#325= ENDMILL(#326,3,.RIGHT.,F.,$);
#326= MILLING_TOOL_DIMENSION(10.,S.,S.,S.,S.,S.,$);
#327= MILLING_TECHNOLOGY(S.,TCP.,S.,120.,S.,T.,F.,F.,$);
#328= MATERIAL('TIN','TIN',());
#329= MILLING_TOOL_DIMENSION(125.,S.,S.,S.,S.,S.,$);
#330= MATERIAL('TIN','TIN',());
#331= MILLING_TOOL_DIMENSION(18.,S.,S.,S.,S.,S.,$);
#332= MATERIAL('EM5CO5','HSS/CO',());
#333= ELEMENTARY_SURFACE('SECURITIY PLANE',#347);
#334= ELEMENTARY_SURFACE('PLANAR FACE1-DEPTH PLANE',#351);
#335= ELEMENTARY_SURFACE('SIDE1-DEPTH',#354);
#336= ELEMENTARY_SURFACE('STEP1-DEPTH',#358);
#337= ELEMENTARY_SURFACE('POCKET1-DEPTH',#362);
#338= ELEMENTARY_SURFACE('SLOT1:DEPTH',#396);
#339= ELEMENTARY_SURFACE('SLOT2:DEPTH',#397);
#340= ELEMENTARY_SURFACE('TEE1:DEPTH',#398);
#341= ELEMENTARY_SURFACE('SIDE1:DEPTH',#399);
#342= AXIS2_PLACEMENT_3D('SETUP1',#401,#402,#403);
#344= AXIS2_PLACEMENT_3D('PL_WORKPIECE_EXampLe2',#404,#405,#406);
#347= AXIS2_PLACEMENT_3D('PL_MAIN_SECPLANE',#407,#408,#409);
#348= AXIS2_PLACEMENT_3D('PL_HEAD1',#410,#411,#412);
#351= AXIS2_PLACEMENT_3D('PLANAR FACE1',#413,#414,#415);
#352= AXIS2_PLACEMENT_3D('SIDE1-PLACEMENT',#416,#417,#418);
#354= AXIS2_PLACEMENT_3D('SIDE1-DEPTH',#419,#420,#421);
#355= AXIS2_PLACEMENT_3D('STEP1-PLACEMENT',#422,#423,#424);
#358= AXIS2_PLACEMENT_3D('STEP1-DEPTH',#425,#426,#427);
#359= AXIS2_PLACEMENT_3D('CF_HOLE1',#434,#435,#436);
#362= AXIS2_PLACEMENT_3D('POCKET1-DEPTH',#428,#429,#430);
#363= AXIS2_PLACEMENT_3D('CF_HOLE3',#441,#442,#443);
#368= AXIS2_PLACEMENT_3D('HOLE1',#437,#438,#439);
#369= AXIS2_PLACEMENT_3D('HOLE2',#440,#445,#446);
#370= AXIS2_PLACEMENT_3D('HOLE3',#444,#442,#443);
#374= AXIS2_PLACEMENT_3D('HOLE4',#445,#446,#447);
#376= AXIS2_PLACEMENT_3D('HOLE5',#451,#452,#453);
#382= AXIS2_PLACEMENT_3D('HOLE6',#454,#455,#456);
#383= AXIS2_PLACEMENT_3D('HOLE5_NEAR_POCKET',#455,#452,#453);
#385= AXIS2_PLACEMENT_3D('HOLE5',#456,#452,#453);
#386= AXIS2_PLACEMENT_3D('HOLE6',#457,#458,#459);
#388= AXIS2_PLACEMENT_3D('HOLE6_NEAR_REGION',#460,#461,#462);
#389= AXIS2_PLACEMENT_3D('HOLE6_NEAR_POCKET',#461,#458,#459);
#391= AXIS2_PLACEMENT_3D('HOLE6',#462,#458,#459);
#392= AXIS2_PLACEMENT_3D('SLOT1',#463,#464,#465);
#393= AXIS2_PLACEMENT_3D('SLOT2',#466,#467,#468);
#394= AXIS2_PLACEMENT_3D('TEE SLOT',#469,#470,#471);
#395= AXIS2_PLACEMENT_3D('SIDE SLOT',#472,#473,#474);
#396= AXIS2_PLACEMENT_3D('SLOT1:DEPTH',#475,#476,#477);
#397= AXIS2_PLACEMENT_3D('SLOT2:DEPTH',#476,#478,#479);
#398= AXIS2_PLACEMENT_3D('TEE SLOT:DEPTH',#477,#478,#479);
#399= AXIS2_PLACEMENT_3D('SIDE SLOT:DEPTH',#478,#479,#480);
#400= AXIS2_PLACEMENT_3D('SIDE SLOT:SQUARE',#480,#481,#482);
#401= CARTESIAN_POINT('SETUP1 LOCATION',(123.,123.,45.));
#402 = DIRECTION(' AXI',S(0.,0.,1.));
#403 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#404 = CARTESIAN_POINT('WORKPIECE1:LOCATION',S(0.,0.,0.));
#405 = DIRECTION(' AXI',S(0.,0.,1.));
#406 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#407 = CARTESIAN_POINT('SECPLANE1: LOCATION',S(0.,0.,30.));
#408 = DIRECTION(' AXI',S(0.,0.,1.));
#409 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#410 = CARTESIAN_POINT('PLANAR FACE : LOCATION',S(0.,0.,5.));
#411 = DIRECTION(' AXI',S(0.,0.,1.));
#412 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#413 = CARTESIAN_POINT('PLANAR FACE1:DEPTH LOCATION',S(0.,0.,-5.));
#414 = DIRECTION(' AXI',S(0.,0.,1.));
#415 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#416 = CARTESIAN_POINT('SIDE1 PROFILE:LOCATION',S(0.,0.,0.));
#417 = DIRECTION(' AXI',S(0.,0.,1.));
