Product range models in injection mould tool design

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

Additional Information:

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

Metadata Record: https://dspace.lboro.ac.uk/2134/7260

Publisher: © Carlos Alberto Costa

Please cite the published version.
This item is held in Loughborough University’s Institutional Repository (https://dspace.lboro.ac.uk/) and was harvested from the British Library’s EThOS service (http://www.ethos.bl.uk/). It is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
PRODUCT RANGE MODELS IN INJECTION

MOULD TOOL DESIGN

By

Carlos Alberto Costa

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of Loughborough University

July 2000

© by Carlos Alberto Costa (2000)
"Life is an opportunity, benefit from it.
Life is beauty, admire it.
Life is a dream, realise it.
Life is a challenge, meet it.
Life is a duty, complete it.
Life is a game, play it.
Life is costly, care for it.
Life is wealth, keep it
Life is love, enjoy it.
Life is a mystery, know it.
Life is a promise, fulfil it.
Life is sorrow, overcome it.
Life is a song, sing it.
Life is a struggle, accept it.
Life is a tragedy, confront it.
Life is an adventure, dare it.
Life is luck, make it.
Life is too precious, do not destroy it.
Life is life, fight for it."

Mother Teresa.
DEDICATION

To Ziza,

Fernanda, Giovanni and Felipe,

and to my parents.
ACKNOWLEDGEMENTS

"In ordinary life we hardly realise that we receive a great deal more than we give, and it is only with gratitude that life becomes rich. It is very easy to over-estimate the importance of our own achievements in comparison with what we owe others."

Praise and thanks to God, for everything YOU always give me.

This work has been carried out at Loughborough University and financially supported by CNPq ("Conselho Nacional de Desenvolvimento Científico e Tecnológico" - Brazil) and UCS (University of Caxias do Sul - Brazil).

Many people have directly or indirectly supported this research and I would like to thank all of them, in particular my friends in room GG101 for sharing their lives and friendship with me during these years of my work. Thank you very much Manuel, Denny, Jie, Shahin, Bing, Nilo, Osiris, Aysin, Muzaffer, Zoha and both Richards, you have been good friends.

Naturally, there are also some special thanks yous.

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

Jenny Hardings and Keith Toh for your friendship, technical discussions and unconditional help and support. I wish you every success in every step of your lives.

Moss Plastic Parts for providing industrial experience to support my work.

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

My parents for all your support, and a special thanks for your practical support and visits over the last four years.

My brothers, sisters and my wife’s family thank your for your constant attention, help and contact. You managed to shorter significantly the physical distance between us.

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

The use of information models, which may be shared by many different software applications to support activities throughout the whole product life cycle, is recognised as a powerful approach to support integrated, team based product development. While much work has been done into the concepts of Product and Manufacturing Models, there is a need to explore the definition of information models to support reuse of design information.

The research reported in this thesis explores the structure of a Product Range Model, which can capture information and knowledge, based on design history, of the range of ways in which a product can be designed. The work uses injection mould tooling as a type of product against which to pursue the research. The relationships between design functionality and potential solution sets for injection mould tool designs have been explored. Major emphasis has been placed on understanding the interactions, which take place between the range of potential solutions and the requirements of each particular design.

A Product Range Model structure has been defined based on four main interacting classes; Functions, Design Solution, Interactions and Knowledge Links. An experimental object oriented system has been produced which comprises a Product Range Model, a Product Model and related decision support applications. Experiments have been performed to demonstrate how a Product Range Model can offer valuable support to design re-use. The research contributes to the understanding of the general structural requirements of such model based information systems. It shows how valid design options can be provided through the definition of the Interactions class. It also shows how Knowledge Links can be defined and used to provide a mechanism to identify critical information in a Product Model needed by each Interaction element.
# TABLE OF CONTENTS

1. AN INTRODUCTION .......................................................... 1

2. LITERATURE REVIEW ....................................................... 6

2.1. Introduction .............................................................. 6

2.2. Information Support Systems ........................................ 6

2.3. Information Modelling: requirements, applications and exploitation ......................................................... 8

2.3.1. Product Modelling ................................................... 8

2.3.2. Software Applications Modelling ................................ 12

2.3.3. Modelling References, Methods and Tools ....................... 12

2.3.4. Architecture systems to support CE ............................. 15

2.3.4.1. MOSES (Model Oriented Simultaneous Engineering System) ..................................................... 15

2.4. Engineering Design Information Reuse ................................ 18

2.4.1. Computational Support to Engineering Design ............... 18

2.4.2. Redesign .............................................................. 20

2.4.3. Design Reuse ......................................................... 22

2.4.3.1. Design Reuse Concepts ...................................... 22

2.4.3.2. Design Reuse Computational Approaches ................... 23

2.4.4. Product Architecture ................................................ 28

2.4.5. Functional Design .................................................... 31

2.5. Injection Moulding Software Tools .................................. 34

2.5.1. Injection Mould Design ........................................... 34

2.5.2. Classification of Injection Mould Design Support Systems 35

2.5.3. A Framework for Injection Mould Design Research Activities ......................................................... 36

2.5.4. Specific Support Systems .......................................... 37

2.5.4.1. Initial Design Decisions Support ........................................... 37

2.5.4.2. Main Injection Mould Systems Design Support ........... 38

2.5.5. Integrated Support Systems ....................................... 39

2.5.6. Information Based Support System .............................. 44

2.6. Summary ................................................................. 46
3. COMPUTER AIDED ENGINEERING SYSTEM ARCHITECTURE - THE RESEARCH ENVIRONMENT

3.1. Introduction

3.2. Research Environment
   3.2.1. General Description
   3.2.2. MOSES system architecture
   3.2.3. RM-ODP (Reference Model for Open Distributed Process)
   3.2.4. UML (Unified Modelling Language)
      3.2.4.1. UML Diagrams
      3.2.4.2. A process to support UML notation

3.3. Computational Tools
   3.3.1. ObjectStore Database
   3.3.2. Visual C++
   3.3.3. Rational Rose 98 Enterprise Edition

3.4. Industrial Collaborator

4. PRODUCT RANGE MODEL CONCEPT FOR INJECTION MOULD DESIGN

4.1. Introduction

4.2. Computational support for injection mould design
   4.2.1. General Issues
   4.2.2. Information modelling to support injection mould design reuse
   4.2.3. The nature of a Product Range Model

4.3. Product Range Models supporting information reuse in injection mould design
   4.3.1. Injection Mould Product Range
   4.3.2. Function and Design Solutions Relationships
   4.3.3. Design Solutions and Product Information Relationships
   4.3.4. Information Model Relationships

5. REPRESENTING INJECTION MOULDS AS A PRODUCT RANGE

5.1. Introduction

5.2. Injection Mould Product Range
   5.2.1. General Injection Mould Structure
   5.2.2. Injection Mould Functions
   5.2.3. Injection Mould Design Solutions
7.3.3. Composite Interactions
  7.3.3.1. General Definition
  7.3.3.2. OR_Interaction
  7.3.3.3. AND_Interaction

7.4. Interactions Sets Supporting Design Solution Decisions
  7.4.1. Requirements to Support the Design Solution Decisions
  7.4.2. Interactions States
  7.4.3. Interaction States Associated with Design Solutions

7.5. Interaction Testing Process
  7.5.1. General Interaction Testing Process
  7.5.2. Simple Interactions Testing Process
  7.5.3. Composite Interactions Testing Process
    7.5.3.1. General Testing Process
    7.5.3.2. OR_Interaction Testing Process
    7.5.3.3. AND_Interaction Testing Process
    7.5.3.4. Composite Interaction Testing Process Sequence
  7.5.4. Evolution of the Interactions conditions
  7.5.5. Reassessing Interactions of DS already Chosen

7.6. Summary

8. PRODUCT MODEL AND PRODUCT RANGE MODEL
   RELATIONSHIPS

8.1. Introduction

8.2. General Relationships between Product Range and Product Information

8.3. The Product Model Structure
  8.3.1. General Aspects
  8.3.2. Injection Moulding Product Model Information Structure
    8.3.2.1. Product Specification Structure
    8.3.2.2. Design Solutions Structure

8.4. Knowledge-Links to enable Information Model Relationships
  8.4.1. General Aspects
  8.4.2. Knowledge-Link Elements Definition
  8.4.3. Retrieving Product Information from Product Model
    8.4.3.1. Information Retrieval
    8.4.3.2. Information Location
    8.4.3.3. Knowledge-Link Information Structure
8.5. Requirements for Storing Design Decision Information in the Product Model 138
   8.5.1. General Aspects 138
   8.5.2. Product Data Information Relationships 139

8.6. Summary 141

9. EXPERIMENTAL SYSTEM IMPLEMENTATION 142
   9.1. Introduction 142
   9.2. Design and Implementation of the Experimental System 142
       9.2.1. Objectives and Scope 142
       9.2.2. Experimental System Environment and Description 143
       9.2.3. Visualisation of the Results 145
       9.2.4. Injection Mould Information 145
   9.3. Product Range Model Structure 148
       9.3.1. General Description of the Experiment 148
       9.3.2. Injection Mould Functions 148
       9.3.3. Injection Mould Design Solutions 151
       9.3.4. Injection Mould Interaction Elements 154
           9.3.4.1. Simple Interactions 156
           9.3.4.2. Composite Interactions 158
       9.3.5. Knowledge-Links 163
   9.4. Product Range Model Supporting Functional Design Through the Reuse of
       Information and Knowledge 165
       9.4.1. General description of the experiment 165
       9.4.2. An Injection Mould DFF application - Basic Dialogues 166
       9.4.3. Design solutions interactions with initial product specifications 169
       9.4.4. Interactions with Design Solutions Decisions 176
       9.4.5. Changing the Product Specification during the Design Process 180
       9.4.6. Storing Final Design Solutions in the Product Model 181
       9.4.7. Composite Interactions Behaviour 184
   9.5. Evaluation of the Results 190

10. DISCUSSION, CONCLUSIONS AND FURTHER WORK 191
   10.1. Introduction 191
   10.2. Discussion 191
   10.3. Conclusions 195
1. An Introduction

The need to manufacture products of good quality, at competitive prices and "on time" are objectives which drive most manufacturing companies aiming to improve their position and future in their markets place. To achieve these objectives, information from different phases of product life cycle should be taken into consideration in the early stages of the product design process. Philosophies, such as Concurrent Engineering, propose better ways and structures to cope intelligently with the different design activities. It allows experts, involved with the different stages of product design and manufacturing, to work together in teams, sharing information to support the decision making processes of design. However, to gain the full benefits from Concurrent Engineering adoption, besides changes in the company's organisational structure, it is also important to implement software systems, which support this new way of working.

Significant advances in the computational power of desktop computers have contributed to the increased use of software applications, e.g. CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), Analysis and Simulation packages, to support different design activities. However, to work within an integrated approach, such as Concurrent Engineering, these tools should be able to cooperate with each other to provide more efficient and less error-prone product development and manufacture. One approach, to support such cooperation, is to use common information structures, which may be shared by many different software applications for activities throughout the whole product life cycle. This information model based approach has been used in this research.

The work reported in this thesis contributes to the area of information modelling to support product design and manufacture, and focuses on injection mould tooling as the product area. Injection mould tool design is one of the major phases of injection moulded component design and manufacture, and has a significant influence on the final quality and cost of such components.

In injection mould design, the injection mould tool can be characterised as a kind of product range, where the design concept is well understood. Therefore, there will be variants in the design process that use different pieces of information, and these need
to be considered together, to achieve a well-balanced design solution. Hence, the capture and modelling of such information into information structures is necessary to support the injection mould integrated design process.

Commercial systems to aid the design of injection moulds are available, such as Moldflow© and C-Flow©. These systems support the design analysis of particular functionalities of the mould, e.g. feeding, cooling, etc. Although useful support is provided by these systems, they appear as independent and specific applications within a mould design environment, focusing on design analysis rather than integrated design decision support.

The concept of information models, which support the reuse of design experience, has significant potential in the design of injection mould tooling. Much work has been done into the concepts of Product Model and Manufacturing Model, where a data representation of product and manufacturing information is captured, respectively. Such information models support information sharing between design team members, which is a major contribution to the effectiveness of the design process. However, further value can be obtained from such models if they are tailored to support reuse of design information. To do this an information model should also capture the information and knowledge built through time about a specific range of products. The author’s research is based on the thesis that product range information and knowledge can be represented in a model, which can support design reuse within an integrated design environment. The representation of information and knowledge to characterise a product range is termed the Product Range Model.

In this thesis the definition of the terms information and knowledge are adopted as in (Harding 1996). Information is defined as structured data that has some meaning and knowledge as information with added value that relates to how it may be used or applied. Examples of information in the context of this research are: mould configuration, which can be 2 or 3 plates; mould function, which can be eject plastic component, feed mould, cool core; mould design solutions, which can be eject pins; H-Type runner layout, matrix impression distribution. Examples of knowledge are, for instance, the conditions for applying specific design solutions, such as to use a submarine gate the feed system of the mould must be a cold type; to use a X-Type runner layout the impressions must be in a matrix distribution; to use stripper plates the geometry of the plastic component, in the parting line, must be not complex.
To achieve a suitable representation of information and knowledge, the relationships between design functionality and potential solution sets for the injection mould tool design have been explored. Major emphasis has been placed on investigating the interactions, which take place between potential solutions for satisfying particular design specifications.

The aim of this research has been defined as "to provide a greater understanding of the role and information structure of a Product Range Model together with its relationships with other information models in variant design software systems for injection moulds". In order to explore the research thesis, the objectives of the research can be stated as:

I. To understand the concepts involved in supporting variant design of injection moulds;

II. To understand works that have contributed in the area of injection mould design support tools;

III. To understand the mould system elements design and how these elements interact with each other and with other information during the injection mould design process;

IV. To understand the process of modelling information structures and application functionality in integrated CAE system architectures;

V. To define a product range information structure to represent injection mould systems elements in a variant design environment;

VI. To explore the relationships between a Product Range Model, Product Model and application software, to provide design decision support;

VII. To build an experimental system to test the Product Range Model concept;

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

This chapter sets the work into context for readers. This thesis has been organised into ten chapters and the overall structure and contents are depicted in Figure 1-1. Chapter 2 presents a survey involving relevant areas of related works. It starts with the main issues that must be considered in product information modelling, moves to the reuse
Chapter I

of information in the design process and finally provides a survey of the main contributions of injection mould design support tools.

Chapter 3 provides a description of the "environment" used to define the boundaries of this work and the development of an experimental system to check the ideas and concepts. A description of the RM-ODP (Reference Model for Open Distributed Processing), UML (Unified Modelling Language) notation and the methodology and tools used to guide the development of this research are provided.

Chapter 4 highlights the contribution of the work in the context of the problem area that the research has addressed. This chapter also defines the contents of chapters 5, 6, 7 and 8, which provide a description and understanding about the nature of the Product Range Model based on the injection mould product range.

Chapter 5 provides a representation of the injection mould as a kind of product range, identifying the general relationships that must be captured in the Product Range Model. Functions and design solutions related to the main injection mould systems are described, as well the kind of interactions that take place during the injection mould design process.

Chapter 6 describes function and design solution information structure representation in the Product Range Model that makes the foundation for the critical issues investigated in this research, i.e. Interaction elements, which are addressed in Chapters 7. Chapter 8 investigates the required information relationships between the Product Range Model and the product model, to support the evaluation of the Interaction elements and the capture of the design results of the Product Range Model.

Chapters 9 provides a description of the experiments conducted to explore representation of the Product Range Model and to demonstrate the extent to which is can provide support to design process.

Finally, Chapter 10 presents the conclusions achieved by this research, as well as the recommendations for further developments in this area.
Chapter 1
Introduction

Chapter 2
Literature Survey
• Information modeling
• Design reuse
• Injection Mould tools review

Chapter 3
Research Work Environment
• RM-ODP; UML; Methodology; Tools

Chapter 4
Product Range Model Concept
• Problem area discussion

Chapter 5
Injection Mould as a Product Range

Chapters 6, 7 and 8
Product Range Model Structure
• Functions, Design solutions and Interactions Relationships
• Information models relationships

Chapter 9
Product Range Model Experimental System

Chapter 10
Discussions of the results
Conclusions
Recommendation for further works

Figure 1-1 – Thesis Structure Description
2. Literature Review

2.1. Introduction

This chapter presents some of the main topics currently researched and related to the main theme of this thesis, information modelling for supporting injection mould design reuse. It also provides background information for further discussions in other chapters of this thesis. Section 2.2 describes an overview of information support systems into concurrent engineering environment, followed by main aspects involved in product information modelling, described in section 2.3. Section 2.4 reviews issues related to design reuse and is followed by section 2.5, which reviews researches on computational tools to support injection mould design.

The review realised in this chapter is critically accessed in chapter 4, which highlights the contribution of the work in the context of the problem area that this research has addressed.

2.2. Information Support Systems

The demand for higher quality and lower cost products with shorter development lead-times has forced industries to focus on new product development strategies. It is believed that most of the total life cycle costs of a product (65-90%) are determined at the design stages (Venkatachalam, Mellichamp et al. 1993; Dowlatshahi 1994).

Concurrent Engineering (CE) or Simultaneous Engineering (SE), has been recognised as a feasible approach to improve design efficiency, where the design of a product and all its related processes in the manufacturing system are taken into consideration simultaneously (Sohlenius 1992; Syan 1994; Molina, Al-Ashaab et al. 1995). Two approaches are usually addressed for the CE implementation, named team based and computational based (Jo, Parsaei et al. 1993).

From the computational implementation viewpoint, the CE paradigm can be viewed as an integration of functional software tools, which aim to support design team members to share knowledge and information, and to keep track of the other's needs, constraints, decisions and assumptions (Harrington, Soltan et al. 1996; Tichem and
Storm 1997; Urban, Ayyaswamy et al. 1999). (Dowlatshahi 1994) presents and discusses different approaches for computational implementation of CE.

Thus, to support these requirements related to integration and information sharing, within a flexible computer based structure, two main elements must be considered in the development and implementation of integrated information supported systems:

- Common and structured source of information, and
- Software applications.

The first element, also named information models, provides an information repository, which is used to capture the information related to the life cycle of an artefact. To represent this information, well-defined information structure, or information data model, are required (McKay, Bloor et al. 1996). This element stores company information (Figure 2-1).

The second element is responsible for supporting the life cycle functional activities involved in the product development, such as design and manufacturing. This element shares information stored in the information models, and hence it is created based on the first element, i.e. information data model. For this reason such element is also named data model driven applications (Young, Canciglieri-Jnr et al. 1998) (Figure 2-1).

![Diagram](image-url)  
*Figure 2-1 – The general information system concept (Young, Canciglieri-Jnr et al. 1998)*
This research is primarily concerned with the first element, i.e. information models, to support the reuse of information in design.

2.3. Information Modelling: requirements, applications and exploitation

2.3.1. Product Modelling

The design process can be defined as a series of activities by which the information about the designed object is changed from one information state to another, and hence solve a design problem (Dixon 1995). Therefore, a clear and unambiguous definition and representation of the information involved in the whole product development process, has become a vital issue to support not only computer design applications, but also other life-cycle applications, such as manufacturing applications. Such representation has been advocated by different authors through the concept of product modelling (Kimura 1992; Krause, Kimura et al. 1993; Anderl 1997).

Product modelling aims to provide a consistent representation of products information, which can be reached and used by one or more software systems during all stages of product design and manufacture.

(Anderl 1997) states that basic approaches of product modelling are:

- Provision of tools to support product description throughout all product life-cycle phases and the development and design phases in particular;
- Interdisciplinary modelling features, integrating various developments disciplines, such as mechanics, electrics, electronics and software, and
- Parametric and constraint modelling, aiming at the representation of design decisions and design know-how.