#418 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#419 = CARTESIAN_POINT('SIDE1 PROFILE:DEPTH',S(0.,0.,-50.));
#420 = DIRECTION(' AXI',S(0.,0.,1.));
#421 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#422 = CARTESIAN_POINT('STEP1 PROFILE:LOCATION',S(230.,40.,0.));
#423 = DIRECTION(' AXI',S(0.,0.,1.));
#424 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#425 = CARTESIAN_POINT('STEP1 PROFILE:DEPTH',S(0.,0.,-20.));
#426 = DIRECTION(' AXI',S(0.,0.,1.));
#427 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#428 = CARTESIAN_POINT('POCKET1 PROFILE:LOCATION',S(50.,0.,0.));
#429 = DIRECTION(' AXI',S(0.,0.,1.));
#430 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#431 = CARTESIAN_POINT('POCKET1 PROFILE:DEPTH',S(0.,0.,-20.));
#432 = DIRECTION(' AXI',S(0.,0.,1.));
#433 = DIRECTION(' REF_DIRECTION',S(1.,0.,0.));
#434 = CARTESIAN_POINT('S(160.,150.,0.));
#435 = DIRECTION('S(0.,0.,1.));
#436 = DIRECTION('S(1.,0.,0.));
#437 = CARTESIAN_POINT('S(0.,0.,0.));
#438 = DIRECTION('S(0.,0.,1.));
#439 = DIRECTION('S(1.,0.,0.));
#440 = CARTESIAN_POINT('S(0.,0.,0.));
#441 = CARTESIAN_POINT('S(160.,30.,0.));
#442 = DIRECTION('S(0.,0.,1.));
#443 = DIRECTION('S(1.,0.,0.));
#444 = CARTESIAN_POINT('S(0.,0.,0.));
#445 = CARTESIAN_POINT('',(130.,120.,0.));
#446 = DIRECTION("",(0.,0.,1.));
#447 = DIRECTION("",(1.,0.,0.));
#448 = CARTESIAN_POINT("",(0.,0.,0.));
#449 = CARTESIAN_POINT("",(0.,-60.,0.));
#450 = CARTESIAN_POINT("",(0.,0.,0.));
#451 = CARTESIAN_POINT("",(190.,120.,0.));
#452 = DIRECTION("",(0.,0.,1.));
#453 = DIRECTION("",(1.,0.,0.));
#454 = CARTESIAN_POINT("",(0.,0.,0.));
#455 = CARTESIAN_POINT("",(0.,-60.,0.));
#456 = CARTESIAN_POINT("",(0.,0.,0.));
#457 = CARTESIAN_POINT("",(245.,110.,0.));
#458 = DIRECTION("",(0.,0.,1.));
#459 = DIRECTION("",(1.,0.,0.));
#460 = CARTESIAN_POINT("",(0.,0.,0.));
#461 = CARTESIAN_POINT("",(0.,-40.,0.));
#462 = CARTESIAN_POINT("",(0.,0.,0.));
#463 = CARTESIAN_POINT("",(230.,90.,0.));
#464 = DIRECTION("",(0.,0.,1.));
#465 = DIRECTION("",(1.,0.,0.));
#466 = CARTESIAN_POINT("",(85.,90.,0.));
#467 = DIRECTION("",(0.,0.,1.));
#468 = DIRECTION("",(1.,0.,0.));
#469 = CARTESIAN_POINT("",(0.,90.,0.));
#470 = DIRECTION("",(0.,0.,1.));
#471 = DIRECTION("",(1.,0.,0.));
#472 = CARTESIAN_POINT("",(160.,90.,0.));
#473 = DIRECTION("",(0.,0.,1.));
#474 = DIRECTION("",(1.,0.,0.));
#475 = CARTESIAN_POINT("SIDE1:DEPTH",(0.,0.,-10.));
#476 = CARTESIAN_POINT("SIDE1:DEPTH",(0.,0.,-10.));
#477 = CARTESIAN_POINT("SIDE1:DEPTH",(0.,0.,-10.));
#478 = CARTESIAN_POINT("SIDE1:DEPTH",(0.,0.,-10.));
#479 = CARTESIAN_POINT("SIDE1:DEPTH",(-22.,0.,-20.));
#480 = DIRECTION("",(1.,0.,0.));
#481 = DIRECTION("",(0.,0.,-1.));
#482 = DIRECTION("X",(1.,0.,0.));
#483 = DIRECTION("",(1.,0.,0.));
#484 = POLYLINE("TEST1",(#485,#486));
#485 = CARTESIAN_POINT("",(25.,15.,0.));
#486 = CARTESIAN_POINT("",(45.,15.,0.));
#487 = COMPOSITE_CURVE_SEGMENT(CONT_SAME_GRADIENT,"T",#488);
#488 =
TRIMMED_CURVE('TEST2', #489, (PARAMETER_VALUE(270.)), (PARAMETER_VALUE(0.)), .T., .PARAMETER.);
#489 = CIRCLE('', #490, 5.);
#490 = AXIS2_PLACEMENT_3D('', #491, #492, #493);
#491 = CARTESIAN_POINT('', (45., 20., 0.));
#492 = DIRECTION('', (0., 0., 1.));
#493 = DIRECTION('', (1., 0., 0.));
#494 = POLYLINE('', (#495, #496));
#495 = CARTESIAN_POINT('', (50., 20., 0.));
#496 = CARTESIAN_POINT('', (50., 30., 0.));
#497 = COMPOSITE_CURVE_SEGMENT(.CONT_SAME_GRADIENT., .T., #498);
#498 =
TRIMMED_CURVE('', #499, (PARAMETER_VALUE(0.)), (PARAMETER_VALUE(90.)), .T., .PARAMETER.);
#499 = CIRCLE('', #500, 5.);
#500 = AXIS2_PLACEMENT_3D('', #501, #492, #493);
#501 = CARTESIAN_POINT('', (45., 30., 0.));
#502 = POLYLINE('', (#503, #504));
#503 = CARTESIAN_POINT('', (45., 35., 0.));
#504 = CARTESIAN_POINT('', (25., 35., 0.));
#505 = COMPOSITE_CURVE_SEGMENT(.CONT_SAME_GRADIENT., .T., #506);
#506 =
TRIMMED_CURVE('', #507, (PARAMETER_VALUE(90.)), (PARAMETER_VALUE(180.)), .T., .PARAMETER.);
#507 = CIRCLE('', #508, 5.);
#508 = AXIS2_PLACEMENT_3D('', #509, #492, #493);
#509 = CARTESIAN_POINT('', (25., 30., 0.));
#510 = POLYLINE('', (#511, #512));
#511 = CARTESIAN_POINT('', (20., 30., 0.));
#512 = CARTESIAN_POINT('', (20., 20., 0.));
#513 = COMPOSITE_CURVE_SEGMENT(.CONT_SAME_GRADIENT., .T., #514);
#514 =
TRIMMED_CURVE('', #515, (PARAMETER_VALUE(180.)), (PARAMETER_VALUE(270.)), .T., .PARAMETER.);
#515 = CIRCLE('', #516, 5.);
#516 = AXIS2_PLACEMENT_3D('', #517, #492, #493);
#517 = CARTESIAN_POINT('', (25., 20., 0.));
#518 = POLYLINE('', (#519, #520));
#519 = CARTESIAN_POINT('', (0., 0., 0.));
#520 = CARTESIAN_POINT('', (0., 40., 0.));
#521 = COMPOSITE_CURVE_SEGMENT(.CONT_SAME_GRADIENT., .T., #522);
#522=
TRIMMED_CURVE("",#523,(PARAMETER_VALUE(180.)),(PARAMETER_VALUE(90.)),.F.,.PARAMETER.);
#523= CIRCLE("",#524,10.);
#524= AXIS2_placement_3D("",#525,#492,#493);
#525= CARTESIAN_POINT("",(#10.,40.,0.));
#526= POLYLINE("",(#527,#528));
#527= CARTESIAN_POINT("",(#10.,50.,0.));
#528= CARTESIAN_POINT("",(#60.,50.,0.));
#529= COMPOSITE_CURVE_SEGMENT(.CONT_SAME_GRADIENT.,.T.,#530);
#530=
TRIMMED_CURVE("",#531,(PARAMETER_VALUE(90.)),(PARAMETER_VALUE(0.)),.F.,.PARAMETER.);
#531= CIRCLE("",#532,10.);
#532= AXIS2_placement_3D("",#533,#492,#493);
#533= CARTESIAN_POINT("",(#60.,40.,0.));
#534= POLYLINE("",(#535,#536));
#535= CARTESIAN_POINT("",(#70.,40.,0.));
#536= CARTESIAN_POINT("",(#70.,0.,0.));
#537= TOLERANCED_LENGTH_MEASURE(40.,#780);
#538= TOLERANCED_LENGTH_MEASURE(40.,#780);
#539= TOLERANCED_LENGTH_MEASURE(40.,#781);
#541= TOLERANCED_LENGTH_MEASURE(0.5,#781);
#542= TOLERANCED_LENGTH_MEASURE(41.,#782);
#544= TOLERANCED_LENGTH_MEASURE(30.,#782);
#545= TOLERANCED_LENGTH_MEASURE(15.,#782);
#546= TOLERANCED_LENGTH_MEASURE(22.,#781);
#548= TOLERANCED_LENGTH_MEASURE(30.,#781);
#549= TOLERANCED_LENGTH_MEASURE(22.,#781);
#551= TOLERANCED_LENGTH_MEASURE(0.5,#781);
#552= TOLERANCED_LENGTH_MEASURE(23.,#781);
#554= TOLERANCED_LENGTH_MEASURE(40.,#781);
#555= TOLERANCED_LENGTH_MEASURE(15.,#781);
#556= TOLERANCED_LENGTH_MEASURE(22.,#781);
#558= TOLERANCED_LENGTH_MEASURE(40.,#781);
#559= TOLERANCED_LENGTH_MEASURE(22.,#781);
#563= TOLERANCED_LENGTH_MEASURE(0.5,#781);
#564= TOLERANCED_LENGTH_MEASURE(23.,#781);
#565= TOLERANCED_LENGTH_MEASURE(15.,#781);
#566= TOLERANCED_LENGTH_MEASURE(15.,#781);
#567= TOLERANCED_LENGTH_MEASURE(10.,#781);
#569= TOLERANCED_LENGTH_MEASURE(0.5,#781);
#570= TOLERANCED_LENGTH_MEASURE(11.,#781);
#574= TOLERANCED_LENGTH_MEASURE(25.,#781);
#575= TOLERANCED_LENGTH_MEASURE(25.,#781);
#576= TOLERANCED_LENGTH_MEASURE(9.,#781);
#578= TOLERANCED_LENGTH_MEASURE(15.5,#781);
#579= TOLERANCED_LENGTH_MEASURE(8.3,#781);
#581= TOLERANCED_LENGTH_MEASURE(1.75,#781);
#582= TOLERANCED_LENGTH_MEASURE(0.5,#781);
#583= TOLERANCED_LENGTH_MEASURE(11.,#783);
#584= TOLERANCED_LENGTH_MEASURE(20.,#781);
#585= TOLERANCED_LENGTH_MEASURE(20.,#781);
#586= TOLERANCED_LENGTH_MEASURE(10.,#783);
#587= TOLERANCED_LENGTH_MEASURE(0.5,#783);
#588= TOLERANCED_LENGTH_MEASURE(11.,#783);
#780= PLUS_MINUS_VALUE(0.5,0.,3);
#781= PLUS_MINUS_VALUE(0.025,0.,3);
#782= PLUS_MINUS_VALUE(0.1,0.1,3);
#783= PLUS_MINUS_VALUE(0.015,0.,3);