(Krause, Kimura et al. 1993) address different types of product model, as following:

- Structure-oriented product models: represented by the structure of the product, such as, bill-of-material or structure of classification;
- Geometry-oriented product models: computer internal models with the primary propose of representing the shape of one specific product. Typically used as part
of basic CAD systems, they provide a basis for further computational applications;

- Feature-oriented product models: represent often used shapes patterns such as coherent geometric items, called form features. Works on this subject have been extensively explored in the literature (Rosen 1993; Salomons, van Houten et al. 1993; Chen, Swift et al. 1994; Allada and Anand 1995);

- Knowledge-based product model: are characterised by the use of AI (Artificial Intelligence) techniques like object-oriented programming, ruled-based reasoning, constraints and truth maintenance systems;

- Integrated product models: cover the abilities of the models above. All types of product information can be stored in an integrated product model, and

- Model standardisation using STEP: one of the most significant approaches towards the implementation of integrated product models. The STEP (Standard for the Exchange of Product Model Data) defines a neutral data format for the representation and exchange of product data, through the ISO 10303, particularly 4x's series of Parts (Ashworth, Bloor et al. 1996). The goal is the complete and system-independent representation of all products related data during the product life cycle (Gu and Chan 1995). The STEP standard documentation is partitioned into different parts and is based on a three-layer architecture, which is composed by integrated resources, application protocols, implementation methods and description methods (Ghodous and Vandorpe 1998).

(Ashworth, Bloor et al. 1996) address the potential application of STEP in architecture product data representation to support a range of engineering requirements, and explain the methodology to produce the exchangeable data models, which is focused on integrated resources. Hence, such models can be used for both the product database implementation and the application programming interface within a collaborative environment (Goh, Hui et al. 1996; Jasnoch and Haas 1996; Urban, Ayyaswamy et al. 1999). However, besides the advances provided by the use of STEP, its applications for product modelling have been primarily focused on the geometric aspects, which is part of the product information (Gu and Chan 1995). Recent efforts within STEP, such as Application Protocol 214 (AP 214) start to take in consideration besides the modelling of the geometry, also the product
organisational data, for modelling generic product structures (Mannisto, Peltonen et al. 1998).

It is recognised that only geometric information of a product is not enough to support downstream phases of product development in an integrated environment. Product modelling can provide a high-level computer interpretable communication because it stores and communicates the definition of a product rather than its geometric representation or presentation, i.e. paper (Kunz, Luiten et al. 1996).

(McKay, Bloor et al. 1996) address that for the computational representation of product data two elements are required namely, product model, which contains the actual data that is specific to the particular product, and product data model, which defines the form and content of the product data. Three technologies are identified as required to support the implementation of information systems based on product data, namely: (I) product database, which stores the product data; (II) product data model, which defines the form of the product data that applications operate upon, and (III) the product data editors and applications, which allows the product data to be manipulated by people and software respectively. To maintain the independence of the product data in relation to the database technology used to the implementation and data editor and applications, three criteria are applied: conceptuality, extensibility and structural integrity. The authors present a framework for product data, which supports the description of electromechanical products.

(Lei, Taura et al. 1996) define product data as “the facts, concepts or instructions about a product or a set of products in a formal manner suitable for communication, interpretation or processing by human or by automatic means”, and product data model as “information model which provides an abstract description of the product data”. Thus, the product data is evaluated and changed through different stages of a collaborative design process, at different levels of abstraction. The authors propose a more expanded concept for product data model, which captures design history, design alternatives, assignments, etc., together with constraints among decisions.

Similar definitions have been addressed by (Xue, Yadav et al. 1999) in relation to product features, through the concepts of class features and instance features. While the class features are defined in terms of product data structure, instances features (actual product data) are generated using the class features as their templates.
In this thesis the term "information structures" is also used meaning information data models. For example, product model information structure, which means product data model.

(McKay, Bloor et al. 1997) states also that creation of product data model can be performed by two ways, named: product data modelling and product data engineering. The former is a bottom-up approach and based on the applications/tools a product data model is defined. However, such an approach has the drawback of not guaranteeing integrity of the information model, once that it has been built mainly upon particular applications "interest" (views). On the other hand, the later way, considered a top-down approach, looks at the engineering process and information requirement rather than the application, which provides more integrity in the final product data model.

Therefore, the definition of the product data model will be significantly influenced by the way that information is required and manipulated by different product life cycle activities.

(Ghodous and Vandorpe 1998) state that to facilitate the integration of CAX/CAY systems and adequately support engineering and managerial activities, it is not sufficient to represent only data between processes, but it is also necessary to represent the processes. The authors present an integrated metamodel for simultaneous and normalised modelling of products and their development processes.

In addition to the product modelling information, manufacturing modelling information has been addressed as an important aspect to support different applications that are part of the product life cycle, e.g. design for manufacture, CAPP, etc. (Kimura 1992). Works in this area are identified as related to application of STEP to design and development of a manufacturing resource data (Kjellberg and Bohlin 1996; Al-Ashaab and Young 1997); ontology to supports a new process modelling approach for manufacturing resources and activities (Bonfati, Monari et al. 1995); representation and capture of manufacturing information and knowledge (Molina, Ellis et al. 1995); modelling of integrated product and manufacturing information to support process planning (Rudas and Horváth 1997) (Gao and Huang 1996). However, such topic will not be covered by this research work.
2.3.2. Software Applications Modelling

While product modelling supports the definition of a more suitable data structure for providing integration and information sharing, software applications capture mainly the functionality involved in the engineering process.

Applications act as pieces of software, which has knowledge in specific areas of a product life cycle, such as DFx (Design for X, where X can be manufacturing, assembly, function, etc.), and that support the design and manufacturing of such a product (Tichem and Storm 1997). Also, applications are responsible for the main link between the information system and the end user.

Depending on the context, different names and definitions are addressed to the applications, such as agents (Harding and Popplewell 1996; Harrington, Soltan et al. 1996), actors (Tichkiewitch 1996) or supporting tools (Tichem and Storm 1997).

The development of software applications into an integrated design environment is highly dependent on the information data model with which they share information and is a reason why these applications are also called data model driven applications (Young, Canciglieri-Jnr et al. 1998). Also, these software applications are significantly influenced by user requirements, e.g. graphical interfaces and a specific process to be followed.

2.3.3. Modelling References, Methods and Tools

As information involved in the development of an artefact can be very complex in engineering practice, open system architecture must be used as basis for the implementation of product modelling technologies, allowing therefore a common description and comparison between information systems. Also, powerful and disciplined methodologies and languages are required to capture and represent this information data and the way that it is changed (processes), providing a reference for discussions and software systems design and implementation (Court, Culley et al. 1995; McKay, Bloor et al. 1997).

(Kosanke and Vernadat 1999) summarises main works carried out in the area of enterprise integration in the last years, identifying 4 main areas: systems integration, application integration, business integration and enterprise integration. While the last
one appears as a tendency into the integration issue, the volume of information involved is significantly higher.

CIM-OSA (Kosanke 1995; Kosanke and Vernadat 1999), PERA (Purdue Enterprise Reference Architecture), GRAI-GIM (Doumeingts, Marcotte et al. 1995; Carrie and Macintosh 1997) are examples of reference models that allow describing an integrated system, enabling the creation of enterprise models and taking into consideration different viewpoints (information, function, recourses, organisation, etc.). However, such references are focused on enterprise integration and representation, which is not the main goal of this research.

ISO/RM-ODP (International Standard Organisation/Reference Model for Open Distribute Processing), OMG/CORBA (Object Modelling Group/Common Object Request Broker Architecture) and OSF/DCE (Open Software Foundations/Distributed Computing Environment) are examples of some standards for open distributed processing, which aim to enable interaction between systems and applications. Such standards allow also developing system architectures which are themselves open, achieving interoperability and portability among their individual component systems (Blair, Coulson et al. 1996). However, such standards define a general framework upon which the system must be developed, and formal languages or methodologies to represent specific levels of such framework are still required.

The RM-ODP is an important reference model for this research in the way that it provides a reference for describing open distributed systems in different viewpoints. Such reference models will better explained in Chapter 3. However, the RM-ODP mainly highlights the content of the essential views of the system, and does not dictate either, how the information system should be designed and implemented, or which tools, techniques or even syntax and semantic should be used to design and implement the requirements represented at each viewpoint. Therefore, to support and guide the design and implementation of the software system consistently and efficiently, progressing through each level of the RM-ODP, an application of a formal language (notation) associated with a method/process is required.

(Molina, Ellis et al. 1994; McKay, Bloor et al. 1997) have addressed how some of these tools, such as IDEF0 (ICAM DEFinition language for functional modelling),
Booch Methodology (Booch 1994) can be applied as languages for describing a system into RM-ODP.

(Court, Culley et al. 1995) compares some of the different kinds of methods and techniques for modelling engineering design information, based on different criteria. However, such comparison is not made upon a reference model view.

In the last years, object-oriented technology has gained significant attention as a programming tool because of its advantages in the handling of complexity, modularity, encapsulation, reusability, extensibility and abstraction of real-world objects (Taylor 1992; Zhou, Greenwell et al. 1994; Wainwright, Leung et al. 1996). This provides the facilities for information retrieval and communication (Ghodous and Vandorpe 1998).

As a result, object oriented technology has migrated from simple and isolated languages to more comprehensive methods, which allow the analysis, design and representation of a whole information system. (Monarchi and Purh 1992) present a comparison among some of this methods.

Also, when applied to integrated and concurrent design environments, where information models become sources of data and knowledge for different applications, object oriented technology can bring also significant advantages upon other technologies previously used, e.g. relational databases (Kung, Du et al. 1999).

For this reason, the use of object oriented representation for product and process design has gained substantial attention by the literature (Shen and Barthès 1997; Gorti, Gupta et al. 1998; Kung, Du et al. 1999). (Ghodous and Vandorpe 1998) show some of the relationships between object-oriented models and EXPRESS, which is an "object-flavoured" information specification language, to support the transformation and unification of different product and process data models.

The research work on this thesis applies the object-oriented technology for the definition of information structures and development of software applications.

One of the last and more significant developments into object oriented technology analysis, design and representation is the UML (Unified Modelling Language). It allows capturing since the functionality until the final representation of the object and their properties into an object oriented system (Texel and Williams 1997; Booch,
Rumbaugh *et al.* 1999; Jacobson, Booch *et al.* 1999). Such notation and its relation to this research are explained in more details in Chapter 3.

### 2.3.4. Architecture systems to support CE

The three elements addressed above, product modelling, applications modelling and reference, methods and tools, should be taken into consideration in any system that purport to support CE activities into an integrated environment. Attempts have been made in defining such environment based on different approaches and methodologies, such as PACT (Palo Alto Collaborative Testbed) (Cutkosky, Engelmore *et al.* 1993), SHADE (SHAred Dependency Engineering) (Olsen, Cutkosky *et al.* 1995), SHARED (Gorti, Gupta *et al.* 1998).

However, since this research has been significantly influenced by MOSES (Model Oriented Simultaneous Engineering System) architecture, such a system is focused on this thesis, and is described in the next sub-section.

#### 2.3.4.1. MOSES (Model Oriented Simultaneous Engineering System)

The MOSES (Model Oriented Simultaneous Engineering System) project has been researched at the Manufacturing Engineering Department of Loughborough University in cooperation with Leeds University under ACME funding. The specification of the research into MOSES focused on a computer based system that (I) provides product and manufacturing information, (II) enables decision support based on these information sources and (III) is co-ordinated in a manner that makes it suitable for operation in a simultaneous engineering environment (Ellis, Molina *et al.* 1995).

The basic architecture of the MOSES is depicted in Figure 2-2, where three main elements are identified: information data models, integration environment and software applications environment. Two information models have been implemented as object-oriented databases, named Product Model and Manufacturing Model, which are linked to an open set of computer applications, by an integration environment. An application called Engineering Moderator ensures that the evolving product design considers the different life cycle activities that are represented by the application environments. The application configuration and functionality will be dependent on the needs of the host organisation (e.g. DFM, DFA, etc.).
The ISO/RM-ODP (ISO/IEC 10746-1) was taken as reference model to guide the MOSES research in defining a CAE-RM (Computer Aided Engineering - Reference Model), enforcing, therefore, the generic and modular characteristics desirable in this model. The definition of the first three viewpoints, e.g. enterprise, information and computational viewpoints, has been mainly focused (Lim, Juster et al. 1997). For the modelling of these different viewpoints of the system, Booch Object Oriented Methodology (Booch 1994), IDEFO (Colquhoun, Baines et al. 1993) and EXPRESS (ISO CD 10303-11) methodologies and standards have been applied (Molina, Ellis et al. 1994).

The following sub-section describes each of the components of MOSES architecture.

2.3.4.1.1. **Information Models**

As mentioned above, two information models have been explored in MOSES research, named Product Model and Manufacturing Model.

The Product Model contains information about the product related to its life cycle and is based, wherever possible, on the evolving STEP standard. The Product Model is a source and repository of information for many applications, and, as such, allows information to be shared between the many users and software components of the CAE system. Thus, all agents involved in the design process (humans or software) must actively participate in information sharing by utilising the common product model database.
Chapter 2

The Manufacturing Model describes available manufacturing process (injection moulding, machining processes, etc.), resources (machines, tools, fixtures, etc.) and strategies (how these resources and processes are used and organised), providing a consistent source of manufacturing information for applications. It has four levels based on *de-facto* standard namely, Factory, Shop, Cell and Station, which represent the functionality of the manufacturing facility of any firm. It has been modelled using information related to machining and injection moulding (Al-Ashaab 1994; Molina, Ellis *et al.* 1995).

The concepts of Product Model and Manufacturing Model in MOSES has been explored into the field of injection moulding design (Al-Ashaab 1994; Lee 1996). Such works are reported in Chapter 3 of this thesis.

2.3.4.1.2. Strategist Applications

Strategist applications are specialist expert applications, which assist users of CAE systems in the evaluation, modification and extension of the product design using criteria, which are closely allied to particular design perspectives. The strategists can be refereed as different DFx applications (2.3.2) with particular interests in the different phases of a product design and manufacture. Therefore, each application will hold its own criteria, rules and knowledge, which the final product should respect. The MOSES project has focused on two particular kind of applications: Design for Manufacturing (DFM) (Al-Ashaab 1994; Al-Ashaab and Young 1995) and Design for Function (DFF) (Lee 1996; Lee and Young 1998) perspectives.

2.3.4.1.3. Integration Environment

An Integration Environment is required to enable the many elements of the MOSES CAE System to work together, even though they may be distributed over many computing platforms, and even located at several sites. The integration environment must satisfy the requirements of each individual element and provide support for interactions and communication between applications. However such an environment is not very related to this research.

2.3.4.1.4. Engineering Moderator
Chapter 2

The Engineering Moderator is a specialist manager or coordinating program whose role is to provide driven concurrency within the MOSES system. The engineering moderator must be capable of (Harding and Popplewell 1996):

- Promote communication and negotiation between design agents (combination of strategists applications and human designer);
- Identify that significant problems may have occurred in the design;
- Determine the course of action to follow when a significant problem is identified, and
- Maintain communication between interested agents until the conflict of interest has been resolved.

The Engineering Moderator works by monitoring and analysing the proposed input to the product model in order to access whether the user, or any of the application environments, should be advised of the design change. To achieve this, the Moderator has an in-built knowledge base that contains details of the types of product information change, which are particularly sensitive to each application environment. Three expert modules composed the Engineering Moderator: Knowledge Acquisition Module, Design Moderation Module and Design Agent Module. (Harding and Popplewell 1996; Popplewell and Harding 1996) show more details about the structure and the contents of this application. As the integration environment, this element is not related to this research.

2.4. Engineering Design Information Reuse

2.4.1. Computational Support to Engineering Design

The development of a product involves several activities and steps in order to conceive, design and commercialise it. (Ulrich and Eppinger 1995) present these steps as Concept Development, System-Level Design, Detail Design, Testing and Refinement and Production Ramp-up phases, where the design phases have been recognised as some of the most important activities during a product cycle life.
Dixon 1995 defines the design process as a series of activities by which the information about a designed object is changed from one information state to another, and hence solve a design problem. Such a process requires different types of knowledge. This opinion is also shared by (Court, Culley et al. 1995).

Evbuomwan, Sivaloganathan et al. 1996 present a comprehensive review about design, discussing such aspects as definitions, theory and methodology, classifications, philosophies, models, methods and systems. The authors describe three main kinds of design models, namely: prescriptive, descriptive and computational. While prescriptive models tend to look at the design process from a global perspective, covering the procedural steps, e.g. (Pahl and Beitz 1996), descriptive models are concerned with designers' actions and activities during the design process, e.g. Suh's Axiomatic Approach (Suh 1995). Computational models are related to the application of computational technologies to support design (Al Hamando and Kumura 1994). This research is mainly concerned with computational support for design reuse, more specifically focused on an information based approach.

To provide computational support for design activities, models that represent such activities, i.e. information and process models, should be understood and built (Krause, Kimura et al. 1993; Al-Salka, Cartmell et al. 1998).

Hsu and Woon 1998 address that to support design processes two main difficulties must be overcome: modelling the complex interaction between various facets of a product and reasoning about the generation and selection of feasible solutions. The authors survey several techniques for modelling representation (e.g. languages, geometric models, objects, etc.) and reasoning techniques (e.g. CBR, KBS, neutral networks, etc.). The reasoning techniques are still classified as driven by data or by knowledge, depending on the quantity of information that it requires.

Grabowski, Lossack et al. 1996 address two main approaches used in design systems, named process oriented (strategies for solving design problems) and information oriented (modelling information needed for design). An architecture of a knowledge based design system is presented encompassing components of both approaches.
(Ullman 2000) states that design work has focused on the mantra "design is the evolution of information punctuated by decision-making". Definitions of data, model, and knowledge are presented, as well as how each of these information sources relates to another, to support design decision-making.

(Gorti, Gupta et al. 1998) address the importance of modelling the design process together with the modelling of design products, i.e. artefacts, for the automating design process and representing design history. An integrated approach to modelling the design enterprise as a whole is presented.

(Tichern and Storm 1997) state that a design model which supports the application of computational tools should be capable of handling both, product models and life-phase system models. A model of the design process, which serves as a basis for developing life cycle computational supporting tools e.g. DFx's, is proposed.

Works dealing with models to capture the design process or design rationale, as either general or focused on specific design phases, can be found also in (Dowlatshahi 1994; Dowlatshahi(1) 1994; Lahti, Mantyla et al. 1997; Gao, Zeid et al. 1998).

This research is not focused on the modelling of the design process, but how information can be modelled to support design reuse activities.

2.4.2. Redesign

Three main categories/types of design problem have been identified by the literature, depending on the availability of the information and knowledge during the design process (Al Hamando and Kumura 1994; Evbuomwan, Sivaloganathan et al. 1996):

- Original/Non-Routine design: characterised by ill-defined goals, development of new concepts and lack of knowledge. This kind of design can be further divided in innovative and creative;

- Redesign: expanding the boundaries of the existing design principles to adapt functions and other changes of the new product; knowledge about decomposition is available, however modifications makes it necessary to acquire new knowledge. It can be further divided in adaptive and variant design;
Chapter 2

- Routine design: there is sufficient knowledge about function structure and goal structure. Compiled plans are available for goals and subgoals. The modification is restricted to some features;

Among the categories presented above, redesign has attracted significant interest because of the advantages in allowing the reusability of the knowledge and information from past experience and previous products (Fowler 1996). Additionally, this approach allows companies to have a maximum payback for each new development through the production of different products (variation) based on the same design concept.

(Fowler 1996) defines variant design as “a technique supporting retrieval of an existing design specification for the propose of adapting that design specification for use in the design of a new, but similar artefact”. (Evbuomwan, Sivaloganathan et al. 1996) states that variant design is a kind of redesign where based on a proven design as a basis for generating further geometrically similar designs of differing capacities.

(Fowler 1996) also states that during the redesign process the designer:

- Can seek inspiration for solution from existing design solutions;
- May have a conceptual solution in mind but seek for design solutions that are conceptually similar, or
- Can have an overall idea of the structure and organisation of satisfying artefact but can finish faster given a previously designed solution.

The above activities raise issues related to storing, manipulation and retrieving of information and past experiences to support design, which computational tools can be applied quite successfully. (Finger 1998) identifies these issues as representing, capturing, organising and retrieving the design knowledge, and addresses that there is a need for knowledge and information representation of an artefact, in order to develop computational environments to support design reuse.

CAD systems provide extensive support in the detailed phases of design, through features-based modelling and parametric design, however actual redesign is not supported since no reuse of information is performed (Fowler 1996; Finger 1998). Therefore, applications of computational tools to support design reuse of information
have gained significant attention by industry and research community in the last years.