#1000= ELEMENTARY_SURFACE('4 HOLE1 STRAIGHT',#1001);
#1001= AXIS2_PLACEMENT_3D('HOLE1',#1002,#1003,#1004);
#1002= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-40.000));
#1003= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1004= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1010= ELEMENTARY_SURFACE('4 HOLE1 TAPERED ',#1011);
#1011= AXIS2_PLACEMENT_3D('HOLE1',#1012,#1013,#1014);
#1012= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-40.000));
#1013= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1014= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1020= ELEMENTARY_SURFACE('4 HOLE2 FLAT BOTTOM ',#1021);
#1021= AXIS2_PLACEMENT_3D('HOLE1',#1022,#1023,#1024);
#1022= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-15.000));
#1023= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1024= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1030= ELEMENTARY_SURFACE('4 HOLE2 CONICAL BOTTOM ',#1031);
#1031= AXIS2_PLACEMENT_3D('HOLE1',#1032,#1033,#1034);
#1032= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-30.000));
#1033= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1034= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1040= ELEMENTARY_SURFACE('4 HOLE2 TAPERED ',#1041);
#1041= AXIS2_PLACEMENT_3D('HOLE1',#1042,#1043,#1044);
#1042= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-0.5000));
#1043= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1044= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1050= ELEMENTARY_SURFACE('4 HOLE3 FLAT BOTTOM ',#1051);
#1051= AXIS2_PLACEMENT_3D('HOLE1',#1052,#1053,#1054);
#1052= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-15.000));
#1053= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1054= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1060= ELEMENTARY_SURFACE('4 HOLE2 THROUGH BOTTOM ',#1061);
#1061= AXIS2_PLACEMENT_3D('HOLE1',#1062,#1063,#1064);
#1062= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-40.000));
#1063= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1064= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1070= ELEMENTARY_SURFACE('4 HOLE4 CONICAL BOTTOM ',#1071);
#1071= AXIS2_PLACEMENT_3D('HOLE1',#1072,#1073,#1074);
#1072= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-15.000));
#1073= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1074= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1080= ELEMENTARY_SURFACE('4 HOLE5 TAPERED ',#1081);
#1081= AXIS2_PLACEMENT_3D('HOLE1',#1082,#1083,#1084);
#1082= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-0.5000));
#1083= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1084= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1090= ELEMENTARY_SURFACE('4 HOLE6 CONICAL BOTTOM ',#1091);
#1091= AXIS2_PLACEMENT_3D('HOLE1',#1092,#1093,#1094);
#1092= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-25.000));
#1093= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1094= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1100= ELEMENTARY_SURFACE('4 TAPM10 ',#1101);
#1101= AXIS2_PLACEMENT_3D('HOLE1',#1102,#1103,#1104);
#1102= CARTESIAN_POINT('HOLE1: DEPTH ',(0.000,0.000,-15.5000));
#1103= DIRECTION(' AXIS ',(0.000,0.000,1.000));
#1104= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1110= ELEMENTARY_SURFACE('4 HOLE5 TAPERED ',#1111);
#1111= AXIS2_PLACEMENT_3D('HOLE1',#1112,#1113,#1114);
#1112= CARTESIAN_POINT('HOLE1: DEPTH', (0.000,0.000,-0.5000));
#1113= DIRECTION(' AXIS',(0.000,0.000,1.000));
#1114= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1120= ELEMENTARY_SURFACE('4 HOLE6 CONICAL BOTTOM ',#1121);
#1121= AXIS2_PLACEMENT_3D('HOLE1',#1122,#1123,#1124);
#1122= CARTESIAN_POINT('HOLE1: DEPTH', (0.000,0.000,-20.000));
#1123= DIRECTION(' AXIS',(0.000,0.000,1.000));
#1124= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

#1130= ELEMENTARY_SURFACE('4 HOLE6 TAPERED ',#1131);
#1131= AXIS2_PLACEMENT_3D('HOLE1',#1132,#1133,#1134);
#1132= CARTESIAN_POINT('HOLE1: DEPTH', (0.000,0.000,-0.5000));
#1133= DIRECTION(' AXIS',(0.000,0.000,1.000));
#1134= DIRECTION(' REF_DIRECTION',(1.000,0.000,0.000));

ENDSEC;

END-ISO-10303-21;
C. Output of Data Packet VFB

ISO-10303-21;

HEADER;
FILE_DESCRIPTION(('Data Packet Example: Hole1 Straight'),'2;1');
FILE_NAME('DPacketV1.stp',2012-12-06T17:39:51+13:00,'Xi Vincent Wang',"'),'ST-Developer for Java 1.0',"');
FILE_SCHEMA(('COMBINED_SCHEMA'));
ENDSEC;

DATA;
#1=ROUND_HOLE('HOLE1 STRAIGHT ',#2,(#8,#17),#24,#28,#33,#35);
#2=WORKPIECE('PART 2',#3,0.01,$,$,$,#4,#5,#6,#7);
#3=MATERIAL('FC200','CAST IRON ',());
#4=CARTESIAN_POINT('CLAMPING_POSITION1',(25.0,25.0,-20.0));
#5=CARTESIAN_POINT('CLAMPING_POSITION2',(205.0,25.0,-20.0));
#6=CARTESIAN_POINT('CLAMPING_POSITION3',(25.0,155.0,-20.0));
#7=CARTESIAN_POINT('CLAMPING_POSITION4',(205.0,155.0,-20.0));
#8=MULTISTEP_DRILLING($,$,'BORING1 HOLE1',20.0,$,#9,#14,#16,$,48.0,0.0,$,#35,5.0,15.0,10.0,1.0);
#9=MILLING_CUTTING_TOOL('BORING TOOL 36',#10,#12,200.0,#15,32.0);
#10=BORING_TOOL(#11,3.,RIGHT,.,T.,S.,F.);
#11=MILLING_TOOL_DIMENSION(36.0,11.0,0.0,4.3,0.3,,$,S,);  
#12=CUTTING_COMPONENT(30.0,#13,$,1200.0,#14);
#13=MATERIAL('EMOS5CO5','HSS/CO',());
#14=MILLING_TECHNOLOGY($,.TCP.,2.3,$,0.1,.T.,F.,F.,$);
#15=DIRECTION('X',(1.0,0.0,0.0,));
#16=MILLING_MACHINE_FUNCTIONS(.F.,.,F.,S.,F.,T.,S.,S.,$());
#17=BORING($,$,'BORING2 HOLE1',20.0,$,#18,#23,#16,$,45.0,36.0,$,$,$,1.0,$);
#18=MILLING_CUTTING_TOOL('SPADE DRILL 40',#19,#21,160.0,#15,32.0);
#19=SPADE_DRILL(#20,4.,RIGHT,.,F.,F.,$);
#20=MILLING_TOOL_DIMENSION(40.0,20.0,0.0,13.5,0.8,$,S,);  
#21=CUTTING_COMPONENT(30.0,#22,$,1200.0,#23);
#22=MATERIAL('TIN','TIN',());
#23=MILLING_TECHNOLOGY(0.0050,.TCP.,2.3,$,T.,F.,F.,$);
#24=AXIS2_PLACEMENT_3D('HOLE1',#25,#26,#27);
#25=CARTESIAN_POINT(',(0.0,0.0,0.0,0.0,);  
#26=DIRECTION(',(0.0,0.0,0.1,0,));
#27=DIRECTION(',(1.0,0.0,0.0,));
#28=ELEMENTARY_SURFACE('4 HOLE1 STRAIGHT',#29);
#29=AXIS2_PLACEMENT_3D('HOLE1',#30,#31,#32);
#30=CARTESIAN_POINT('HOLE1: DEPTH',(0.0,0.0,-40.0));
#31=DIRECTION(' AXIS',(0.0,0.0,1.0));
#32=DIRECTION(' REF_DIRECTION',(1.0,0.0,0.0));
#33=TOLERANCED_LENGTH_MEASURE(40.0,#34);
#34=PLUS_MINUS_VALUE(0.025,0.0,3);
#35=THROUGH_BOTTOM_CONDITION();
ENDSEC;
END-ISO-10303-21;
Appendix II

Research Publication Abstracts
A. Journal Papers


_DIMP: an interoperable solution for software integration and product data exchange_

_Xi Vincent Wang and Xun William Xu*

*Department of Mechanical Engineering, University of Auckland, 20 Symonds Street, Auckland, New Zealand*

_(Received 2 December 2010; final version received 8 May 2011)_

Today, globalisation has become one of the main trends of manufacturing business that has led to a world-wide decentralisation of resources amongst not only individual departments within one company but also business partners. However, despite the development and improvement in the last few decades, difficulties in information exchange and sharing still exist in heterogeneous applications environments. This article is divided into two parts. In the first part, related research work and integrating solutions are reviewed and discussed. The second part introduces a collaborative environment called distributed interoperable manufacturing platform, which is based on a module-based, service-oriented architecture (SOA). In the platform, the STEP-NC data model is used to facilitate data-exchange among heterogeneous CAD/CAM/CNC systems.