2.4.3. Design Reuse

2.4.3.1. Design Reuse Concepts

(Sivaloganathan and Shahin 1999) state that design reuse is aimed at maximising the use of engineering creativity and expertise in design, by reuse of successful past experience in part or in whole for new designs. The authors discuss the current research carried out in design reuse based on a classification of different categories: focused innovation, cognitive studies on design reuse, computational perspective of design reuse, use of standard components, design reuse tools and methods, design reuse systems, and issues in design reuse.

(Fothergill, Arana et al. 1996) defines design reuse as “the design of a specific article to satisfy a customer specification in the context of an existing history of designs of similar articles forming a product family. The assumption is that the overall functional requirements in the product family remain the same, and that re-design does not involve the synthesis of new ways of solving design problems. It does involve re-use and adaptations of existing solutions and parts, and, possibly, new combinations of sub-solutions”.

In spite of all attractions offered by reuse, (Ormerod, Mariani et al. 1997; Busby 1999) address that some problems can be identified in design practice, such as problems with encoding reuse information, problems in situating reuse within the design process, and problems in retrieving reuse information.

(Finger 1998) addresses that functions, behaviour, form or even context can be used to retrieve a prior design, however there is no formal representation available for these attributes.

(Shahin, Andrews et al. 1999) state that some relevant pieces of information, defined by the authors as data models, to design reuse are: list of prioritised requirements, function list, annotated function and means, parts tree and feature-based model of parts and assemblies. These data models are based on the structure of the DFD (design function deployment).
However, when aiming the computational application of design reuse into CE environments, other issues must also be considered such as the computational approach applied for the retrieval process, representation of design process and representation of the more appropriate information structure.

2.4.3.2. Design Reuse Computational Approaches

The applications of computer based system to support design reuse has increased in the last years, as can be seen in the Proceedings of the Engineering Design Conference'98 (Sivaloganathan and Shahin 1998).

(Duffy, Smith et al. 1998) identified research work in computer based systems focused on supporting design reuse and classified them within three main computational approaches, named: (I) indexing and information retrieval; (II) knowledge utilisation, which is further divided in case based reasoning, model based reasoning, plan reuse and customised viewpoints, and (III) exploration and adaptation. A comprehensive comparison of the systems reviewed against an existing design reuse model process is presented and discussed.

Bellow is presented some computer based systems, identified in the literature, that support design reuse:

- DEKLARE (Design Knowledge Acquisition and Re-Design Environment), where a framework for supporting adaptive/variant design of mechanical artefacts is defined. Functional, physical and process design models are used through constraint management (Fothergill, Forster et al. 1995; Fothergill, Arana et al. 1996). However, DEKLARE characterises as a process based approach driven by a mapping of constraint definition, and is applied to more simple and defined product structure;

- RODEO (Reuse of Design Objects), where based on a given requirement specification, suitable redesigned objects are retrieved from a design database. Thus the current design problem is solved by instantiation or by (small) adaptations. A feature-based model describes design objects and requirement specifications by their properties (Altmeyer, Schuermann et al. 1994; Altmeyer, Schuermann et al. 1994) However, the reuse of designed objects is focused on a CAD framework;
Chapter 2

- DESPERATO (DESgn Process Encoding & Retrieval by Agent Designated Operations), where reuse in highly innovative and creative design task is investigated through the use of a computer-based indexing system. Intelligent agents that conduct encoding and retrieval operations on the user's behalf, have an interface with an object oriented database (Ormerod, Mariani et al. 1997);

- DEDAL, where an intelligent guide for browsing multimedia design documents supports reuse of engineering experience. A model for retrieval and indexing of multimedia design information is used (Baya, Gevins et al. 1992). However, the reuse in DEDAL is applied to management aspects of the design documents rather than the product design itself;

- NODES (Numerical and Object based modelling system for conceptual engineering DESign), where knowledge of design solution objects and their numerical relations are modelled. NODES is an interactive modelling system, which supports to build, manipulate and analyse a model of the design artefact and provide information feedback on the model. The knowledge is obtained by accumulating solutions of problems defined within that domain (Duffy, Persidis et al. 1996).

(Fowler 1996) addresses that two main research approaches have been developed to support redesign, named analogical reasoning and cased based reasoning. While the former is associated with the application of Knowledge-Based Systems (KBS), the later is associated with Case-Based Reasoning systems.

2.4.3.2.1. Case-Based Reasoning

Case-Based Reasoning (CBR) is a general paradigm to Artificial Intelligence (AI) problem solving based on the recall and reuse of specific experiences (Maher and Gomez de Silva Garza 1997). The CBR systems provide new solutions by analogy of past design situations, based on an adaptation of the previous selected solutions.

The main argument underlying CBR systems, is that human problem solving does not always involve reasoning from first principle, but may alternatively be a matter of relating information about a problem to past experience of solving problems (Lees 1997). This argument provides some implementation advantages of CBR systems in relation to KBS, such as not requiring an explicit domain model and identifying only
significant features that describe a case (Watson and Marir 1994; Maher and Pu 1997). Also, CBR uses actual past experiences to learn and solve new problems, rather than generalised heuristics, as in knowledge-based systems/expert systems.

Figure 2-3 describes the four main processes of CBR cycle. (Aamodt and Plaza 1994) define these processes into a framework, as Retrieve, Reuse, Revise and Retain. Thus, the process starts when a CBR system retrieves a suitable case from a case library by matching indexes established for the new problem. The information and knowledge found in the case retrieved is then, reused to provide an initial solution to the problem. Unless the solution fully satisfies the problem, the retrieved case will be revised, based on domain rules, heuristics or human intervention, producing a new case that can be retained, if considered as a valuable solution.

Figure 2-3 - CBR process

In order to deal with the CBR approach (Maher and Gomez de Silva Garza 1997) address two main considerations, named, representation of cases, and process models for recalling and adapting design cases. While representation of cases is responsible for storing the cases in a form that the "reasoner" (computer or human) can manipulate, the process models are responsible by finding a relevant design experience, recognising differences between the selected case and the new design problem, and changing the select case to the new design problem.
A review of researches and the commercially available CBR software tools, applied to
design, can be found in (Watson and Marir 1994; Maher and Gomez de Silva Garza
1997; Maher and Pu 1997; Watson and Perera 1997).

However, building a real CBR system within more complex design areas can raise
some limitations such as, the availability of cases; the expressiveness of the cases
representation language; the complexity of the retrieval algorithm and the knowledge
needed for the adaptation process (Maurer 1996).

Also, the application of CBR can be found in several areas, but few systems are
placed in the mechanical design domain, more specifically CADET and KRITIK
(Fowler 1996; Maher and Gomez de Silva Garza 1997). The problem is that
mechanical design is a complex area and the representation and use of cases requires a
significant assistance of generalised knowledge and heuristics (NedeB and Jacob
1997; Gao, Zeid et al. 1998). This means that Knowledge-Based Systems can still be a
better approach and alternative for design problem solving in this area (Chapter 4;
section 4.2.2).

It is not the main goal of this research work to assess points in adopting the CBR
approach, but the state that it is a well-known and important alternative approach.

2.4.3.2.2. Knowledge-Based System

While CBR focus on how to select, represent and organise actual cases, Knowledge-
Based Systems (KBS) are focused on how to capture, represent and apply
comprehensive knowledge models (analogies) to solve new design situations.

(Santhanam and Elam 1998) define KBS as a system which utilises AI methods and
store knowledge of a specific problem or technique to provide decision support.

(Blount, Kneebone et al. 1995) define KBS as computer systems which are
programmed to include an internal representation of know-how about particular tasks.
This know-how is used to solve problems, give advice and draw inferences.

(Dixon 1995) provides a more general definition, where KBS is a special class of
computer programs that claim to perform, or assist humans in performing, specific
intellectual tasks. The author makes a differentiation of the KBS from other kind of
programs, by the use of explicit knowledge (Figure 2-4). This provides certain
flexibility in terms of changing the knowledge without entering or modifying the
computer code that expresses the problem-solving algorithm. Therefore, besides being more difficult and complex to develop than domain specific applications, KBS once developed are more general and can be applied to wider range of products.

![Figure 2-4 - General structure of a KBS (Dixon 1995)](image)

KBS is also defined as KBE (Knowledge Based Engineering systems) when applied to the engineering area. (Anderson 1994) discusses the application of KBE systems in concurrent engineering environments, focusing on its relation with product models. (Chapman and Pinfold 1999) define KBE as an engineering method that represents a merging of object-oriented programming, artificial intelligence techniques and computer aided design technologies, giving benefits to customised or variant design automation solutions. The ultimate goal of the KBE system should be to capture the best design practices and engineering expertise, i.e. product and process model representation, into a corporate knowledge base. The authors report the use of the KBE tool applied to structural body description in automotive design.

The application of KBS requires the capture, representation and validation of such knowledge (Kiritsis 1995) (Caillaud and Noyes 1996). Even though different commercial methodologies for KBS development, e.g. KADS, STAGES, GEMINI, are available, they do not provide enough detail to be applied into product design process (Blount, Kneebone et al. 1995). Blount et al. address also four main types of knowledge used within the whole design process, named: entity attribute, topological connection, function entity and manufacturing attribute. The mapping between such models identify key areas in which knowledge engineering might be concentrated in the first stage of system development.
Chapter 2

(Sainter, Oldham et al. 1998) state that there is a need for a standard product knowledge representation language to provide the ability to exchange product knowledge as well as product data. This enables reuse and sharing of product knowledge between different knowledge based systems.

(Deneux and Wang 2000) state that in addition to a product model and a design process model, computer assisted design requires also a knowledge model, which is a necessary link between product and process models. Three types of classical approaches for knowledge modelling in design are addressed, namely, symbolic representation, e.g. IF-THEN; connectionist representation, e.g. neural; and representation of imprecise knowledge, e.g. fuzzy. The authors propose a hybrid model to represent expert knowledge and to support the decision making process in redesign, based on a fuzzy representation of design constraints.

(Harrington and Soltan 1995) suggest that one of the critical issues about KBS is when the knowledge starts to become wider, larger and more complex, for example within a Concurrent Engineering environment, where different expertise systems are placed together. The development of KBS as separate entities has additional advantages besides reducing the complexity of development and maintenance. However to cope with this approach, conflicts negotiation method or strategies must take place (Popplewell and Harding 1996).

The research performed in this work has been based on a KBS approach, however the modelling of the knowledge has been defined within an information based approach. The reuse of these pieces of knowledge has been defined as an important issue, as well as the simplicity of the model created to represent this knowledge. This issues are highlighted in the contribution of this research in Chapter 4.

2.4.4. Product Architecture

A product can be seen in both functional and physical terms. The functional elements of a product are the individual operations and transformations that contribute to the overall performance of the product. The physical elements are parts, components and subassemblies that can implement the product functions. This visualisation is achieved through the definition of a product architecture, which is a fundamental requirement to provide the necessary functionality and variance in a product family.
(Ulrich and Eppinger 1995) define product architecture as a scheme by which the functional elements of a product are arranged into chunks and by which these chunks interact with each other. This architecture can be modular or integral, depending the way that the product has been defined. In the modular architecture the relationships between chunks are well-defined and each chunk implements one or few functions. Products with this architecture usually correspond to the aspect of variety. The integral architecture corresponds more to the aspect of performance and usually the one chunk can implement several functions. The interactions between parts of product are not well defined.

(McKay, Erens et al. 1996) presents some definitions related to the product variety, where the product family identifies the commonality and differences between the individual products that form a product range. A variant of a product family is an individual product that conforms to the product family, since it has all features that are common to the family, and parameters. Finally, product range is defined as a set of variants of a single product family. In their work, (McKay, Erens et al. 1996) use of product modelling techniques to describe families of products without redundant data.

To avoid misinterpretation of the concept developed in this research it is appropriate to highlight the difference between the product range definition provided by (McKay, Erens et al. 1996), and the Product Range Model definition in this research. The Product Range Model is a separated information model from the product model, which stores the range of ways of designing a product range, based on its functionality, rather than a product model that stores a set of variants of a single product family. The limitations of the product model information structures to support design reuse are discussed in Chapter 4.

(Erens and Verhulst 1997) define product family as a product with identical/standard interfaces, i.e. interfaces between the product's components, for all variants in each product architecture domain. Three product architecture domains, namely functional, technological realisation and physical realisation, are defined.

(De Lit, Danloy et al. 2000) define product family as a group of products based on a specific design concept or derived from a standard/parent product. The authors present a product structure model for product families, which is able to deal with partial information of the product at early design stages.
(Tseng and Jiao 1997) present a variant approach to support product definition, where new products are evolved from existing products. Four domains are defined: customer, functional, physical and process. An approach based on recognising functional requirement patterns from past design efforts and developing product specification for a new design is proposed.

The importance of functions in defining a product architecture has been also recently recognised for major standards such as STEP (Mannisto, Peltonen et al. 1998; Mohrmann 1999), where the concept of Product Class is used to represent products with similar characteristics.

(Erens and Verhulst 1997) address how product's architecture support the development of a product family. Product's architecture is defined as the composition of a product from a number of component products, where this architecture is responsible for describing the components, together with their interfaces and operators. The modularity and integration, based on product variety and performance, respectively, are addressed as main aspects in defining the product architecture. The product's architecture is defined in three domains, defining the required function, technological realisation and physical realisation, and their relationships. The relationships between functions (functional domain) and solution principles (technological model) are addressed through four main steps: (I) decomposition, (II) allocation, (III) composition and (IV) validation.

(Allen and Carlson-Skalak 1998) present a method for defining product’s architecture, which is composed by three main steps, namely, identify product modules, identify function structure and finally identify system function structure.
The use of product architecture defines an important line of research, which makes also use of the product modelling principles for its applications.

Besides the advantages provided in the definition of a product architecture, the information structures, discussed above, are limited in terms of supporting design reuse (Costa and Young). This is because they can only support variants of existing designs. However, the understanding and definition of product architecture, more specifically related to the associations between functions and physical solutions, is an important issue in the definition of the Product Range Model information structure.

2.4.5. Functional Design

Functional design is a fundamental concept to the design process, which allows the representation of the product in terms of functional intents, rather than purely geometry. (Chittaro and Kumar 1998) identify different approaches to represent function and how functional reasoning can be applied to engineering.

Functional representation has been recognised as an essential aspect for developing future "intelligent" computer aided conceptual design tools (Al Hamando and Kumura 1994; Beng, Britton et al. 1998). This functional representation provides computer tools with the link between design functions and structure (physical) embodiments used to realise the functions. Such link can offer some benefits such as, support to the initial stages of the design (Chakrabarti and Tang 1996), store and capture of design rationale (Chandrasekaran, Goel et al. 1993; Tseng and Jiao 1997) or support product structure and design management (Tichkiewitch 1996; Gorti, Gupta et al. 1998; Snooke and Price 1998).

Two basic approaches are used to enable the transformation from functional requirement to geometry structures within a design problem (Al Hamando and Kumura 1994):

- Top-down approach, which starts with functions and look at the resulting shape ($f \Rightarrow \text{requires} \Rightarrow S$), and
- Bottom-up approach, which starts with known shape elements and asks for underlying functions ($s \Rightarrow \text{provides} \Rightarrow F'$).
Due to the nature of the function, techniques are necessary to decompose complex functions into more simple functions that can be manageable as design tasks and implemented by a design solution, allowing parallel development of different subsystems of the same product. Two approaches are traditionally adopted for functional decomposition (Ringstad 1997). These are:

- **Function/means tree**: where functions and solutions are shown in a hierarchical structure. Solutions to previous functions give rise to new functions and so on. Functions are expressed by verb/noun pairs and means realise functions including solution principles;

![Function/means decomposition diagram](image)

**Figure 2-6 - Function/means decomposition**

- **Axiomatic approach**: where functions and solutions are mapped in terms of functional requirements (FR) and Design Solutions (DS) (Suh 1995);

![Mapping between functional requirements and design solutions](image)

**Figure 2-7 - Mapping between functional requirements and design solutions**
The importance of functions in the representation of the product architecture has been addressed in the section 2.4.4. As a result functions are included in the representations of the product model information (McKay, Erens et al. 1996; Mannisto, Peltonen et al. 1998).

(Henderson 1993) explores the relationships among functionality, features, and dimensions and tolerances within a product model. A framework is defined, where a product model is divided into a physical realm model and meta-physical model. The first one holds physically based models, e.g. geometry, topology, material specification, dimensions, tolerances and features. The second one represents meta-knowledge of physical design, such as the reasons for entity existence in physical model, e.g. needs, functions, physical principles, instances, relations, constraints, design intentions. Into the last model, the concept of PDU (Product Definition Unit) is addressed, which holds attributes such as types, characteristics, relations and constraints with other PDU, and links with physical entities. The main link between both models is defined through the features.

(Baxter, Juster et al. 1994) address some limitations related to traditional product functional modelling approaches in computational applications to support the redesign process. A data model, which allows the representation of the functions of each feature, component and assembly in the product, is presented. Both, functional and structural hierarchies are represented in the functional data model.

Design reuse is where functions gain great power in that, associated with product constrains and other requirements, they can be used to identify more precisely potential design solutions (Hashemian and Gu 1997). This can be achieved by the application of different computational/mathematical approaches, such as constraints management (Forster, Fothergill et al. 1997), fuzzy representation of design constraints (Deneux and Wang 2000), constraint-based approach (Bowen 1997; Chung, Hwang et al. 2000) or graph-theoretic methods (Feng, Huang et al. 1996; Beng, Britton et al. 1998). However, these approaches are usually focused on the design process rather than information based.

This research work does not intent to explore the concepts behind functional theory, which can be a quite extensive subject, but state its importance in providing guidance in design reuse within the Product Range Model.
2.5. Injection Moulding Software Tools

2.5.1. Injection Mould Design

Four main activities can be identified during the cycle of injection moulded parts development, which are plastic component design, injection mould design, manufacture of the injection mould and manufacture of the plastic component (injection moulding process). Each activity has particular importance during this cycle, however because of its cost, time limitation and complexity, the injection mould is considered as a vital element, receiving particular attention on its design process. This research has defined injection mould design as its area of application.

Injection mould can be defined as an arrangement, in one assembly, of one or more number of hollow cavity spaces (impressions) built to the shape of the desired product, with the purpose of producing, usually, a large number of plastic parts or products (Rees 1995). The injection mould is mounted in an injection machine, which perform the following process steps: close the mould, inject the plastic into the cavity spaces, cool the mould, open the mould and eject the plastic component (Malloy 1994; Strong 1996). Such steps represent some of the main functions that must be coped with during the injection mould design (Catic and Raos 1989; Menges and Mohren 1993; Sebastian 1993).

Representations of the conventional design flow process for injection mould can be found in (Menges and Mohren 1993; Belofsky 1995; Rees 1995), highlighting how interactive and complex this process can be. These interactions can be related to external and internal aspects of the injection mould design. Some of the external aspects are requirements of the customer, injected plastic component, injection moulding machine, capabilities of machining processes. Some of the internal aspects are number of cavities, type of mould or injection mould systems definition (Figure 2-8). As a result, knowledge, experience and judgement in injection mould design are usually built only by years of experience, making difficult to establish a "defined" design methodology in this area, and therefore resulting in the lack of experienced people.
These issues associated with the availability of new tools (software and hardware) has put injection mould design as a challenging area for exploration of the concepts of software support systems based on the concept of concurrent design. However, because of the diversity of information involved, researches conducted in this area have been focused on several different subjects.

The following sections provide a clarification on injection mould design support system researches, identifying some aspects of interest for this thesis.

2.5.2. Classification of Injection Mould Design Support Systems

Traditionally, two main kinds of approaches have been applied and investigated to support injection mould design (Al-Ashaab and Young 1995; Chin and Wong 1996):

- Mathematical simulation analysis approach, which represents the injection moulding mathematically and assists the design process by simulation analysis, and

- Artificial intelligence based approach or computer advise systems, which use knowledge-based and expert systems approach.

The first approach can be categorized broadly in three fields of work, material flow, solid mechanics and heat transfer problem solving. Several software packages are commercially available (MoldFlow, C-Flow, etc.), which usually can perform plastic flow analysis, cooling analysis, shrinkage analysis, warpage analysis and/or stress...
analysis (Yeung and Lau 1997). A geometrical model, usually in a format of FEM mesh, is used to analyse and simulate based on different conditions initially defined by the user. This allows detection and solution of some problems, before the effective manufacture of the injection mould.

However, besides being a powerful approach, it requires the construction of a mathematical model for its application. This demands, besides a minimum knowledge about the finite element theory and analysis, definition of process conditions and certain assumptions, which usually are not available at the early design stage. For example, the user needs to know where and how to locate the gates, cooling, etc., in order to avoid the creation of a false model (Önalir, Kaftanoglu et al. 1997). Also, these tools appear as independent and specific applications within an injection mould design environment focusing on the design analysis rather than integrated design decision support.

Artificial Intelligence based approaches, on the other hand, focus on the capture of information and knowledge used during the design process. In this approach heuristics can be used to create rules to support the designers to compose their design. One of the great advantages of this approach is the fact of the designer does not need to be an expertise. The system can help him/her in the basic ideas in the very beginning of the design process. However, this approach demands formal procedures to capture, store and share this information in the right time for the designer. This is one of the main challengers of researches in this area. Computational techniques such as knowledge-based systems, expert systems, rule-based expert systems, case-based reasoning systems, information modeling are usually applied in this type of approach.

The research presented in this thesis is focused on this second type of approach, more specifically related to the modelling of information to support design reuse in injection mould design.

2.5.3. A Framework for Injection Mould Design Research Activities

A framework classifying works developed around injection mould design, into artificial intelligence based approach, is shown in Figure 2-9. This framework tries to make a differentiation among main researches conducted in this area, where three main kind of research fields can be identified, namely specific support systems, integrated support systems and information based support systems.
2.5.4. Specific Support Systems

Specific support systems represent works that support specific phases of the injection mould design process. Basically works in this area can be classified as supporting initial design decisions and main injection mould systems design.

2.5.4.1. Initial Design Decisions Support

Initial design decisions are characterised as the first phase in injection mould design process. Cost estimation of mould (Raviwongse and Allada 1997) (Chin and Pun 1994; Chin(1) and Wong 1996), best layout of mould impressions (Hu, Thevalingam et al. 1998) and definition of the optimal parting lines (Weinstein and Manoochehri 1996; Hui 1997; Serrar and Gabriele 1997; Weinstein and Manoochehri 1997; Zhang, Lee et al. 1997) are some of the main issues in this area.

The evaluation of the mouldability of plastic components is also one of the main inputs to injection mould design, however this issue is usually considered in the area of plastic component design (Dighe, Jakiela et al. 1993). Plastic component information i.e. geometry, material, dimensions, quantity, etc., is one of the main
specification of injection mould design. Initiatives exploring the design of plastic components for injection moulding can be found in (Pratt, Sivakemar et al. 1993; Zhang, Nee et al. 1994; Borg and MacCallum 1995; Wood and Ulman 1996).

2.5.4.2. Main Injection Mould Systems Design Support

The design of specific systems of the injection mould can be considered as the second phase in the design process. The injection mould systems are related to the main phases of the injection moulding process and the main functions of the injection mould, such as feeding, cooling, and ejecting the component from the mould (Mok, Chin et al. 1994). The application of AI-based systems has been identified mainly in the area of feed systems.

(Wang, Lee et al. 1996) propose an algorithm for optimising the ejector system design in plastic injection moulds. A CAD system based on heuristic knowledge and analysis to support the ejector system design is proposed. A simplified heuristic and quantitative analysis for determining the ejector force and size is suggested where the balance among the ejector forces is a critical factor. Basically, ejector pin, sleeve and plate are used as major ejection techniques. Geometric information is extracted from 3D moulding part including type of moulding feature, surface pattern and edge style.

(Ong, Prombanpong et al. 1995) presents a knowledge-based and object-oriented approach for the design of the feed system for plastic injection moulds, called CADFEED (Computer Aided Design of Feeding System). This system allows the calculation of type, location and size of gating system. Different criteria such as, aesthetic, symmetry, geometry, mould configuration, material are used to determinate the potential candidates for gates, which can be managed by the designer.

(Irani, Kim et al. 1995) presents a system named AMSD (Automated Mold Design System) to support the design of gate and runner systems. A CAE tool is integrated with an iterative redesign (new design analysed, evaluated and redesigned, if necessary) and knowledge stored in a features representation of the part. Eighteen performance parameters are used to evaluate the gate, while four are used to the runner.

Such approaches can provide significant support to injection mould design. However, they are focused on specific systems of injection mould, and hence do not take into
consideration more general aspects of the design, such as the evaluation of different decisions about other injection mould systems and their interactions.

2.5.5. Integrated Support Systems

The integrated support systems propose more comprehensive and general environments/architectures for supporting injection moulding design, where philosophies such as Concurrent Engineering take a significant important as guidance basis. The Integrated Molding System (IMS), developed at CIMP (Cornell Injection Molding Program), can be considered as an example of such systems (Wang 1997), where a suggestion of implementation of the CE concept within the injection moulding area is presented. However, it is mainly concerned with the application of simulation analysis tools approach.

(Lee, Li et al. 1997) present a knowledge-based injection mould design system - IMOLD (Figure 2-10). Data involved during the design process is divided in non-geometrical, (transferred by database) and geometrical (transferred by Parasolid neutral file). The authors state that there are two main kinds of parts in one set of mould. The first one is product dependent, where shape and size are decided directly by the product. They are designed by interactive design, such as core and cavities, inserts and sliders heads. The second one is standard parts, where the part shape is the same for any product, only size is product dependent. They are retrieved from a 3D (three-dimensional) standard part library and determinate by inference of knowledge base. Design knowledge is represented by an object-oriented method and stored in a form of product model. The structure of a product model used is presented and is composed basically by object interface, attributes, rules, method and relationship. The aspect of object relationships is mainly addressed between geometry and the knowledge stored in each object is significantly rigid, once it is not shown how changes or improvements on such knowledge can be made.
(Kruth and Willems 1994) present a prototype intelligent support system for the design of injection moulds, developed in the PROMISES project, and composed by three main tools integrated, CAD/CAM system, expert system and a relational database system (Figure 2-11). The system has the following modules, product definition, cavity layout, slide design, inject design, cooling design, ejection design, mould type, plate selection, ancillary components and cost calculations.

The geometrical information from CAD representation and technological information are stored in an object-oriented, feature based mould model. There are three kinds of classes of objects namely, (sub) assemblies, components and features (Willems, Lecluse et al. 1996).

One of the key points about this work is how each module has access to a centralised model, called Mould Model that is responsible for managing all decisions and information made during the design process, such as geometrical, technological and functional information. Thus, the Mould Model takes each decision made into consideration for further phases of the design process.
Figure 2-11 - Integration of CAD/CAM, database and expert system tools (based on (Kruth and Willems 1994))

(Chin and Wong 1995; Chin and Wong 1996; Chin and Wong 1999) present a knowledge-based system named, EIMPPLAN (Expert Injection Molding Part Planning System). Four main modules compose the system, named: Expert Plastic Material Selection Module (ESMATL); Expert Mold Design Module (ESMOLD); Expert Mold-Making Process Planning Module and Expert Molding Production Planning Module (Figure 2-12). The first two modules, part of the EIMPPLAN-1, are responsible for determining the most appropriate material based on input provided, and determining the major injection mould design features according to the part design requirements and plastic material characteristics, respectively. The other two modules are responsible for generating the mould-fabrication process plan, the moulding part-production plans, and the time and cost estimations.

In the ESMOLD, feature-based design is used as a way to connect product design to mould design. Also, libraries of part design feature primitives, such as parting line, mould cavity geometry, mould core geometry and undercut classification, assist the user in defining the part design features. Finally, the result, major injection mould features, are classified into cavity number and layout, mould construction type, undercut release mechanism, ejector system, type of gate, cooling system, cavity/core finish and mould insert material.

The work highlights how the plastic component interacts with the injection mould design through feature-based design.
Chapter 2

Figure 2-12 - The EIMPPLANAM Framework (based on (Chin and Wong 1996))

(Lee, Chen et al. 1997) focus the concurrence of the injection mould design process, where the relationships and interactions among the injection mould activities and tasks analysis is the main focus. A system framework involving user interface, knowledge-based mould facility (pre-moulding process, moulding layout, feed system design, cooling system design and venting design), supporting facility (i.e. knowledge bases, databases and libraries) and supporting tools (software analysis and CAD systems) is proposed (Figure 2-13). At the supporting facility, the knowledge bases include modules for mouldability assessment, undercut detection, moulding features design, and detail moulding design. The databases store information about moulding materials and moulding machine capabilities. The libraries hold information on mould base, moulding features and cooling features. To support the construction of the system a mould model, a knowledge model and a data model were developed.

One key point addressed by their work is the need for coping with interactions between different systems of the mould. However, no solution for this issue is proposed.

(Mok, Chin et al. 1994) presents a system called KBMOLD, where the functional design viewpoint of injection moulds is addressed. Four main functional systems are considered in the injection mould design, named: feed system, eject system, cooling system and mould construction. Plastic part information and mould specifications are the main input of the system that generates a code. This code is used, in turn, to search
for previous designs in a search database, and to select main methods of achieving each function in an analysis by function. These methods will allow the designer to choose hardware alternatives, which will be stored in a Knowledge Base with the final dimensions.

![Figure 2-13 - System framework for concurrent mold development (based on (Lee, Chen et al. 1997))](image)

(NedeB and Jacob 1997) present a system that aids the designer to choose working principles for injection mould functions, based on prior experiences with them. The system is composed of a decision support system (DSS) for the designing engineer and a system for fault diagnosis on the shop floor. Quality control loops connecting design and shop floor supports fault detection related to the injection mould process on the shop floor (Figure 2-14).

Both works presented in (NedeB and Jacob 1997) and (Mok, Chin et al. 1994) reuse design information based on actual past experience, i.e. based on a Case-Based Reasoning approach. However the interactions between different design decisions are still an issue, which is pointed out but not solved. Chapter 4 highlights this as one of the motivation issues where this research makes contribution.
2.5.6. **Information Based Support System**

Information based support system researches are characterised as works that have the main goal to provide information structures to support the injection mould design phases. Two main kind of information structures can be identified in this respect, product information and manufacturing information.

Most of researches presented in sections 2.5.4 and 2.5.5 address and make use of information models, more specifically product models. However, usually such models do not represent the main focus of the research, being defined in a restricted context.

(Shaharoun, Razak *et al.* 1997) address the application of a product model to support product description, however it is applied to plastic product description. (Al-Ashaab 1994; Al-Ashaab and Young 1995) address how the capabilities of the injection moulding process can be represented in a manufacturing model to support injection mould design. This manufacturing model captures information about mouldability features, mould elements and injection moulding machine elements, and can be used in the design stage to support design for mouldability (plastic product), design of the mould and selection of injection machine. The plastic component is considered as the main product Figure 2-15. For representation, **EXPRESS** language and **EXPRESS-G** were used (Al-Ashaab and Young 1997).
Chapter 2

Figure 2-15 - Manufacturing model used by DFIM application (Al-Ashaab and Young 1995)

(Lee 1996; Lee and Young 1998) presents a system to enable concurrent design for injection moulding components (Figure 2-16). Design for manufacture information supports the injection moulding design process concurrently based on the design for function process. The functions of product are analysed in terms of mouldabilities that in turn are analysed in terms of mould design aspects (core and cavity and mould elements systems), allowing a feedback to designer. Translation mechanisms to deal with different information have been addressed. The concept of Product Range Model is proposed to capture functional and manufacturing information about plastic components, however its nature has been not explored. Chapter 4 highlights Lee’s work identifying points in which the research presented in this thesis makes further advances.

Figure 2-16 - Relation between functions and mouldability features (Lee 1996)
Chapter 2

(Webb, Gerdes et al. 1995) describe a database system that minimises data communality and software incompatibility problems in integrated injection moulding design and manufacturing environment. The system provides a common storage for all relevant information of the shop (plant) and allows applications use and shares this information. Four different kinds of database are identified: shop resources (machine, tool and human), part description library (feature templates, feature rules and material database), moldbase description library (component templates and assembly templates), and manufacturing procedures library (feature-manufacture procedures and procedure-reduction rules).

The use of injection mould standard components, as information to support the design process has also been investigated by some of the researches addressed in section 2.5.5.

(Willems, Lecluse et al. 1996; Kruth, Willems et al. 1997) state the importance of associating functionality to the injection mould components, allowing these components to become available for CAD systems as high level objects. However, such association is applied to standard components, rather than general design solutions.

2.6. Summary

This chapter has provided a survey on the three main areas involved with this work.

Information models are an important element in CAE systems to provide a reliable source of information to support the product life cycle activities. The representation of information models is achieved by the consistent definition of an information structure, i.e. information data model, which must be taken into consideration for the development of data model driven applications. Although the concept of the information model is usually associated with the product model, additional information model can be defined to support life cycle activities, e.g. manufacturing model.

The reuse of information in design requires in addition to the product information the knowledge associated with how this information can be used. The capture and representation of functional and physical models support both the definition of product architecture in the case of product families, as well as the reuse of previous
design solutions. While product architecture is mainly related to product model
information structure to support composition of design variants, design reuse systems
are focused on indexing and retrieving of previous design solutions.

Works on injection mould design support apply reuse mainly related to the heuristic
knowledge, where specific knowledge bases are created for each particular injection
mould system. While this approach can provide a better result for advanced phases of
design, it is also limited in terms of evaluating initial interactions between different
injection mould design decisions. Information models have been used in injection
mould design, however not for supporting design reuse.

These three main issues are critically reviewed in chapter 4 where the contribution of
the work in the context of the problem area that this research is highlighted.

3.1. Introduction

This chapter sets the research environment and identifies MOSES concepts that have been used as a basis for this research. A description of the tools and methodology used to evaluate the concepts of the research work are also provided, enabling a clear understanding of the ideas presented in the further chapters of this thesis.

3.2. Research Environment

3.2.1. General Description

Figure 3-1 summarises the environment used to support this research work and the development of an experimental information system, where:

- MOSES (Model Oriented Simultaneous Engineering System) system was adopted as CAE system architecture;
- RM-ODP (Reference Model for Open Distributed Processing) was applied as a background reference model to describe the different levels of this information system, and
- UML (Unified Modelling Language) as notation, i.e. description language, was applied to represent the different views of the information system.

![Figure 3-1 – General research environment](image-url)
To support the computational implementation in this research, the ObjectStore© database and Visual C++© programming environment were employed. While the ObjectStore© database was used to realise and represent the information structures, the Visual C++© was selected to realise the functionality of the experimental system developed. In addition of being the best accessible/available tools, they have been chosen due to their capabilities in integrated working within the object-oriented approach. The Rational Rose tool was used to support the diagrammatic representation of the information structures developed, i.e. static view, as well as the system functionality, i.e. dynamic view.

These tools are addressed in the following sub-sections.

3.2.2. MOSES system architecture

As addressed in Chapter 2, section 2.3.4.1, three main elements are identified in the MOSES system, namely information data models, integration environment, and software applications, or data model driven applications (Young,Canciglieri-Jnr et al. 1998). This research is focused mainly on the first element, e.g. information data models, to support design information reuse.

Two main information models have been traditionally investigated by researches around MOSES, named Product Model and Manufacturing Model (Ellis,Molina et al. 1995). Al-Ashaab (Al-Ashaab 1994) and Lee (Lee 1996) have investigated previously the concepts of both, Manufacturing Model and Product Model respectively, applied to the injection moulding area. While the Product Model concept has been used as a repository for storing and retrieving information about the injection moulded component, the Manufacturing Model has dealt with resources, processes and strategies of material removal and injection moulding processes. However, such information models have been used with no major considerations about supporting the injection mould design through the reuse of previous information and knowledge.

The introduction of the Product Range Model into MOSES architecture requires, in addition to defining its functionality, considerations about its relationships with other information models and software applications. This research has defined the product model and a design for function application as the main elements that will relate to the Product Range Model (Figure 3-2).
Chapter 3

Information Product 4- Level Model
Integration I
Environment I
Applications Design for Function Level

Thus, to guide the concepts involved with the introduction of the Product Range Model, the main functionality of the software applications that will share this information model must be captured. Such functionality will support the definition of the Product Range Model data representation, i.e. Product Range Data Model.

The next sections bring an explanation about the tools used to support the design and implementation process of the Product Range Model concept, where an object oriented software application named IMSS (Injection Mould Support System) has been developed (Chapter 9).

3.2.3. RM-ODP (Reference Model for Open Distributed Process)

RM-ODP (Reference Model of Open Distributed Processing) was created to produce a reference model for describing open distributed systems and it is now accepted as a de facto standard (Blair, Coulson et al. 1996). It is divided into five viewpoints (Figure 3-3), which are described in detail in ISO/IEC 10746-1, and summarised as follow:

I. Enterprise viewpoint: describes the information system in terms of what it is required to do. This section of the model captures the business and administrative requirements and policies that justify and orientate the design of the system;

II. Information viewpoint: describes the information system in terms of information structure, information flow and information manipulation constraints;
III. Computational viewpoint: describes the information system in terms of operation and computational characteristics of the process that change the information;

IV. Engineering viewpoint: describes the information system in terms of the engineering resources necessary to support the distributed nature of the processing, and

V. Technological viewpoint: describes the information system in terms of realised components from which it is built.

![Diagram of viewpoints]

Figure 3-3 - Viewpoints of RM-ODP

The RM-ODP provides a standardised way of designing and comparing an information system, which is the reason why it has been used in this research work. However, due to the nature of this research, the application of the RM-ODP has mainly focused on the Information Viewpoint, which is related to the information structure representation and description.

3.2.4. UML (Unified Modelling Language)

3.2.4.1. UML Diagrams

The notation provided by the UML (Unified Modelling Language) has been selected to support the representation of the Product Range Model information system. This choice was made because the range of diagrams available to represent both static and dynamic behaviour of the system, through the several viewpoints of system analysis, design and implementation phases.

UML (Unified Modelling Language) is a standard language to support the design and modelling of multiple perspectives of information systems. Recently, it has become recognised and accepted as a potential notation standard by the OMG (Object
Chapter 3

Management Group) (Booch, Rumbaugh et al. 1999). Such notation defines a set of basic diagrams that provide multiple perspectives (structural/static and behavioural/dynamic) of the system (object-oriented) under analysis or development, allowing a real world representation of the system in development. UML diagrams include Use Case Diagrams, Class Diagrams, Interaction (Sequence and Collaboration) Diagrams, Activities Diagrams, State and Transition Diagrams, Deployment Diagrams, etc.

Table 3-1 provides a brief explanation of the UML diagrams used in this research. A detailed explanation of these diagrams is provided in Appendix-A.

3.2.4.2. A process to support UML notation

In addition to the UML notation, a process that allows the migration through the different phases of system development (e.g. functionality, analysis, design, implementation, etc.) is necessary. To fulfil this requirement, the process proposed by (Texel and Williams 1997) has been applied in this research, where Use Cases drive the subsequent phases of development of an object oriented system. This approach has the advantage that the entire process of creating and defining the object is driven by specific pieces of system functionality becoming more focused and modular.

One difference in the use of this approach is related to the way of capturing the main system functionality, what has been done by traditionally modelling functions and activities through IDEF's diagrams (Colquhoun, Baines et al. 1993; Al-Ashaab 1994; Lee 1996; Kusiak, Letsche et al. 1997). In the author's opinion these tools have a significant potential when there is a need for capturing actual company's processes ("as-is"), which need some sort of re-evaluation ("should-be"). IDEF0 diagrams put a significant effort in initial stage of analysis and design activities, focusing on the final user activities rather than the system functionality. Also, when applying OOT (Object Oriented Technology), where the main issue is to identify and design objects correctly, IDEF0 diagrams tend to analyse objects by their location rather than by their functionality.