Keywords: interoperability; STEP/STEP-NC; software integration; data exchanging; collaborative manufacturing

1. Introduction

In the modern manufacturing business, collaborations do not only exist amongst departments within the same enterprise, but also among business partners and contractors. Thus, the virtual enterprise concept was introduced to describe the consortium of different departments and companies which come together to quickly exploit fast-changing, worldwide product manufacturing opportunities (Zhang et al. 2000). The connection between these partners is required to be highly collaborated, distributed and in an agile regime. In such an environment, a tight data flow is a must between different organisations and processes. A better way of sharing and exchanging product data is also essential.

A Collaborative Product Data Exchange Environment based on STEP

(Drafted for IJCM)

Xi Vincent Wang, Xun W Xu*

Department of Mechanical Engineering
School of Engineering, University of Auckland, Auckland 1142, New Zealand

*xun.xu@auckland.ac.nz

ABSTRACT:

In the modern manufacturing context, CAD/CAM and CNC solutions are normally provided by different vendors, which gives rise to a heterogeneous application environment. Despite many integration approaches developed in the last decades, software integration and product data exchanging are still challenging issues that need to be addressed. In this paper, the authors proposed a collaborative product data exchanging mechanism for a Distributed Interoperable Manufacturing Platform (DIMP). In this platform, STEP (ISO 10303) and STEP-NC (ISO 14649) data models are utilized to support the data flow. A novel data exchanging mechanism is developed to provide the right amount and right level of product data subset to the users based on these models. This mechanism enables the users to work with a reasonable scope of product data, without interfering others’. To realize this concept, data extracting algorithms are developed to provide a customized data domain, and meta-data model compliant with STEP is proposed to guarantee the data tractability. Moreover, synchronization is catered for after the dataset is processed.

KEYWORDS

STEP, STEP-NC, interoperable, data exchange, product data sharing

1. Introduction

During the past few decades, manufacturing business has made big strides with the help of CAX software and CNC tools. Product design starts from CAD (Computer Aided-Design) and is analysed by use of CAE (Computer Aided-Engineering) software. CAPP (Computer Aided-Process Planning) software helps the users to plan manufacturing processes. The output is then sent to a CNC (Computer Numerical Controlled) machine tools for manufacturing the product.

**An interoperable solution for Cloud manufacturing**

Xi Vincent Wang, Xun W. Xu*

Department of Mechanical Engineering Faculty of Engineering, University of Auckland, Auckland 1142, New Zealand

**ABSTRACT**

Cloud manufacturing is a new concept extending and adopting the concept of Cloud computing for manufacturing. The aim is to transform manufacturing businesses to a new paradigm in that manufacturing capabilities and resources are compositional, integrated and optimized globally. This study presents an interoperable manufacturing perspective based on Cloud manufacturing. A literature search has been undertaken regarding Cloud architecture and technologies that can assist Cloud manufacturing. Manufacturing resources and capabilities are discussed in terms of Cloud service. A service-oriented, interoperable Cloud manufacturing system is proposed. Service methodologies are developed to support two types of Cloud users, i.e., customer users and enterprise users, along with standardized data models describing Cloud service and relevant features. Two case studies are undertaken to evaluate the proposed system. Cloud technology brings into manufacturing industry with a number of benefits such as openness, cost-efficiency, resource sharing and production scalability.

1. **Introduction**

Cloud computing (CComputing) is a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interactions [1,2]. It provides resources to a user on the "pay-as-you-go" basis. There are three common types of CComputing structure, i.e., Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS provides a bunch of physical and virtual machines, based on which users are able to install and deploy their own operation systems and execution environment, database, and web server. A PaaS client is able to develop and run its applications at the software layer. Finally, SaaS simplifies the utilization of a large amount of software applications remotely, elasticity and scalability.

During the past few years, many successful CComputing business cases are found world-wide [3–7]. Among various types of models, the key characteristic is that of pay-as-you-go. In the increasingly globalized manufacturing context, customer-oriented manufacturing is a promising approach to improving the service quality and competitiveness, in particular for the small and medium-sized enterprises (SMEs). Thus, a new concept of advanced manufacturing model is proposed worldwide, namely Cloud manufacturing (Manufacturing). In

Manufacturing Interoperability in the Cloud context
(Drafted for IJPR)

In a modern manufacturing business, collaborations not only exist amongst its own departments, but also among business partners. Cloud Manufacturing can assist this type of collaborations. As a new model of manufacturing network, Cloud Manufacturing combines Cloud Computing with manufacturing under service-oriented architecture. It is set to fundamentally change how products are designed, manufactured, shipped and maintained. Manufacturing interoperability can be achieved in two ways, process-centric interoperability based on Cloud and data-centric interoperability based on neutral data models. Cloud Computing is discussed in this paper from the manufacturing perspective. Interoperable and advanced manufacturing technologies are analysed to support the Cloud concept. Finally, Cloud Manufacturing is discussed with parallel manufacturing systems, and the research bottlenecks are identified. It is believed that Cloud Manufacturing can provide a strong support to the manufacturing industry, in particular for collaborative and interoperable manufacturing.