Therefore, for the analysis and design of the Product Range Model application, emphasis was put on capturing the software application functionality through the use of Use Cases. Appendix-A presents a description of the process applied and Appendix B shows an example of this process application in the experimental system context.
<table>
<thead>
<tr>
<th>Diagram</th>
<th>Description</th>
<th>Elements</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Representation</td>
<td>Represents a set of objects that share the same attributes, methods, relationships and semantics.</td>
<td>• Name</td>
<td><img src="#" alt="Class Name" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Attributes</td>
<td><img src="#" alt="Class Attributes" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Methods</td>
<td><img src="#" alt="Class Methods" /></td>
</tr>
<tr>
<td>Use Cases Diagrams</td>
<td>Represent the high level functionality of a system (what the system “should” do). Use Cases are extracted from the discussions among final users, system designers and managers.</td>
<td>• Use Cases</td>
<td><img src="#" alt="Start Design Process" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Actors</td>
<td><img src="#" alt="Select Existing Product" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Relationships (dependency, generalisation and association)</td>
<td><img src="#" alt="Designer" /></td>
</tr>
<tr>
<td>Class Diagrams</td>
<td>Represent the internal structure (attributes and methods) and relationships between a set of objects. The relationships define mainly the way that such objects will be implemented.</td>
<td>• Classes</td>
<td><img src="#" alt="Product" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Relationships (generalisation, association and aggregation)</td>
<td><img src="#" alt="Plastic Component" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><img src="#" alt="Injection Mould" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><img src="#" alt="Injection Mould" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><img src="#" alt="Injection Mould" /></td>
</tr>
</tbody>
</table>

Table 3-1 – UML diagram description
<table>
<thead>
<tr>
<th>Diagram</th>
<th>Description</th>
<th>Elements</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category/Packages</td>
<td>Represent the associations between main parts of the system (Categories/Packages). Categories represent a set of common objects with similar functionality, and category diagrams support the modular design and implementation of the system.</td>
<td>♦ Categories/ Packages  ♦ Relationships (dependency association)</td>
<td><img src="image1" alt="Category Diagram" />  <img src="image2" alt="Category Diagram" />  <img src="image3" alt="Category Diagram" />  <img src="image4" alt="Category Diagram" /></td>
</tr>
<tr>
<td>Sequence Diagrams</td>
<td>Capture and represent the collaboration required between classes, through their methods. Basically the behavioural aspects of objects are focused, showing which methods are required to satisfy a specific Use Case.</td>
<td>♦ Classes  ♦ Methods</td>
<td><img src="image5" alt="Sequence Diagram" />  <img src="image6" alt="Sequence Diagram" />  <img src="image7" alt="Sequence Diagram" />  <img src="image8" alt="Sequence Diagram" /></td>
</tr>
</tbody>
</table>

Table 3-1 – UML diagram description (Cont.)
<table>
<thead>
<tr>
<th>Diagram</th>
<th>Description</th>
<th>Elements</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Diagrams</td>
<td>Represent the computational steps to be performed by a specific method. The dynamic aspects of the system are focused. Each activity results in some action that, in turn results in changes in state of the system or return of a value.</td>
<td>♦ Activity flows</td>
<td><img src="image" alt="Activity Diagram" /></td>
</tr>
</tbody>
</table>
| State and Transition Diagrams   | Represent the internal behaviour of a class during its life, and show how events can change such class life cycle phases. | ♦ Class states  
♦ Methods                      | ![State and Transition Diagram](image)                                    |

Table 3-1 – UML diagram description (Cont.)
3.3. Computational Tools

3.3.1. ObjectStore Database

ObjectStore is a pure object-oriented database tool (Object Design 1998) and was used to implement the information model schemas of the Product Model and Product Range Model. Besides the ObjectStore database application itself, two other tools, named ObjectStore Database Designer and ObjectStore Inspector, have been used to support the computational implementation process.

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

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

A brief explanation of the process of designing, generating and inspecting the database, using the ObjectStore tools, is included in Appendix A.

3.3.2. Visual C++

The functionality of the software application has been realised by using Microsoft Visual C++®, as a computational programming environment. This tool allows the development and implementation of visual interfaces between the user and the information models stored in the database. The tool uses an object-oriented approach and is based on Windows MFC (Microsoft Foundation Classes) (Pappas and Murray III 1997).

3.3.3. Rational Rose 98 Enterprise Edition

Rational Rose is a tool that supports the object-oriented modelling of software systems. Besides Booch and OMT (Object Modelling Technique) notations, Rational Rose supports the modelling of UML diagrams through different views (static or dynamic) of the system in different levels of refinement, such as Use Case, Logical, Component, Process and Deployment views (Quatrani 1998). In this research this
software has been used to support the design and representation of the Product Range Model and product model information structure, and the dynamic behaviour of the software application developed.

### 3.4. Industrial Collaborator

In order to ensure that the research and exploration undertaken are relevant and realistic in terms of today’s manufacturing requirements, part of the information about injection mould design has been collected from an industrial collaborator, named Moss Plastic Parts Ltd. (Kidlington), part of Bunzl plc group. Among some of the typical plastic components produced by the company are: general protection caps and plugs, fittings for tubular products, packaging closures and containers, etc.. The design of injection moulds, as well their manufacture is realised at the plant, which has been visited.
Chapter 4

4. Product Range Model Concept For Injection Mould Design

4.1. Introduction

This chapter defines the problem area of this research and provides an outline of the subsequent chapters, which report on how these problems have been addressed. Two main sub-sections in this chapter identify issues related to computational support for injection mould design; the need for a new information model to support design information reuse, and the nature of such an information model, i.e. Product Range Model.

In addition to presenting the justification of this research, this chapter establishes the main research issues relating to development of the main structure of the Product Range Model, which are presented in Chapters 5, 6 and 7.

4.2. Computational support for injection mould design

4.2.1. General Issues

Significant advances have been made in the area of commercial CAD systems which support injection mould design. However such systems are mainly concerned with geometric modelling aspects of the product, and tend to be limited in providing support for decisions that go beyond geometric information. When dealing with information reuse such systems seldom offer any help except recover product information through some means of particular codification or specific identifier, e.g. ID number.

As previously discussed in the literature survey (Chapter2; section 2.5) computational support tools for injection mould design are based on mathematical simulation and AI based approaches.

From this survey, a number of AI related issues, which need to be addressed to improve computational support in injection mould design, have been identified. These are:
I. The need for a multidisciplinary approach involving information and decisions in upstream activities, e.g. plastic component design, and in downstream activities, e.g. mould manufacture and injection moulding process;

II. Different domains of expertise which can demand and result in different types of knowledge representation;

III. Highly empirical design process, requiring expertise and knowledge to be accumulated over the years before information can be effectively reused;

IV. The interactivity and dependence between different functions and consequently injection mould solutions, which achieve such functions must be captured;

V. The high complexity of the plastic component geometry associated with higher accuracy and shorter delivery time makes the definition of a set of rules very difficult.

In order to provide an integrated and complete solution to support injection mould design, the major problems listed above must be resolved. However, due to the recognised importance of information structures, as a core element supporting software applications in major concurrent engineering environments (Krause, Kimura et al. 1993), this thesis has focused on issues mainly related to subjects III and IV, i.e. design reuse and the interactivity between different injection mould functions based on information models.

Thus, a key element in this research is the consistent modelling of an information structure, which allows storage of design information for future reuse. Such a structure must fit into a major CAE system architecture in order to provide useful support for a concurrent engineering environment. Hence, in addition to the main information associated with the injection mould functions and their respective design solutions, details of its relationships with other design information, can be stored allowing an intelligent process of information reuse. This is in line with (Fowler 1995; Gorti, Gupta et al. 1998) in that data representation related to product and design process is an important issue which requires consideration in the development of future information systems. Thus, the provision of information support forms the basis of this research approach to facilitate reuse in injection mould design.
4.2.2. Information modelling to support injection mould design reuse

As previously described in section 2.3, Chapter 2, two main approaches have been applied to supporting design reuse, namely Cased-Based Reasoning (CBR) and Knowledge-Based Systems (KBS).

Even though CBR is well accepted as a feasible approach for supporting design reuse in many design areas (Watson and Perera 1997), injection mould design is characterised as a very interactive and complex process and deals with a large range of information. As result, cases can vary significantly from design to design, demanding additional knowledge for the adaptation process, as shown in (NedeB and Jacob 1997). Additionally, CBR has traditionally focused on retrieving/adapting previous cases rather than looking at the information structure required to support reuse in a concurrent engineering environment.

KBS approach, on the other hand, reduces the manipulation of information to specific situations/objects, and is therefore more compatible with the definition of information structures (Gorti, Gupta et al. 1998). It allows capture, and therefore gives a better representation of the injection mould design information and knowledge in terms of specific sets of elements, which the designer deals with at each design stage (Lee, Li et al. 1997). This feature, when looked at using an information modelling approach, can make better use of the power of object oriented technology, e.g. object oriented databases (Kung, Du et al. 1999). This research examined this approach and focused on information structures to capture design information and knowledge, which support design reuse.

From section 2.3, Chapter 2, the use of information models to support design and manufacturing activities, through specific computational applications, has been identified as a major element in future CAE systems (Jo, Parsaei et al. 1993; Krause, Kimura et al. 1993). Similarly, the reuse of information has attracted significant attention from the research community because of the advantages that it can provide to support design activities (Sivaloganathan and Shahin 1999). This thesis has explored the nature of the Product Range Model and shows how such an information model can provide good design support, particularly for information reuse in the cases of product ranges.
Also, from section 2.3, product model research has been focused on identifying better structures for representing information for a specific product in development (McKay, Bloor et al. 1996; Anderl 1997) and more recently on providing a representation for product ranges (McKay, Erens et al. 1996). The International Organization for Standardization (ISO) initiatives like AP 214 (Mohrmann 1999) are also starting to improve representations for product ranges. However such efforts have been focused on product functional/structural model representations to support design reuse. Although offering a more consistent source of information for each product, such information structures provide only a variant data model for a range of products and are, therefore, limited in terms of the support they provide for design reuse. The work in this thesis goes beyond such structures and suggests a separated information structure to more actively provide design support.

To provide design reuse support, an information model must accommodate general pieces of information that compose the knowledge acquired over a period of time for product ranges. This means that both ways of reusing information and ways in which such information is evaluated, before being offered to the designer, must be captured. Attention must be paid in the differentiation between the meaning of the terms data, information and knowledge (Harding 1996). While data is related simply to words and number, the meaning of which has not been defined; information is structured data that has some meaning; and knowledge is information with added value that relates to how it may be used or applied.

Section 2.4.5 has identified that product functions can be applied to drive the design process providing, intelligent information retrieval, or reuse, for high-level design support (Al Hamando and Kumura 1994; Beng, Britton et al. 1998). However, when applied with this meaning, the inclusion of functions in a product model can make the structure of such an information model relatively complex (Henderson 1993; Baxter, Juster et al. 1994), interfering with the main purpose of this information model.

To support this intelligent information retrieval, design decisions interactions should also be captured. However, once again, such an approach can make the product model structure more complex than necessary (Lei, Taura et al. 1996). Constraints management and propagation can be used to guide this process (Fothergill, Arana et al. 1996), which results in the approach becoming process driven, rather than
information driven, despite the additional support provided by necessary information structuring (Ullman 2000). Also, these approaches are usually focused on the constraints of the geometric aspects, which play a role in more detailed design stages (Chung, Hwang et al. 2000).

In injection mould design support, the above scenarios are still true, as reviewed in section 2.5 of Chapter 2. To support injection mould design reuse intelligently, there is a need to capture the relationships between product elements and their functionality (Sebastian 1993; Kruth, Willems et al. 1997). However, such a relationship is still general since only valid information should be provided to the designer. The particular attributes and reasons for the choice, or not, of each design solution should also be captured (Lee, Li et al. 1997). The problem of such an approach is that it is design process based rather than information based, and is focused on if-then rules created inside of the object methods, which becomes limited in terms of flexibility for changing or adding new rules. Furthermore, the need to consider interactions between different injection mould system solutions is an issue not properly addressed by most of researchers (Lee, Chen et al. 1997).

An approach to deal with interactions in injection mould design has been to create a set of rules inside an expert model (Kruth and Willems 1994). However, besides being based on a relational database, what can present some obstructions when applied to concurrent design, this approach can become very complex in terms of building the expert model, as more experience is added to the system.

With respect to the problems and limitations previously exposed, this thesis argues that the Product Range Model can provide good support for design decisions through the intelligent reuse of information.

4.2.3. The nature of a Product Range Model

(Lee 1996) addressed the concept of a Product Range Model as a means of providing a link between design for function and design for manufacturing applications (Chapter 2; section 2.5.6). His work is limited to the relationships between product functions and product geometry solutions. However, Lee did not explore the nature of the Product Range Model, or its components, pointing it out as an issue requiring further investigation. This was therefore defined as a critical point for this research.
In the Product Range Model defined by Lee, product ranges are composed by initial product definition data, which represent the sequential dependency between the product range functions; functional requirement data, which supports the evaluation of a form feature; and form-function relations data, which supports the relationships between a particular product range function and form features. However, Lee’s Product Range Model structure has significant limitations. These are: the relationships captured are limited to functions and form; the knowledge required to evaluate the application of each form feature is captured by "if-then" rules through hard code, which makes the Product Range Model structure rigid for further changes and limited in the reuse of this knowledge and information; the Product Range Model has not being tested within an integrated design environment as a separated information model from the Product Model. The work reported in this thesis provides an advance in these issues exploring other types of relationships rather than function-form, in addition to a more flexible information structure to represent the Product Range Model and to support design reuse.

Even though the Product Range Model maintains a strong dependency on the product model, the former has a different functionality to the latter. The product model aims to capture the most appropriate hierarchical information structure to represent the product and share such information with different life-cycle software applications. On the other hand, the Product Range Model aims to support the initial phases of design activity, where general decisions are taken, through the reuse of information, dealing therefore with more abstract levels of understanding design decisions, i.e. concepts of the solutions adopted in the injection mould design (Costa and Young ). Thus, the information inside the Product Range Model is based on the experience and knowledge acquired for a particular kind of product range.

In this respect, this research has proposes the hypothesis that design reuse can be supported by the computational representation of product ranges information and knowledge within an integrated design environment and that this computational representation should be structured in terms of functions, design solutions, interactions and knowledge links.

This research has defined the use of sharable information structures as approach for supporting this integration and, hence the work reported in this thesis has explored the nature of an information model, termed Product Range Model. The Product Range
Chapter 4

Model must provide to the designer, based on functional enquiry, sets of valid design solution options, which can be chosen and stored in the Product Model. Therefore, in addition to the definition of an information structure to represent this information model, its relationship with the Product Model must also considered.

This has been done by defining an information structure able to capture not only the relationships between functions and design solutions related to a particular product range, but also the knowledge related to the application of these design solutions in specific design situations.

To pursue issues related to the definition of the Product Range Model, the following questions have been addressed by this work:

♦ How to represent the relationships between functions and design solutions for an injection mould tool?

♦ How each design solution can capture the required knowledge of past experience to support injection mould design reuse in an information based approach?

♦ How information stored in each design solution can be checked against the content of other information models, i.e. product model?

These issues are briefly discussed in section 4.3 and will be answered in chapters 6, 7 and 8 of this thesis, respectively, where a detailed exposition of each topic is provided.

4.3. Product Range Models supporting information reuse in injection mould design

4.3.1. Injection Mould Product Range

The injection mould design characterises one of the main phases of the injection moulding cycle and during its design different kinds of interactions take place, such as interactions with the plastic component, the injection machine and the injection mould internal specifications (Chin and Wong 1996). The description of injection moulds as a product range, including its functions, design solutions, and interactions are addressed in Chapter 5.
Figure 4-1 depicts the Product Range Model within the context of MOSES architecture, focusing on its relationships with other information models and software applications. Injection moulds are highlighted as a kind of product range. The relationship between the product range functions and design solutions is pointed out as the main content of the Product Range Model.

However, in order to offer useful information, in addition to relationships between functions and design solutions, the Product Range Model has been expanded to also capture the reasons that make a design solution eligible, or not, for a specific design situation, represented in Figure 4-1 by *injection mould design criteria*. These criteria are related to the different design conditions that must be respected for the eligibility of each design solution and, in order to perform an assessment of such criteria, the interactions between the Product Range Model and product model information must be considered.
This research has identified three kinds of relationships within the Product Range Model, namely: (I) between injection mould functions and injection mould design solutions, (II) between injection mould design solutions and product information, and (III) between information models structures. In order to support these relationships and clarify the questions addressed by this research, the issues addressed in the following sections must be resolved in the Product Range Model.

4.3.2. Function and Design Solutions Relationships

This relationship allows an association between specific functions required from the injection mould and all possible solutions that could potentially be applied to achieve such functions to be made. For instance, to eject a plastic component a set of possible ejection techniques is available (e.g. pins, stripper, etc.). Each solution, in turn, can be applied to one or more function. Using the same example, an ejection pin, can be used to either eject the plastic product, the runner or eventually the gate.

The relationship between product range functions and design solutions is not complex in itself, even though a good understanding about the product structure is required for its clear definition. Such relationship have been well addressed in the form of functional and physical models (Baxter, Juster et al. 1994; Erens and Verhulst 1997). However, as previously mentioned, such relationships are usually applied from the perspective of a product architecture representation. They provide support in terms of checking the composition of product variants, based on basic specifications. This research agrees with the need for such representation, but in order to provide a more active way to support design reuse in product range cases, the relationship between functions and design solutions must be captured outside the product model.

Thus, both, functions and design solutions are considered part of the major structure of the Product Range Model, which, in turn, is a separate information model, providing therefore more flexibility and individuality for the information models.

The injection mould functions and design solutions information structure and relationship in the Product Range Model are discussed in chapter 6.
4.3.3. Design Solutions and Product Information Relationships

Even though the relationship between functions and design solutions is an important piece of information to guide the designer to potential design solutions, it is not enough to point out which of them can be applied in a specific design situation.

As pointed out in section 4.3.1, each design solution should be assessed against different design criteria to provide the designer with useful information, or valid design solutions options (Figure 4-2).

In this respect, two kinds of information, which interact with each design solution, have been identified for the injection mould design process. The first one is related with initial product specifications, such as the plastic component characteristics (geometry, material, annual rate, etc.), and injection mould basic definitions (type of mould, number of impressions, etc.). The second one is related with the choices made by the designer, to achieve specific functions of the injection mould, during the design process. In Figure 4-2 these choices are represented by “chosen design solutions”.

Whilst the first type of information relates to more rigid decisions, the second can be considered as more flexible decisions, since they may be changed to achieve a well balanced final design to fulfil the multiple functions required from the injection mould. Hence, for each function specified, and as a result of the check interaction process, a set of acceptable, and non-acceptable, design solutions can be offered to the designer, to assist his/her decision. As decisions about design solutions are taken, more information is considered for further design solution interactions.

In this research, interaction elements have been defined to capture the way that each design solution criteria is evaluated against the product information. Thus each interaction is considered as a particular element, which holds information enabling evaluation of whether a design solution is considered eligible or not under a specific design situation. The nature of these interactions is addressed in chapter 7.
Figure 4-2 – Product Range Model general relationships
4.3.4. Information Model Relationships

The Product Range Model has been defined, in this research, as an separated information model. Thus, to allow the assessment of the Product Range Model design solutions criteria, other aspects have to be considered regarding the relationship between both information models, e.g. Product Range Model and product model. In this respect, two additional elements need to be considered to enable the "Check Interactions" process (Figure 4-3).

The first element is related to the representation of the heuristic knowledge behind each design solution in relation to the product information, both product specifications and chosen design solutions. This element represents the knowledge and experience acquired though time by the design team.

The second element is related to the representation of the knowledge of product data structure, and in this research, this element has been termed as a Knowledge Link. In contrast to the first element, this element must have an "understanding" of other information model structures, and hence capture and represent the knowledge behind the product hierarchical tree, or Product Data Model, to provide the path to retrieve the right pieces of information from the product model. For instance, the number of impression of the injection mould, or a specific property of the product's plastic material that this injection mould is going to produce.

Besides supporting the information data model relationships, the knowledge link elements provide the knowledge stored in the Product Range Model a good portability against further changes in the structure of the Product Data Model.

The Knowledge Link element is explained in chapter 8.
Figure 4-3 – Elements Involved in the PRM and PM relationship
5. Representing Injection Moulds as a Product Range

5.1. Introduction

This chapter investigates the representation of the injection mould as a class of product range and explores the related elements and design aspects. The general structure, functionality and corresponding design solutions for injection moulds are briefly described in section 5.2. The interactions that take place during the design process for injection moulds are first explored in section 5.3. This chapter also identifies the general relationships that must be captured by the Product Range Model, and establishes a basis for the relationships to be presented in detail in Chapters 6 and 7.

5.2. Injection Mould Product Range

5.2.1. General Injection Mould Structure

The injection mould is a tool used in the injection moulding process, which produces the plastic component from molten plastic. There exist a wide variety of concepts in injection mould design and mould configurations, e.g. two plate, three plate moulds, etc. Even though there are unique aspects to be considered in each particular design case, generic areas of functionality and a general configuration for injection moulds can be identified.

(Menges and Mohren 1993) provided basic classifications for the main structures of injection moulds based on the quantity of plates and ejection techniques. A general configuration of an injection mould is depicted in Figure 5-1, which identifies some of the main elements.

The main aspects that classify the general configuration of injection mould tools may be:

- The number of moulding impression, which defines how many plastic components are produced by each injection cycle;
- The mould configuration, which defines the number of main mould plates, e.g. 2-Plates or 3-Plates;
- The type of runner, which may define the existence, or absence, and type of the runner system, e.g. cold runner or hot runner system (runnerless), and

- The ejection technique, which defines the requirement for, or absence, of additional plates for the efficient ejection of the plastic component, i.e. such as stripper plate, sliders or lifters.

Figure 5-1 – Injection mould tool general description

In addition to consideration for the main plates, injection moulding tools encompass specific systems, such as feeding, cooling, ejection systems, which are closely associated with the steps, which constitute the injection moulding process cycle. These steps, in turn, define the main functions of the injection mould.

In this respect, injection moulds can be characterised as a class (kind) of product range, where a set of functions and means to achieve these functions can be defined. This is shown in Table 5-1 below.
Chapter 5

### Injection Mould Functions

The injection mould functions have been identified in (Menges and Mohren 1993; Rees 1995; Rosato and Rosato 1995) and can be associated with some of the main systems of the injection mould, as depicted in Table 5-1.

However, the relationships presented above are of a general or abstract nature, since some of these functions can be further decomposed into more elementary sub-functions, to form a functional tree for the injection mould product range. For example, the feed mould function can be further decomposed into three main sub-function, e.g. provide means of entry into mould interior (achieved by the sprue system), convey molten material from sprue to impressions (achieved by the runner system) and finally the control of material flow into the impression (achieved by the gate system) (Rosato and Rosato 1995). A similar kind of decomposition can be carried out for the cooling, ejection or venting functions. As a result of the functional decomposition, the sets of possible ways to achieve the lower level of functions become more restricted.

In this respect, this research has explored these functions to provide a more detailed definition for the injection mould product range functionality (Figure 5-2). This functional structure can capture a more precise meaning for the intended functional requirement of injection mould design, and has been taken as a basis for the definition of the functional structure in the injection mould Product Range Model.

#### Table 5-1 – General injection mould functions and systems

<table>
<thead>
<tr>
<th>Injection Mould Function</th>
<th>Injection Mould Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produce the shape, size, and surface texture of the moulded component;</td>
<td>Impression (cavity and core)</td>
</tr>
<tr>
<td>Facilitate the escape of trapped air and gas;</td>
<td>Venting system</td>
</tr>
<tr>
<td>Provide the flow of material from the machine nozzle to the moulding impressions;</td>
<td>Feeding System</td>
</tr>
<tr>
<td>Eject the moulded article from the impression;</td>
<td>Ejection system</td>
</tr>
<tr>
<td>Control the temperature of the mould to chill plastic to rigid state</td>
<td>Cooling system</td>
</tr>
<tr>
<td>Hold impressions (cavity and core) in fixed and correct position</td>
<td>Mould base (plates) and Alignment system</td>
</tr>
<tr>
<td>Fulfil requirements for producing economically and product functionality</td>
<td>Impression system and mould configuration</td>
</tr>
</tbody>
</table>
It is important to note that the decomposition, shown in Figure 5-2, captures a general representation of the injection mould product range functions, and that it is not necessary for all moulds to encompass all these functions. Rather, the configuration of the functions depends on the main specification and decisions made during a particular design of an injection mould. In a similar way, the resulting functional structure depends, on the experience and understanding of individual companies.

The following section presents the solutions, which are commonly applied in the design of injection moulds.

5.2.3. Injection Mould Design Solutions

There are a large number of mould design solutions, and this reflects the diversity in approaches for satisfying the functional requirements of injection moulds. The literature contains vast amounts of material about this subject, and this is usually structured in terms of the main systems of the injection mould (Pye 1989; Cracknell and Dyson 1993; Menges and Mohren 1993; Rees 1995). Even though differences can
be perceived in terms of the varied terminology used, these design solutions reflect the knowledge acquired, in specific areas, to the injection mould product range.

In this thesis five main systems of injection mould were of interest and have been studied more closely, namely feeding system, cooling system, ejection system, venting system and impression distribution. The design solutions related to these systems have been extracted mainly from the literature addressed above and also from the industrial collaborator.

5.2.3.1. Solutions for Feeding System

The feeding system is responsible for conducting the molten material from the nozzle of the injection machine to each mould impression. The feed system is composed of three sub-systems, namely the sprue, runner and gate systems. The sprue is the "interface" between the injection moulding machine nozzle and the mould. Even though the sprue is a key element of the injection mould, there are few variations in terms of design which are significant. For this reason, only the runner and gate systems have been addressed in this research.

5.2.3.1.1. Runner system

The runner system is responsible for conducting the material from the sprue to the gate. The runner should be designed to encourage equal pressure transmission to each mould impression, utilise the shortest possible flow route to each impression, be ejected with ease from the mould, and be of the lowest possible shot weight while functioning adequately. Two kinds of runner systems have been addressed by the literature, these are the cold and hot runner systems.

The hot runner enables the extension of the conditions present at the machine nozzle to the gate, and usually comes as a separate module to the mould, which is called a hot manifold. The use of this technology results in less scrap from the injection moulding process, but is more costly to implement and the cost factor must be considered. The hot runner is usually bought as a standard element, where the main design aspects to be considered are: the number of impressions, the distribution and distances between the hot nozzles and the processed plastic material.

In the case of cold runner systems, two other aspects need to be considered, namely the shape of the cross section and the layout of the runner. While the cross section of
the runner is designed to provide the best flow of material through the mould, the layout should offer the shortest, or best, path between the machine nozzle and the gate. The typical shapes for the cross sections are full round (circular), semi-circular, trapezoidal, modified trapezoidal, rectangular and cross types. The choice for their usage depends on the type of plastic material, temperatures, layout chosen and other mould configurations. The runner layout is strongly dependent upon the number of impression defined and can be T-shape, S-shape, H-shape, circular, etc.. A general configuration of the runner system is depicted in Figure 5-3.

5.2.3.1.2 Gate System

The gate is a channel, or orifice, that controls the way in which the polymer flows from the runner into the mould impression. The gate has a small cross-sectional area than the runner. The gates are designed to enable quickly freezing of the molten polymer, to allow simple or automatic degating, to provide small witness marks, to provide better control of the filling of multi-impression and to pack the impression with material in excess of that required to compensate for shrinkage.
Depending on the design specifications and configuration of the injection mould the gate used can be: sprue, edge, rectangular, overlap, fan, tab, diaphragm, ring, spoke, flash, submarine, winkle or pinpoint types (Figure 5-4).

![Gate System Diagram]

Figure 5-4 – Gate system design options

### 5.2.3.2. Solutions for Ejection System

The main function of the ejection system is to extract, automatically, or otherwise, the moulded component from the injection mould. Three sub-systems compose the ejection system, namely: ejection grid, ejection plate assembly and ejection techniques (Pye 1989).

The ejector grid, named “ejection house” in Figure 5-1, is the part of mould, which supports the mould plate and provides a space into which the ejector plate assembly can be fitted and operated. The ejector plate assembly is the part of mould, where the ejector element is attached, and consists of an ejector plate, a retaining plate, one or more ejector rods, and is complemented by the guiding and supporting ejector plate assembly, return systems and stop pins.

The ejection technique is the part of mould that makes the actual ejection (contact) of the plastic component, and for this reason is the focus of this research. The ejection technique can be met by different configurations such as normal pins, stepped pins, D-Shaped pins, sleeve pins, blade ejectors, valve ejectors, air ejection, stripper ejection and pullers. Figure 5-5 presents a classification for the design options for ejection techniques.
5.2.3.3. Solutions for Cooling System

The main function of the cooling system is to control the temperatures in the mould to allow both good conditions for the hot material flow inside the mould and ejection of the plastic component in a rigid state. The application of fluids, mainly water, through the holes or channels characterises the most common techniques used. However, other material conduction techniques are also used, for instance, head-rods.

Depending on the characteristics of the plastic component, the design specifications and main functionality of the injection mould, different options of cooling system can be used, such as passing channels ("U", Rectangular, "Z" types circuits), concentric channels, angle hole, baffles, stepped layouts, etc.. Figure 5-6 shows a general representation of these options.
5.2.3.4. Solutions for Venting System

The venting system plays two principal roles in the injection moulding cycle. It is responsible, firstly, for permitting the escape of air entrapped inside the impression to escape, resulting in the faster and complete filling of the impression thereby avoiding gas marks on the moulded product. Secondly, it must allow the influx of air to break the vacuum, to facilitate easier ejection of the plastic component.

A good venting system design must establish a balance between two conflicting objectives; firstly, the venting system must provide the least resistance to escaping air through the use of the largest escape channels possible. The design must, however, not permit the breach of plastic material, under pressure during the injection cycle, into the escape channel. Depending on the length and layout, venting may also be necessary to permit a faster flow of material inside of the runner channels.

The configurations for venting design solutions consist, typically, of a combination of parting line venting, vent pins and venting inserts (Figure 5-7).
5.2.3.5. Impression Distribution Design Solutions

The impression distribution system is responsible for the layout of the impressions in the mould, and aims to provide a better balance of forces inside the mould with maximum productivity, e.g. number of components per cycle. The design options are highly dependent on the number of impressions defined. This work is not concerned with the aspects related with the number of impression calculations. More information about such aspects can be found in (Menges and Mohren 1993; Rosato and Rosato 1995).

The configurations for impression distribution can be divided basically in circular, rectangular and in line (Figure 5-8). In the case of this research, these kinds of design solutions have been identified based on the experience of the industrial collaborator mentioned in Chapter 3.
5.2.4. **Standard Mould Design Solutions**

The use of standard components is largely practised in injection mould design and manufacturing. Companies such as Hasco®, D-M-E®, Isotrix®, offer a wide range of injection mould elements including for example, plates, ejectors elements, cooling elements, and hot runners systems, which can be ordered using their identification codes. (Culley and Theobald 1997; Culley 1999) has addressed the importance of standard components as essential ingredients for all engineering systems and (Kruth, Willems et al. 1997) has focused on this issue in injection mould design area.

Thus, the availability of information relating to standard components is a significant aspect in the provision of support to the injection mould design process and, can be seen as one typical way in which information can be readily reused. However, it is important to realise that information about standard components represent only a limited part of the design solutions, and standard components can only be selected after a particular type of solution has been chosen. In turn, a particular design solution must be based on design criteria and knowledge.

This work provides support to the designer by offering a combination of valid design solution concepts, which, in turn, will be associated, when possible, to relating standard solutions.
5.2.5. A General Relationship between Injection Mould Functions and Design Solutions

In order to provide the most appropriate design support, in terms of capturing injection mould design information, a formal relationship must be established between functions and alternative design solutions. This relationship must be able to represent and identify possible design solutions, which can be applied to achieve a specific function. The reverse must be also true, where it must be possible to identify what functions can be achieved by a particular design solution. For instance, to fulfil the function: Eject Plastic Component, different ejection techniques can be used, such as eject pins, eject valves, stripper plates, air, etc. On the other hand, the eject pins design solution is not exclusively used for the above function, but can also be used for Eject Feature or Eject Feeding functions (Figure 5-9). These relationships are not new and are present in the mind of injection mould designers. However, they will form the basis of the Product Range Model, providing a formal representation of these pieces of information and knowledge.

![Ejection Functions and Design Solutions](image)

**Figure 5-9 - Ejection functions and design solutions relationships**

In this case, functions and design solutions are part of the information, which is captured and used to support the injection mould design reuse process. For example, "eject plastic component" is one function, which is common for most injection moulds. Similarly "ejection pins" can be a design solution normally considered when evaluating ways of eject the plastic component from the mould.
While functions and design solutions themselves are considered information, the relationship between them captures part of the knowledge associated with their use and application. For example, the relationship between the function "eject plastic component" and a set of ejection design solutions represents when such set of design solutions should be considered. Similarly, "ejection pins" and "pullers" are related to the function "eject feeding". Although these pieces of information and knowledge together can narrow the set of design solution to achieve a specific injection mould function, they are not enough to provide valid options to the designer. Another information and knowledge need to be considered and are discussed in section 5.3.

In addition to the relationship between functions and design solutions, another association, between functions, is highlighted in Figure 5-9, where a parent function can be decomposed into one or more sub-functions, resulting in a hierarchical functional tree.

Figure 5-10 shows a general representation of the above relationships based on the notation of UML class diagram. It demonstrates how the functions and design solutions have been modelled in the Product Range Model. The relationship between functions and design solutions is characterised as many to many (M::N), and the relationship between functions is characterised as one to many (1::N). More details relating to this association in the Product Range Model are described in Chapter 6.

- Figure 5-10 – General functions and design solutions relationship

### 5.3. Injection Mould Design Interactions

#### 5.3.1. General Interaction Aspects

The process of injection mould design is highly iterative and complex and is influenced by the characteristics and specifications of the plastic component, the
specifications of the injection mould, and capabilities of the manufacturing resources which produce the mould and the plastic component (Chin and Wong 1996).

In general, injection mould design can be divided into two distinct levels of decision making, named initial design decisions and specific system decisions (Costa and Young 1998). During initial design, the general configuration of the mould is decided, e.g. number of cavities, 2 or 3 plates, cold or hot runner, etc. Decisions related to which specific technique is used for each injection mould system are made at the second level.

Two classes of interactions with the injection mould system solutions can also be identified at the two levels of design decisions mentioned previously. The first one is related to how the initial design specifications and requirements can constrain the set of options for possible design solutions, whilst the second is related to how design choices made for each injection mould system can interact with further design decisions (Figure 5-11).

Figure 5-11 - Particular interaction types in the injection mould design process (Costa and Young 1998)

The recognition of these interactions in the injection mould design is not new and have been addressed by the traditional literature in this area (Menges and Mohren 1993; Rees 1995). Recent work has highlighted the importance of exploring such subject (Lee, Chen et al. 1997). Chapter 4 has discussed some of these works. This
research explores the capture of these interactions into an object-oriented approach and provides an information structure to represent them.

As in the case of injection mould functions and design solutions, the injection mould interactions have been extracted in part from the injection mould literature addressed above and from the industrial collaborator. These interactions play a critical role in the definition of knowledge related to the application of each injection mould design solution, complementing the relationship between functions and design solutions and supporting the offer of valid design solutions into particular design situations.

In this work these interactions are mainly related to number of impressions, type of runner system, distribution of the impressions in the mould plate, type of ejection system, mould configuration, type of degating, type of gates, type of runner layout and type of runner cross section area.

Figure 5-12 shows the initial design specification of the injection mould and the plastic component, and their interactions with each other. For instance, the area of the plastic component is a significant factor in the determination of the number of impressions in the mould, while the choice of the feeding point location (in the parting line or top) can be decisive for the mould configuration, e.g. 2 or 3 plates.

In addition to the specifications for the plastic component, there can also be interactions arising from the initial specification of the injection mould. For example, the choice of a hot runner, as well the decisions relating to the injection mould machine and number of impressions, have a direct influence on the mould configuration.

However, these interactions are highly dependent upon each other and relatively apparent. Furthermore, with the growing tendency of global manufacture and use of external sub-contractors in the area of injection moulding, these kinds of decisions are increasingly being made during initial stages of the injection mould design, and hence used as an important input for driving the first type of interactions identified in Figure 5-11.
### Injection Mould Specifications

<table>
<thead>
<tr>
<th>Injection Mould Specifications</th>
<th>Number of Impression</th>
<th>Degating Type</th>
<th>Type of Runner</th>
<th>Estimate Shot Weight</th>
<th>Injection Machine</th>
<th>Mould Configuration</th>
<th>Plates Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Impression</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Degating Type</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of Runner</td>
<td></td>
<td>✓</td>
<td>X</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Estimate Shot Weight</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Injection Machine</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mould Configuration</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plates Set</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

### Plastic Component Specifications

<table>
<thead>
<tr>
<th>Plastic Component Specifications</th>
<th>Component Volume/ Shot Weight</th>
<th>Component Projected Area</th>
<th>Production Annual Rate</th>
<th>Feeding Point</th>
<th>Moulding Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Strong Interaction**
- **Light/Indirect Interaction**

Figure 5-12 – Interactions between initial decisions for plastic component and injection mould design

#### 5.3.2. Interactions between Initial Design Decisions and Injection Mould System Solutions

Based on the initial design specifications of the injection mould some assumptions can be made about different options of injection mould system solutions. Ideally, the initial design specifications will be clear enough to allow certain assumptions to be made about the requirements of the injection mould system solutions. In some
circumstances, further clarification will be needed and questions will be raised about the different possible options.

Figure 5-13 shows the interactions between the initial design specifications and injection mould design system solutions. For instance, depending on the method of degating required (automatic or manual), type of the runner (cold or hot), mould configuration (2 or 3 plates) and feeding point, the resulting types of gate solutions can be significantly narrowed to a few or possibly one choice.

### Injection Mould System Solutions

<table>
<thead>
<tr>
<th>Injection Mould Specifications (Initial Decisions)</th>
<th>Impression Distribution</th>
<th>Runner System</th>
<th>Gate System</th>
<th>Ejection System</th>
<th>Cooling System</th>
<th>Venting System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Impression</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Degating Type</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Type of Runner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Injection Machine</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Mould Configuration</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Feeding Point</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parting Line (P/L)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Strong Interaction**
- **Light/Indirect Interaction**

Figure 5-13 – Interactions between initial design decisions and injection mould system solutions

Hence, if these interactions are well captured, the initial phases of the injection mould design can be significantly improved by providing the designer with solutions, which can actually fulfil the specifications. A major emphasis of this research, therefore, involves the identification and capture of the interactions between product specifications and design solutions.
5.3.3. **Interactions between Injection Mould System Solutions**

The injection mould design, as most other design processes, is not a sequential straightforward process; as solutions are chosen, new forms of interactions take place. These forms of interactions are defined in this work as between design solutions for different parts of the mould system, and a general representation of the interactions between injection mould systems is depicted in Figure 5-14. For example, depending on the distribution of mould impressions chosen for the mould plate, different types of runner layout can be chosen, and depending on the type of ejection system selected some kinds of cooling solutions can be ruled out.

### Injection Mould System Solutions

<table>
<thead>
<tr>
<th>Injection Mould Systems (Design Decisions)</th>
<th>Impression Distribution</th>
<th>Runner System</th>
<th>Gate System</th>
<th>Ejection System</th>
<th>Cooling System</th>
<th>Venting System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impression Distribution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runner System</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gate System</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ejection System</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cooling System</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Venting System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

- **Strong Interaction**
- **Light/Indirect Interaction**

Figure 5-14 - Interactions between injection mould system solutions

The interactions presented previously can be related to different design aspects such as geometric or conceptual aspects. For instance, the decision relating to the ejection of the plastic component by a stripper ejection technique, prohibits the use of circular or hexagonal cross section of the runner system which would cause problems during the ejection of the runner system (Figure 5-15(a)). In this case the problem is not geometric, but based on the other aspects such as from previous experience and mechanical aspects. These have been defined as the more significant interactions for this work.
Other interactions can be identified, for example, in the choice of cooling the core internally, or of ejecting the plastic component by its internal part (Figure 5-15(b)). In this case the interactions are mainly related with geometric aspects.

(a) – Interaction between ejection solution and runner solutions

(b) – Interaction between ejection solution and cooling solutions

Figure 5-15 – Interactions between design solutions

Additional kinds of interactions can also be identified, such as manufacturing aspects, which can be considered during the design solutions, where choices or particular interactions that can be captured for situations and experiences relating to individual companies. For instance, the availability or not of tools or machines to produce a cooling channel or a specific cross section geometry for the runner system.