Keywords: Cloud Manufacturing, Cloud, Service-Oriented Architecture, Interoperable Manufacturing, Interoperability

1. Introduction

Over the years, manufacturing business has become increasingly challenging and competitive. Globalization makes manufacturing business no longer a game “in-house”. Resources, materials, knowledge and expertise are often outsourced and shared among business participants world-wide. It is almost impossible to carry out production tasks without the support of suppliers, contractors, retailers and many other type of business partners. Collaborations exist in every operational stage of a supply chain. Interactions
B. Book Chapter


Chapter 1
ICMS: A Cloud-Based Manufacturing System

Xi Vincent Wang and Xun W. Xu

Abstract Nowadays, Cloud Computing technology is providing a new way to do business by offering a scalable, flexible service over the Internet. It creates new solutions and opportunities to the modern enterprises, including the manufacturing industry. In this chapter, the essential features of Cloud Computing are discussed followed by a Cloud Manufacturing concept. In the second part, a service-oriented system called Interoperable Cloud-based Manufacturing System (ICMS) is proposed. ICMS provides a Cloud-based environment integrating existing and future manufacturing resources by packaging them using the Virtual Function Block mechanism and standardized description.

1.1 Introduction

Cloud Computing refers to the delivery of computing as a service, instead of a traditional product. According to the definition of National Institute of Standards and Technology (NIST) [1, 2], Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. In a Cloud Computing environment, decentralized
C. Conference Papers


Distributed Interoperable Manufacturing Platform

Based on STEP-NC

Xi Vincent Wang, Xun Xu, and Enrico Hämmerle
Department of Mechanical Engineering
University of Auckland
Auckland, 1010, New Zealand

ABSTRACT:

Today, CNC technology is a major contributor to the production capacity of industrial companies. Despite the development and improvement in the last decades, full integration of CAD/CAM/CNC system is still not a reality. This paper discusses the application of STEP-NC, a high-level data model, to integrate the existing software solutions found in manufacturing environment. Such a collaborative environment will be achieved based on the platform called Universal Manufacturing Platform (DIMP). The paper is divided into two parts, related research work reviewing and framework of DIMP. The first part of the paper is devoted to review the recent relevant publications. The discussion section that follows introduces the module-based, Service-Oriented Architecture (SOA) of DIMP. In the platform, STEP-NC data model is used to facilitate data-exchange among heterogeneous CAD/CAM/CNC systems.

1. INTRODUCTION

In today’s industry, the management of the manufacturing and business processes is based on various individual systems that form a “CAD-CAM-CNC” chain. Product information is started and created CAD (Computer-Aided Design) systems, while the CAM (Computer-Aided Manufacturing) systems enable the users to add manufacturing information to the design such as manufacturing process, tool path and fixture. The output of a CAM system is usually an NC part program. Such program is read by a CNC (Computer Numerical Control) system as input and transformed to the motion signals. Besides the systems mentioned above, there are other software products such as Computer Aided Engineering (CAE) and Computer Aided Process Planning (CAPP) tools. Furthermore, Product Data Management (PDM) and Product Lifecycle Management (PLM) systems are utilized to deal with the product data from design and manufacture, to service and disposal. All of these systems are developed by using different software tools, data formats, interfaces and databases, thus forming a highly heterogeneous data environment. Based on a survey among 251 executive officers of German-speaking enterprises[1], the main interoperability difficulties of systems in the CAD-CAM-CNC chain. It is evident that more than 75% of the large problems are directly related to the cause of different CAD versions or systems, different file format and conversion.

Data translation and conversion are common ways to cope with data heterogeneity in the CAD-CAM-CNC chain. These exercises often cause huge data loss. The rest of this paper reviews the existing research work in achieving interoperability and portability. A framework of Distributed Interoperable Manufacturing Platform (DIMP) is presented toward the end.

DEVELOPMENT OF A STEP-BASED COLLABORATIVE PRODUCT DATA EXCHANGE ENVIRONMENT

Xi Vincent Wang
School of Engineering, University of Auckland, Auckland 1142, New Zealand
xwan262@aucklanduni.ac.nz

Xun W Xu
School of Engineering, University of Auckland, Auckland 1142, New Zealand
xun.xu@auckland.ac.nz

ABSTRACT:
In a modern manufacturing enterprise, CAD/CAM/CNC solutions are normally provided by various vendors. This forms a heterogeneous application environment. Despite the many integration approaches developed in the last decades, software integration and product data exchanging are still challenging issues that need to be addressed. In this paper, the authors proposed a collaborative product data exchanging mechanism based on a Distributed Interoperable Manufacturing Platform (DIMP). In this platform, STEP (ISO 10303) and STEP-NC (ISO 14649) data formats are utilized to support the data flow. A novel data exchanging mechanism is developed to provide the right amount and level of product data subset to the users.

KEYWORDS
STEP, STEP-NC, interoperable, data exchange, product data sharing

1. INTRODUCTION
During the past few decades, manufacturing business has been developed remarkably with the help of CAX software and CNC tools. The product design starts from CAD (Computer Aided-Design) application and CAPP (Computer Aided-Process Planning) software helps the users to work on the issues are still unsolved. Due to heterogeneous enterprise environments in which business partners find themselves, multiple data formats, interfaces and databases are defined and used, thus forming a highly heterogeneous data environment. According to the calculation done by Parasolid's business development manager (Anonymous, 2000), approximately 20% of the product models imported from different software kernels still contain errors

**Virtual Function Block Mechanism in the Cloud Manufacturing Environment**

Xi Vincent Wang$^{1,a}$ and Xun W Xu$^{1,b,*}$

$^1$20 Symonds Street, Department of Mechanical Engineering, Faculty of Engineering, the University of Auckland, Auckland 0100, New Zealand

*$\text{xwan262}@\text{aucklanduni.ac.nz}$, $\text{xun.xu}@\text{auckland.ac.nz}$

**Keywords:** Cloud Manufacturing, Virtual Function Block, Interoperability, Interoperable Manufacturing

**Abstract:** Cloud Computing (CComputing) is a new manufacturing model that has evolved from Service-Oriented Architecture, networked manufacturing and CComputing. It provides intelligent, interoperable and distributed manufacturing models for the industry. This paper introduces a resource integration mechanism in the Cloud Manufacturing environment. Function Block technology is discussed from the Cloud Manufacturing perspective in detail. Next, a novel integration mechanism is proposed, namely the Virtual Function Block. Based on physical Function Blocks and software agents, Virtual Function Blocks are able to manipulate and integrate manufacturing resources via event states and data flows. During implementation, Creo Parametric was integrated as a Cloud Service with the help of VTBs to evaluate the mechanism.

**Introduction**

In recent years, Cloud Computing (CComputing) has been developed based on web-based technologies. CComputing is the usage of computing resources (e.g. hardware and software) as a service over a network. It provides new models and opportunities to integrate business processes and resources, including manufacturing. Among various types of models, the key characteristic of CComputing is not to pay the whole cost of a hardware/software resource, instead of paying for the amount of service provided (pay-as-you-go). The barriers of entry for costly computing and storage devices become much lower. The true spirit of the Cloud concept can be summarised as the capability of providing globally distributed, fast-responding, on-demand and quantifiable services. Cloud technology provides opportunities to improve the interoperability in the manufacturing environment, thus forming a new manufacturing model: Cloud Manufacturing.

**Cloud Manufacturing: a Novel Manufacturing Model**

According to Xu’s definition, Cloud Manufacturing can be defined as a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable manufacturing resources (e.g. manufacturing software tools, manufacturing equipment and manufacturing capabilities), which can be rapidly provisioned and released with minimal
IV. X. V. Wang and X. Xu, "Virtualize Manufacturing Capabilities in the Cloud: Requirements and Architecture (Accepted)," the 8th ASME 2013 Manufacturing Science and Engineering Conference, Madison, Wisconsin, USA, 2013.

DRAFT: VIRTUALIZE MANUFACTURING CAPABILITIES IN THE CLOUD: REQUIREMENTS AND ARCHITECTURE

Xi Vincent Wang
Department of Mechanical Engineering
University of Auckland
20 Symonds street
Auckland, New Zealand
xi.wang62@aucklanduni.ac.nz

Xun W Xu
Department of Mechanical Engineering
University of Auckland
20 Symonds street
Auckland, New Zealand
xun.xu@auckland.ac.nz

ABSTRACT
In recent years, Cloud Manufacturing concept has been proposed by taking advantage of Cloud Computing to improve the performance of manufacturing industry. Cloud Manufacturing attempts can be summarized as two sectors, i.e. manufacturing version of Computing Cloud, and a distributed environment that is networked around Manufacturing Cloud. It is important to understand the existing manufacturing resources and implement them properly in the Cloud. In this paper, manufacturing resource, ability and relevant essentials are discussed in the service-oriented perspective. The functional requirements of a Cloud Manufacturing environment are discussed, along with an interoperable manufacturing system framework.

Keywords: Cloud Manufacturing, Cloud Computing, Interoperable Manufacturing

INTRODUCTION
Nowadays, Cloud Computing (CComputing) is gaining

Hence, it is a logical thinking of adopting Cloud concept into manufacturing context to improve the production and enterprise performance. A number of manufacturing models via Cloud have been recently proposed (Li et al., 2010; Tao et al., 2011; Rauscher et al., 2011; Tefkaj et al., 2012). Cloud-based manufacturing comes under two categories. The first type is the “manufacturing version” of Cloud Computing, i.e. directly adopting CComputing technology in manufacturing. In this kind of approach, manufacturing software tools, data storage space, and management method can be deployed in the Cloud (Chaliki and Hosamanna, 2013; Wu, 2011; Chandrasekara et al., 2012; Chai et al., 2011). For example, deploying CAx software on the Cloud increases flexibility, extensibility, integrity and easy/limited data storage.

From the provider’s point of view, applications can be easily maintained and utilized on the Cloud server. Version updating, maintaining and integrating can be remotely finished, avoiding periodic service from on-site maintenance specialists historically. Without the costs of freight, on-site setup, transportation and so forth, consumer’s applications are
D. Academic Posters

II. **X. V. Wang** "DIMP: Distributed Interoperable Manufacturing Platform", EPS postgraduate poster competition, 14th – 19th September, 2011, 4th floor Atrium, Faculty of Engineering, University of Auckland.
III. **X. V. Wang** "Moving to the Cloud: an Interoperable Cloud-based Manufacturing System (ICMS)", EPS postgraduate poster competition, 27 August - 6 September, 2012, 4th floor Atrium, Faculty of Engineering, University of Auckland.