This work recognise the existence of these different types of interactions, however special attention has been focused on the conceptual ones.
5.3.4. A General Relationship between Design Solutions and Product Information

As addressed in section 5.3.1, the sets of interactions previously discussed support the definition of the design criteria that define the application of injection mould design solutions, in a design situation. Thus, these interactions play a critical role in that they represent the knowledge and experience, which a company acquires through time. An important aspect is the capture of these interactions, or knowledge, which is made available for reuse. In this respect, one of the main issues, addressed in Chapter 4, is how to capture this knowledge, or design criteria in the Product Range Model.

This work has defined that each design solution has an association with a set of interactions, which together will determine its design criteria. These interactions are checked, against product design information, to determine the suitability of a design solution, during the design process.

As shown in the example in Figure 5-16, a particular type of design solution, e.g. Submarine Gate, must meet some design criteria, which is represented by the interactions with mould configuration, feeding point and ejection technique chosen, in order to become a suitable design solution. In this example each interaction captures not only specific information, e.g. number of impression, but also the way in which this information is used, i.e. knowledge. For example the number of impression, retrieved from the Product Model, must be equal or greater than 2 impressions.

However, the interactions that represent these design criteria are not exclusively used by this design solution, but they can be also used to evaluate other kinds of gates solutions, or even other kinds of design solutions, such runner, cooling or ejection systems.

As a result, the relationship between design solutions and the interactions that represent their design criteria, is defined as bi-directional and M::N type, where a design solution can have one or more interaction to be evaluated, and one interaction can be used by one or more design solutions. This kind of relationship takes into account that a design criterion can be used by one or more design solution, thus avoiding duplication of knowledge.
The design criteria are represented in this research by the interactions and the general relationship between design solutions and interactions are depicted in Figure 5-17. The nature of the interactions in the Product Range Model will be explored in detail in Chapter 7.

**Figure 5-17 – General design solutions and interactions relationship**

5.4. Summary

This chapter has explored the characteristics of injection moulds as a type of product range, and the basic information necessary for the definition of the Product Range Model. The general relationships between injection mould functions and design solutions, and between design solutions and their interactions were identified. While the first type of relationship provides an association between each function and all possible ways to achieve it, the second relationship captures the knowledge behind the design criteria in order to evaluate each design solution. Two kinds of interactions were focused on, i.e. between design solutions and product specifications and between
the design solutions themselves. The information explored in this section provides a basis for understanding the issues addressed in chapters 6 and 7, where the Product Range Model information structure is discussed.

The information structure of the Product Range Model is now discussed in the following chapters.
6. Modelling Function and Design Solution Sets in the Product Range Model

6.1. Introduction

This chapter describes the information modelling performed to capture the function and design solution structure representations in the Product Range Model. The representation of the basic functional and design solutions structures are described in sections 6.2 and 6.3 respectively. Section 6.4 addresses complementary information for supporting the design decision process by expanding those structures, presented in the previous sections.

The information structure presented in this chapter has been built on the injection mould product range characteristics discussed in the Chapter 5 and makes the foundation for the critical issues investigated in this research, addressed in the Chapters 7 and 8.

6.2. Functional Structure

Based on the injection mould functions, discussed in Chapter 5, a UML class diagram representation of the basic functional structure in the Product Range Model is presented in Figure 6-1, where the internal relationship between functions themselves and their attributes are highlighted. This representation is depicted in terms of the Function class. As addressed in the previous chapter, the function elements also have an external association with the set of feasible design solutions. This association is discussed in section 6.4.

The internal relationship allows each function to be decomposed into other functions (sub-functions) hence defining a functional hierarchy structure. The Sub-Functions definition is supported by the Function_Level attribute, which identifies the particular function's level in the tree, and by the Function Root, which defines its hierarchy, or parent function. This approach provides significant flexibility within the structure definition, in addition to using the power of object recursivity.

Each function also has a name (Function_Name), a unique identifier (Function_ID) and a type (Function_Type) attribute. While the first two of these attributes support
the identification aspects, the latter provides a way of classifying such functions into different systems, or structures, of the product. For instance, in the case of the injection mould, different systems, such as ejection, cooling and feeding systems are required for different basic injection mould functions, e.g. eject, cool or feed the mould.

The UML class diagram in Figure 6-2 represents the class Function, which is used to illustrate the major relationships and attributes captured by the functional structure. Each element in the diagram represents an actual function with its own attributes and relationships, such as its specific identification, type, level and associations with root and sub-functions.

While this example has focused on Ejection Function types, e.g. Function_Type = 1, instantiations can also be represented for the different types of functions addressed in Chapter 5, such as feeding, cooling, venting, etc.

Figure 6-2 shows also the relationships between function elements in the hierarchical tree by the attributes Function_ROOT, Sub_Functions and Function_Level. For example, Eject_Base function has Eject_Impression as a parent function, which in turn has Ejection_Function as parent function and Eject_Wall as sub-function.

Thus, the functional hierarchical structure defined in this work allows the capture of two important aspects already discussed in Chapter 5. These are refinement in meaning of the functions as they are decomposed into sub-functions, thus providing a precise guidance to the designer in his/her functional choices, and, as a consequence...
of this, a more restricted set of possible design solutions associated with each function.

![Diagram of Ejection Functions](image)

**Figure 6-2 – Instanciation example of Ejection Functions**

### 6.3. Design Solutions Structure

The design solutions, discussed in Chapter 5, represent physical principles that can be applied to achieve particular functions of the product range. Also, as stated in Chapter 5, the injection mould product range has been traditionally divided into different main systems, which group different sets of related design solutions.

In this way, the structure of information defined for capturing the design solutions, within the Product Range Model, is specific to injection moulds rather than generic as in the one used to represent functions. The elements of the design solutions are identified by their sub-classes, such as *Ejection_DS, Gate_DS, Layout_Runner_DS*, etc. (Figure 6-3). Like the functions, each design solution has attributes related to its identification, e.g. *DS_Name* and *DS_ID*.

This structure provides a good representation of the injection mould product architecture in terms of main systems and, consequently, their associations with the main injection mould functions.
As addressed in chapter 5, the use of commercial standard components is a common practice in injection mould design and manufacture, so an alternative option to manufacturing the mould is buying the component. However, besides the standard components, knowledge about ways of manufacturing specific design solutions are also known by each particular company and hence, can be considered as an important piece of information in supporting the designer in his/her decisions, for instance, evaluation of the costs of a specific design decision.

In this respect, a general class termed *Manufacturing_Options* has been defined as part of the design solutions information structure, and hence each design solution can have a relationship with the *Manufacturing_Options* class, through the attribute named *Manufacturing_Options_Coll*.

The class *Manufacturing_Options*, in turn, has been expanded in *Standard_Solutions* or *Manufactured_Solutions* class types. While the first one captures information of the commercial standard element, the second captures the knowledge for manufacturing specific design solutions.

The standard solutions have received particular interest in this work, since they can represent actual instances of their design solutions. Thus, each standard element, besides the attributes that identify its properties, such as code (*Stand_Code*) and supplier (*Supplier*), has also an association with the *Design_Solution* class through the attribute *Design_Solution_Std*, defined in the *Manufacturing_Option* class.

Figure 6-4 depicts the UML representation of design solution instances. Two objects, derived from two different classes, *Ejection_DS* and *Gate_DS*, are represented and because both are types of *Design_Solutions* class, they inherit all its attributes. Both instances have their particular identification attributes (*DS_Name* and *DS_ID*).

The design solution element *Normal_Ejection_Pins* has an association with different manufacturing options, more specifically with different *Standard_Solutions* objects, (*Z401.* and *Typ(e) A/E..*). This relationship is supported by the attribute *Manufacturing_Option_Coll* in the design solution element. Each standard component has its own attributes related with their identification, e.g. code, name and suppliers (*HASCO* and *D-M-E*).
Figure 6-3 – PRM Design Solutions Structure
6.4. Function and Design Solution Sets Supporting Design

This section develops complementary information related to the application of function and design solution sets, into the Product Range Model, to support the design decision activities. Thus, besides the relationships between functions and design solutions, the aspects related to ways of offering valid information to the designer and the additional attributes necessary to support such aspects need to be focused.

The information structures explored so far, in the previous sub-sections, have been mainly concerned with the modelling of the injection mould individual functions and design solutions structures into the Product Range Model.

However, in order to support the design decision process, the Product Range Model must provide useful information about valid design solutions to be used by the designer. This information is a result of the designer's enquiring about a specific function and the interactions of each possible design solution with both the product specifications and design solutions already chosen.
Thus, to support the Product Range Model functionality previously stated, the definition of a formal association between functions and design solutions is required. Chapter 5 has already defined this relationship as bi-directional and M::N type, where each function can have a set of possible design solutions to achieve it, and each design solution, in turn, can be a potential solution for several different functions. This relationship is shown in Figure 6-5, and is supported by the attributes Design_Solution_Coll in the Function class and Function_Coll in the Design_Solutions class.

![Diagram of General association between functions and design solutions](image)

Figure 6-5 – General association between functions and design solutions

Whilst this relationship provides the designer with different ways of achieving a particular function, it is still not enough to support him/her with valid information for each particular new product designed.

In this respect, three main issues must be considered in the representation of the function and design solutions into the Product Range Model, these are:

- What makes a design solution valid or non-valid for the design application?
- How to represent these different design solution states?
- How to associate the design solution states to a particular function enquiry?

The first issue is related with how a particular design solution can meet its design criteria throughout the interactions with the product and design information. This is the critical issue addressed by this work and is explored in Chapter 7.

However, regardless of the approach taken to solve the first issue, based on the result of the information about design decisions and their interactions, each design solution can assume different states for its application in the product design, as depicted in Figure 6-6.
Figure 6-6 – Different design solutions states during the design process
Figure 6-6 shows the basic interface between the designer and the Product Range Model, and how the results of the functional enquiry and design solution interaction checking process are presented to him/her. In the example, the set of possible design solutions to achieve the function Eject Product, after the interaction checking process, has been transformed into two sub-sets of design solutions, named accepted and rejected design solutions. A cross identifies the rejected design solutions. Also, based on this result, the designer can still select, from the accepted design solutions sub-set, the one(s) that will make part of the final product design, in this case, exemplified by the Ejection Pins solution.

Thus, two different states of the design solutions have been defined in this research, namely accepted and rejected. Also, for the accepted state, a design solution can still assume an additional state, identified as selected, which is the result of the end user action, e.g. designer's selection.

The relationship between the design solutions and the product model, depicted in Figure 6-6, to support the design decisions is explored in Chapter 7.

Figure 6-7 shows a UML State Transition Diagram to represent these different states and why such states are changed. For supporting the state definition, the attribute $DS_{-}State$ has been added in the Design_Solutions class (Figure 6-8). This provides the answer to the second issue addressed in the beginning of this section.
Finally, as a result of the previously described process, each function will have an association not only with the set of all possible design solutions that can fulfil it, but also with two temporary sub-sets of design solutions.

While the set of all possible design solutions is represented by the `Design_Solutions_Coll` attribute in the `Function` class, the other two sub-sets are supported by the introduction of the attributes `Accepted_Design_Solutions` and `Reject_Design_Solutions`. Similarly, a third set of design solutions is defined for the case of selected design solutions chosen by the designer, and is represented by the `Selected_Design_Solution` attribute (Figure 6-8). This provides the answer to the third issue addressed in the beginning of this section.

These attributes can provide the designer with a more complete set of information about his/her functional enquiry, and therefore offer him/her good support in terms of design information, which is one of the main objectives of the Product Range Model.

Figure 6-8 – Expanded functions and design solutions associations

More details about how this structure supports the design process is presented in the Chapter 9, in the experimental system development.

6.5. Summary

This chapter has defined the general information structure to capture the product range functions and design solutions in the Product Range Model, and to support the design decision process. This information structure forms the foundation of this work and is used to explore the critical issues of this research namely, the design solutions interactions with product information, which are addressed the next chapter of this thesis.
7. Modelling Design Solution Interactions in the Product Range Model

7.1. Introduction

This chapter addresses one of the key issues of this thesis, which is the capture and representation of information and knowledge to support the design solution interactions with other product design information. The definition of elements termed Interactions is presented alongside the information structure used to represent them within the Product Range Model. Section 7.2 explores the general aspects related to the definition of the interaction elements, whilst section 7.3 presents a classification of the types of interactions defined in this work, as well as the information structure used to represent them. Section 7.4 explores the application of the interactions to support the design decisions, and this is followed by the explanation of the way in which the interaction elements are tested, in section 7.5.

The interaction information structure, presented in this chapter, complements the relationships between functions and design solutions, presented in Chapter 6, providing information about the valid design solutions that can be applied to satisfy specific design conditions.

The structure identified in this chapter, along with the information structures presented in chapters 6 and 8, defines also the general information structure of the Product Range Model, which is used in the definition of the experimental system developed in chapter 9.

7.2. Design Solutions and Product Information Interactions

7.2.1. General Requirements

The discussion carried out in Chapter 4 identified two issues, related to the relationship between the product range design solutions and product information, needing closer examination in the definition of the Product Range Model. These are, firstly how to represent the heuristic knowledge that capture interactions between product range design solutions and product information, and secondly how to
represent the relationship between the Product Range Model and product model to support the evaluation of these interactions (Figure 4-3). This section is mainly focused on the first issue; the second issue is addressed in the next chapter.

Chapter 5 has highlighted two main types of interactions during the process of selecting design solutions to achieve injection mould functions, namely interactions with product specifications and interactions with other previously made design solution decisions. This work has identified that these interactions are associated with different kinds of injection moulding information, such as number of impressions, mould configuration, type of feed system, properties of the plastic component, techniques chosen for ejection system, runner system, gate system, and cooling system. Figure 5-16 presented an example of some of these interactions.

Together these interactions capture the heuristic knowledge necessary to represent the design criteria applied to each product range design solution. Formal ways of capturing this in the Product Range Model are necessary.

7.2.2. Interactions Elements - Definition

This research has provided a contribution in the definition of a set of elements called Interactions that capture the design information and knowledge, which represent the design criteria for application of each design solution. These elements are responsible for determining the validity of the design solution with which they are associated in particular design cases.

To support these elements and their relationships this work has defined an information structure, which is depicted in Figure 7-1 using a general UML representation of the Interactions class and its relationship with the Design_Solutions class.

Figure 7-1 – Interactions general relationships with Design Solutions
Thus, the information and knowledge required for the evaluation of each interaction element and the determination of the validity of each design solutions are captured together in the combination of the attributes and the behaviour of both classes.

As in the case of the functions and design solutions definitions, to avoid duplication of elements, each interaction element has been defined as unique, and hence has its own identifiers, i.e. `Interaction_ID` and `Interaction_Name`. The attributes `Interaction_Collection` in the `Design_Solutions` class and `Design_Solution_Root` in the `Interactions` class support the many to many bi-directional relationship between the design solution and interactions elements. This allows each design solution to have one or more interaction elements defining its validity and also enables the same interaction element to be used in the definition of the design criteria of one or more design solution.

However, in addition to the identification attributes and relationships with design solutions, there are also issues that need to be considered in the modelling and representation of the interaction elements. Some of these issues are related to what kind of information and knowledge should be stored in the interaction elements; how to test this information and knowledge; how these elements can be arranged together to provide better representation of the design criteria; and where to retrieve the value to be compared. These issues are addressed in the subsequent sections of this chapter.

### 7.3. Representing Knowledge in the Interactions Elements

#### 7.3.1. General Aspects

The information modelled in each interaction element captures part of the knowledge that defines the design criteria for application of each specific design solution. Thus, the interaction elements must represent the validity of particular conditions that determine whether, or not, the application of a specific design solution is appropriate.

In this respect, the design criteria are represented as a set of interaction elements, arranged in a way, which must capture its truthfulness. Thus, the interaction elements, which form this set, must all be true. This means to apply `Boolean "&"` truth tests to all the constituents of the set in order to determine the validity of the design solution with which they are associated.
This work has adopted an object-oriented approach for the modelling of the Product Range Model information structure. In the case of the interaction elements, this approach provides several benefits as it allows the capture of complex conditions within a simple structure, enabling conditions to be extracted quickly and efficiently. This approach also allows the reusability of the interaction elements since each interaction object could be associated with several different design solutions, hence avoiding duplication. This is addressed as one important contribution of this research, which is related to the flexibility and reusability of the knowledge captured in the Product Range Model.

To provide a better representation of the design criteria through the interaction elements, two general types of interactions have been defined, named Simple and Composite interactions. While the former captures individual conditions, the latter captures combinations of different individual conditions.

7.3.2. Simple Interactions

7.3.2.1. General Definition

The simple interactions represent the elementary knowledge, or conditions, that capture part of the design criteria for application of a particular design solution. This kind of interaction can be related to a particular value of an attribute associated with the product characteristics, or with the existence, or absence, of a particular characteristic itself. The product characteristics define the representation of particular properties of a product, such as specifications, geometry, dimensions, material, etc. In this respect, the simple interactions require some kind of link mechanism, which enables retrieval of this product information, stored in the product model, in order to check their own conditions.

This research has provided also a contribution in defining the relationships between information models, i.e. Product Range Model and Product Model. This has been done through another element, termed Knowledge-Links, which is responsible for storing knowledge about the location of specific product information. This element retrieves the correct information from the product model and provides it to the simple interaction, where the comparison process is performed. The Knowledge-Links hence maintain a relationship with the simple interaction elements, which is shown in Figure
7-2. The Knowledge-Links elements are presented in Chapter 8, where the relationships between Product Range Model and product model are discussed.

Two kinds of simple interactions have been defined in this work, namely numerical and existence interactions. While the first is related to the comparison of numbers, the latter is related to the verification of existence of a specific product characteristic.

Figure 7-2 shows the UML class diagram representation for these kinds of interactions. Both, **Numerical Interaction** and **Existence Interaction** classes inherit from **Simple Interaction**, which, in turn, is a kind of **Interactions**. The attribute **Interaction_Type** has been added in the class **Interactions** to support the differentiation between types of interactions.

The **Comparator** attribute, presented in the **Simple Interaction** class, represents the mathematical relational operators, e.g. equal (=), greater than (>), not equal (#), etc., which support the type of comparison to be performed.

7.3.2.2. Numerical Interactions

The numerical interaction elements capture the comparisons between two numerical values, i.e. between a *reference value*, stored in the interaction element (Figure 7-3), and the value retrieved from the product model. For example, in “Number of Impressions == 4”, the *reference value* stored in the interaction is 4, and it will be compared with the actual number of impressions stored in the product model.
In order to allow a compatible comparison between values, the type of parameters that are being compared also need to be considered, e.g. integer, double, float, etc. In the above example, the parameter is an integer type. This is supported through the attribute \textit{Type_of_Attribute}.

Two types of numerical comparison have been identified in this kind of interaction. The first one is related to a quantified number, i.e. numbers that define a quantity or a dimension. For example, number of impressions, length, width, etc.. The second one is related to product properties, represented in the enumeration format, for example \textit{Mould\_Configuration} \{"0" = no configuration defined; "1" = 2-Plate Mould; "2" = 3-Plate Mould\}.

Although the comparison process in both cases is numeric, the results can be significantly different. In the second type of numerical comparison, the zero (0) condition must be interpreted differently, since it can mean that a particular characteristic has not been specified. Thus, an additional attribute must support the differentiation between these two types of numerical interactions.

Figure 7-3 shows the UML class diagram of an example of an element of \textit{Numerical\_Interaction} class, throughout the numerical interaction named \textit{Number\_of\_Impressions\_=_4}. As an object of such a class, it inherits all the attributes of the parent classes. The first two attributes provide identification for this interaction, while \textit{Interaction\_Type} attribute allows the recognition of the type of interaction. In this example, either design solutions elements, \textit{Rectangular\_X\_Layout} and \textit{Matrix\_Impression\_Distribution}, make use of this interaction, which requires that the number of impressions specified for the injection mould in design, be equal to 4 (\textit{Reference\_Value}). Finally, the number of impressions value is provided by a Knowledge-Link element named \textit{Number\_of\_Impression} defined in attribute \textit{pKnowlegdeLink}. 

- 108 -
7.3.2.3. Existence Interactions

The existence interaction focuses on aspects related to the existence, or absence, of a particular characteristic in the product. As previously mentioned, a characteristic defines the representation of specific properties of a product, such as a specific type geometry, type of ejection solution already defined; etc..

In the case of design solutions, this kind of interaction supports testing for equivalence or other types of comparison between particular elements and existing solutions which have previously been chosen by the designer. For example, kinds of ejection solution previously chosen by the designer might be compared to check the applicability of other kinds of design solution, e.g. cooling solutions.

In the case of existence interactions the attributes *Element_Type*, *Element_ID* and *Element_Name* define the characteristics of the element that is to be compared (Figure 7-4). While the *Element_ID* attribute defines precisely the element to be compared, the *Element_Type* attribute defines in which group of elements this comparison should be performed, e.g. ejection solutions, cooling solutions, etc. The *Element_Name* identifies the particular object class that is being compared into the element group defined in the *Element_Type*, e.g. ejection stripper plates in the ejection solutions.

Figure 7-4 shows an UML class diagram of an example of an existence interaction element. Besides the common attributes already explained for the numerical...
interactions, different values are depicted, such as Interaction_Type and different kinds of design solutions may be associated, using the Design_SolutionRoot attribute.

The following example shows the three particular attributes of the ExistenceInteraction class. Element_Name (Matrix Distribution) and Element_ID (55) identify the particular element that is being compared, and the Element_Type represents the type of elements which must be searched, in this case, Impression_Distribution design solutions.

7.3.3. Composite Interactions

7.3.3.1. General Definition

The previous section has explored the aspects of each interaction under the assumption that they are unitary. However, the capture of the heuristic knowledge for representing the appropriate design criteria of each injection mould design solution, can require more than a set of simple rules and therefore different ways of combining simple rules are necessary.

This research has defined two Boolean conditions to support these combinations, namely AND and OR interaction types. Both, AND_Interactions and
**OR_Interactions** classes, are defined as types of **Composite_Interactions** class, which in turn is also a kind of **Interactions** class (Figure 7-5).

The relationships between the **Composite_Interactions** class and **Interactions** class are highlighted in Figure 7-5, and is supported by both, the inheritance association, i.e. composite interaction is a kind of interaction, and the aggregation association, represented by the attribute **Interactions_Set**. Thus, two or more interactions, either simple or even composite can be part of the composite interactions.

Both types of composite interactions, i.e. **AND_Interaction** and **OR_Interaction** have the same kind of attributes, however the difference between them is related to the way that such interactions are evaluated, i.e. their own behavior.

7.3.3.2. **OR_Interaction**

The elements of the **OR_Interaction** type form a set of interactions within the "OR" **Boolean** condition. This means that if any of its interactions is true, i.e. approved, the whole composite interaction becomes approved as well.

An example of this kind of interaction is depicted in Figure 7-6 through the UML class/instance diagrams. One of the requirements that define the validity of the **Matrix_Impression_Distribution** and **H_Runner_Unbalanced_Layout** design solutions is that the number of impression be equal to 4 or 6 or 8. This is represented by the
interaction element *Impression* = 4,6,8, which is composed by another three interactions. The association between the composite interaction and its interaction elements is captured by the *Interaction_Set* attribute.

In this example, the interactions that compose the *Impression* = 4,6,8 interaction, are the same type, i.e. numerical type, and compare the same type of attribute, i.e. number of impressions. However, the value of *Reference_Value* attribute to be compared by each of these interactions is different.

Thus, if during the comparison process with the actual number of impression specified in the product model, any of these simple interactions is approved, the composite interaction *Impression* = 4,6 or 8 becomes also approved. As a result, the design solutions *Matrix_Impression_Distribution* and *H_Runners_Layout* can be offered to the designer as valid options, if other interactions associated with them are also approved.
7.3.3.3. **AND_interaction**

The elements of the **AND_interaction** type form a set of interactions within the "AND" Boolean condition. This means that the whole set of interactions needs to be true, i.e. approved, for the approval of the composite interaction. To explain this kind of interaction, a similar example to the one used in above has been chosen (Figure 7-7).

Based on a company's knowledge and experience, one of the design criteria for applying **In_LIne_Impression_Distribution** design solution is to have the number of impression between 2 and 8. This is represented by the Impression >=2 & <=8 composite interaction element, which is composed by another two numerical interactions elements, named Impression>=2 and Impression<=8. The attributes depicted are the same as shown in the previous example (Figure 7-6), except by the difference in the values of some of them.

![Diagram of AND interaction](image)

Figure 7-7 - Example of AND interaction element
In contrast to the previous example, both numerical interaction elements, rather than only one, need to be approved before the composite interaction $Impression \geq 2 \& \leq 8$ can be approved.

7.4. Interactions Sets Supporting Design Solution Decisions

7.4.1. Requirements to Support the Design Solution Decisions

As highlighted in section 7.3.1, each design solution must have its associated set of interaction elements approved before it can be offered to the designer as a valid design solution option. This means that each interaction, in this set, needs to be tested against its internal condition, and only based on the results of the interaction testing process will a design solution be considered accepted, or not, for its application. Thus, in addition to the knowledge captured in the interaction behaviour, further information is necessary to support the interaction testing process, as well as to provide the interpretation of the results of this process.

This section develops complementary information related to the application of interactions to support the design decision activities, and the representation of this information.

7.4.2. Interactions States

Figure 7-8 expands the representation presented in Figure 6-6 (Chapter 6) by highlighting an example of the interactions associated with injection mould design solutions. Two gate design solutions, Submarine Gate and Pinpoint Gate, are highlighted as well as their sets of interactions. Besides having particular interactions, e.g. $Mould\_Configuration$, these design solutions have also common interactions elements, such as $Runner\_Type == Cold$ or $Number\_Impressions \geq 2$. This example also highlights the power of the approach taken in providing the reusability of the interaction by different design solutions.

After each interaction condition has been tested against the product model information, three main states can be assumed. Figure 7-8 highlights only two of these states, which are represented by the "tick" and "x" boxes and mean approved and reproved interactions states, respectively.
Figure 7-8 – Different states of interactions into Product Range Model
For instance, the Submarine Gate design solution has had all of its interaction states approved, and hence this design solution assumes the state of an accepted design solution. On the other hand, the Pinpoint Gate design solution has had two of its interaction states reproved, i.e. Feeding_Point == Top and Mould_Configuration == 3-Plates, and therefore it assumes the state of a rejected design solution in this design situation.

A third interaction state named non-evaluated, can also be assumed by an interaction after being tested. This situation occurs when specific information is required but does not yet exist to be retrieved from the product model, and as consequence no comparison can be performed by the interaction element. For instance, in the example shown in Figure 7-8 one of the interaction elements is related to the type of ejection system chosen, i.e. Ejection_Type ## Stripper. In this case, if the designer had not yet decided on the type of ejection system this interaction can not be evaluated, and therefore it can not be either considered approved or reproved.

In terms of the final evaluation of the design solution, the non-evaluated interaction state is treated as an approved interaction, i.e. it allows the design solution to be offered to the designer as an accepted design solution, since no reproved interactions have been identified. However, in this situation the designer should be aware of the conditions, or decision, that have to be made if he/she chooses the design solution associated with this interaction state. In this respect the non-evaluated interactions are also an important element to support the design decisions and, therefore, they should be available to the designer during the design.

Figure 7-9 depicts a UML State Transition diagram representing the interaction states named Reproved_Interaction, Approved_Interaction and Non_Evaluated_Interaction. An additional state named Neutral_Interaction identifies the initial state of all interactions before being tested.

The first two interaction states, Reproved_Interaction and Approved_Interaction, are results of the interaction testing process against product information retrieved from the product model. On the other hand, the third interaction state, Non_Evaluated_Interaction, is a result of the absence of the value associated with a product information, which does not allow the checking process to be performed. The interpretation of these results is discussed in the next section. The interaction states
are supported by the attribute named `Interaction_Status` in the `Interactions` class (Figure 7-10).

![UML State and Transition diagram for Interaction states](image)

Figure 7-9 – UML State and Transition diagram for Interaction states

### 7.4.3. Interaction States Associated with Design Solutions

As highlighted in Chapter 6, each design solution can assume either of two different states namely `Accepted_Design_Solution` and `Rejected_Design_Solution`, depending on the interaction testing process results achieved for each interaction associated with the design solution. However, the designer can only appreciate a specific design solution state if the reasons, of "why" such a state has been achieved, are presented to him/her.

In this respect, after the interaction testing process, each design solution should hold different collections of interactions based on their states. This can provide the designer with additional information about the reasons why particular design solutions have been accepted, rejected or even not completely evaluated. This requirement is supported by the inclusion of the relationships `Approved_Interactions` and `Reproved_Interactions` in the `Design_Solutions` class (Figure 7-10).
7.5. Interaction Testing Process

7.5.1. General Interaction Testing Process

The process of testing an interaction element consists of checking the validity of its condition. In the case of the simple interaction elements this process is done by comparing the reference value, or element, against the actual information retrieved from the product model or Product Range Model. In the case of the composite interactions this process is done by checking the set of interactions against the Boolean condition specified, i.e. AND or OR.

Figure 7-11 shows the UML Sequence Diagram for describing this general interaction testing process, where the states conditions and results for Interactions class are highlighted.

1. For each interaction associated with a particular Design_Solution, test the first interaction;
2. Compare the interaction condition;
3-5. If the result of the comparison respect the condition defined in the interaction, change Interaction_Status for Approved_Interaction; If the condition defined in the interaction, change Interaction_Status for Reproved_interactions; If the condition can not be tested, change the Interaction_Status to Not_Evaluated.
6. Return the Interaction_Status for the Design Solution;
   - Repeat the test interaction process until the last interaction.

Figure 7-11 – General interaction testing process – UML Sequence Diagram
In the figure above, each design solution applies the interaction testing process, represented by the method \textit{test\_interaction()}, to each interaction element associated with it. Although this method is part of the behaviour of the interaction element it is called by the design solution object, and consists of the comparison of the interaction internal condition and the determination of its state, which will be returned to the design solution.

\textbf{7.5.2. Simple Interactions Testing Process}

The testing interaction process for simple interactions consists of comparing the value retrieved from the product model, or Product Range Model, with the reference value, held by the simple interaction element. This comparison depends on the mathematical logical operator defined in the interaction. For instance, in the case of two interactions where the first one requires that the number of impressions must be greater than 4 (\texttt{Number\_of\_Impression > 4}) and the second one requires that the number of impressions must be equal 4 (\texttt{Number\_of\_Impression = 4}), the reference value is the same, but the process of comparison is different.

In the case of numerical interactions, the mathematical logical operators are defined as equal (==), greater than (>), smaller than (<), greater than or equal (>=), smaller than or equal (<=), and not equal (!=). For the case of existence interactions, these operators are reduced to equal (==) and not equal (!=), since only the existence of an element is being compared.

Figure 7-12 expands the representation depicted in Figure 7-11 for the simple interaction cases, where the retrieval of an actual value from the product model, realised by the Knowledge Link element, is highlighted.
7.5.3. Composite Interactions Testing Process

7.5.3.1. General Testing Process

The composite interactions are composed by two or more interaction elements, which can be either simple or composite. At a general level, the testing process for this kind of interactions is mainly concerned with the analysis of the testing process results of each associated interaction (Figure 7-13). Thus, in order to be considered approved, the results found must be in agreement with the Boolean condition of the composite interaction, i.e. OR or AND interaction types.
Chapter 7

7.5.3.2. OR Interaction Testing Process

The composite OR Interaction type is considered approved as soon as any of its associated interactions are approved. However, while an approved interaction state is not found, the testing process must be performed on the remaining interactions associated with the OR Interaction element. If all of its interactions have their states reproved, the composite OR Interaction will also be reproved.

Figure 7-14 depicts a graphical representation, based on the UML Activity Diagram, of the OR Interaction testing process.
7.5.3.3. **AND Interaction Testing Process**

In contrast to the OR interaction, the AND interaction type will be approved only if all of its interactions have their states approved. However, this type of interaction will be reproved as soon as any of its interactions is reproved. Figure 7-15 depicts a graphical representation of the AND Interaction testing process, based on the UML Activity Diagram.

---

**Figure 7-14 - OR Interaction testing process - UML Activity Diagram**

**Figure 7-15 - AND Interaction testing process - Activity Diagram**
7.5.3.4. Composite Interaction Testing Process Sequence

As previously mentioned, the composite interaction can be composed by either, simple or composite interactions. Composite interactions can be composed by other composite interactions, which in turn can be composed by other composite interactions, and so on.

To optimise the process of interaction testing in the case of composite interactions, the simple interactions are tested first. This allows eventual reproved conditions of the simple interactions to be detected before entering in the recursivity of composite interactions, and consequently saves processing the sequence.

This is valid for either, OR or AND interactions types. In the OR_Interaction type, as soon as an approved status is found the whole interaction is considered approved, with no need for further testing. In the case of AND_Interaction type, the opposite process applies, and as soon as a reproved status is found the whole interaction is considered reproved, with no need for further testing. Figure 7-16 depicts a UML Activity Diagram of this process described for the AND_Interaction type.

![Figure 7-16 - AND_Interaction testing process sequence- Activity Diagram](image-url)
However, one of the limitations of this testing process sequence is that it does not provide the designer with a more complete perspective of the reasons why a particular design solution has been rejected, e.g. has only one of the interactions been reproved or all of them have been reproved? Such information might eventually motivate the designer to change some product specifications in order to use alternative design solutions. Also, considering one of the major functions of the Product Range Model, which is to provide the designer with valid information, the application of such a testing process sequence should be balanced.

### 7.5.4. Evolution of the Interactions conditions

So far the interaction testing process results have been considered mainly under two possible interaction states namely, approved and reproved. For simple interaction elements the reason for these two states is straightforward, i.e. the condition held by the interaction has, or has not, been found. However, for the composite interaction, the determination of the final status of the interaction will depend on the state of each interaction which forms part of it.

In Figure 7-17 the final status of the OR type interaction, namely `Number_Impression OR Impression/Layout`, can only be known based on the states of the interactions, which are part of it, i.e. `Number_Impression = 5` and `Impression_Distribution == Circular`. Thus, irrespective of the level of recursivity that one composite interaction can have, its final status will be approved or reproved.
However, as previously discussed, in the case of non-evaluated interactions, the interaction element is not considered reproved, and the design solution associated with it can be offered as a valid design solution with pending interactions to be resolved. Considering the simplest case, a simple interaction can have its state as non-evaluated, and hence the design solution associated with it, will be, eventually, considered accepted if the other interactions have been approved (Figure 7-18). In this case the simple interaction, Runner_Type == Cold is directly interfering in the design solution status and consequently is easy to identify the reason for the non-evaluated state.

![Diagram of Non-Evaluated Interaction status](image)

Figure 7-18 - Non-Evaluated Interaction status

Similarly, for the case of composite interactions, the non-evaluated interactions are only considered if they are the only outstanding criteria delaying the full approval of the interaction (Figure 7-19). For the AND_Interaction type, if none of the interactions that compose it have been reproved, the non-evaluated interaction can be treated as approved, changing the final state of the AND_Interaction type to non-evaluated as well. For the OR_Interaction type, if none of the interactions that compose it have been approved, the non-evaluated interaction can be treated also as approved, changing the status of the OR interaction to non-evaluated as well.
Figure 7-19 - Non-Evaluated Interaction states propagation

The importance of the non-evaluated interaction propagation is in the provision of information related to pending aspects of the design decision that must be dealt with, in the case of choosing a not completely evaluated design solution. This process can be particularly helpful in the initial stages of the design, where the general specification of the injection mould may not be precise.

7.5.5. Reassessing Interactions of DS already Chosen

One of the functions of the Product Range Model is deal with temporary design solutions selected by the designer. While these temporary design solutions are not stored in the product model, they are the "responsibility" of the Product Range Model and hence must be constantly monitored against their interactions. For instance, if the product information relating to the specifications or any other design decisions has been changed, a new reassessment should be made in order to check the validity condition of such design solutions.

This situation can be extrapolated to design solutions that have been chosen through the Product Range Model and stored in the product model. However, in this case, such functionality is more related with the software application that will make use of this information, than to the Product Range Model itself.
7.6. Summary

This chapter has explored how the knowledge that defines the validity of each injection mould design solutions can be modelled through the element termed Interactions. This has been defined as one of the main contribution of this research. A definition and classification of these interactions has been provided, as well as how they can support the design decision process. Although the types of interactions defined in this work are not novel, the power of the approach presented is in how these interactions have been modelled as individual objects. The need for a mechanism that can retrieve information from the product model has been stated and will be discussed in the next chapter.
8. Product Model and Product Range Model Relationships

8.1. Introduction

This chapter explores the information relationships between the Product Range Model and Product Model, which are required to support the design decision making process addressed in this work, defining another area of contribution of this research. Two kinds of relationships have been identified between these information models and are presented in section 8.2. Section 8.3 presents a brief description of an injection moulding Product Model defined for this work. Section 8.4 discusses issues related to how product information supports the interaction testing process, through the Knowledge Link elements, as well as the information structure to represent them. Finally, section 8.5 highlights the need for information compatibility between both information models to enable design results from the Product Range Model to be captured in the Product Model.

This chapter completes the discussion of ideas developed in this thesis and together with chapters 6 and 7 defines the Product Range Model information structure, which is used as part of the experimental system developed in chapter 9.

8.2. General Relationships between Product Range and Product Information

Figure 8-1 highlights the two kinds of relationships, identified in this work, between the Product Range Model and Product Model.

The first one is related to how pieces of product information can be retrieved from the Product Model to support the comparison performed during the interaction testing process of simple interaction elements, and is identified as another important point of contribution of this research.

The second kind of relationship is related to how the design information selected from the Product Range Model, i.e. design solutions, can be stored in the Product Model and hence become part of the product information.
The relationships addressed above are significantly dependent on the Product Model information structure as the Product Model is responsible for both, the capture of information related to a product life cycle and making this information available for sharing by different applications that require it.

![Diagram showing relationships between information models](image)

Figure 8-1 - Type of relationships between information models

The next section presents a general description of the Product Model defined in this research work, and the relationships are discussed in sections 8.4 and 8.5 of this chapter.

### 8.3. The Product Model Structure

#### 8.3.1. General Aspects

This work has focused on the information structure required to represent the Product Range Model elements. However, in order to explore the extent to which the concepts behind the Product Range Model can be applied, the definition of a minimum information structure for the Product Model was required. Thus, an explicit injection moulding Product Model structure, composed of injection mould and plastic component types of product, has been defined.

Figure 8-2 depicts a basic representation of the injection moulding Product Model defined, where the association between injection mould product and plastic
component product is highlighted. This association has been defined as one to one bi-directional, which means that each injection mould product can have only one plastic component associated with it, and vice-versa.

Figure 8-2 - General Injection Moulding Product Model

8.3.2. Injection Moulding Product Model Information Structure

8.3.2.1. Product Specification Structure

The product specification represents the initial decisions that are made about the injection moulding products and have been discussed in Chapter 5. For example information about feeding point location, parting line location and projected area are some of the plastic component specifications, whilst the number of impressions, type of degating, type of runner and mould plate configuration are some of the injection mould specifications.

Figure 8-3 expands Figure 8-2 highlighting some of the attributes that represent the specifications related to the injection moulding products. These pieces of information drive a significant part of the design solutions interaction elements identified in this work, as addressed in Chapter 5.

Figure 8-3 - Injection moulding product specifications
8.3.2.2. Design Solutions Structure

The design decisions information structure captures the decisions made by the
designer about the injection mould, and in this work these decisions have been mainly
related to the design solutions. These design solutions are represented in the Product
Model by the class termed Solutions_Techniques, and are associated with the classes
Injection_Mould, Mould_Plate, Injection_Mould_System and Technique_Functions
(Figure 8-4).

Five sub-types of Solutions_Techniques class, which represent the design solutions
associated with the main injection mould system explored in this work, have been
defined, namely, Cooling_Solutions, Ejection_Solutions, Runner_Solutions,
Gate_Solutions and Impression_Layout_Solutions.

A more detailed representation of the Product Model information structure, defined in
this work, is depicted in Appendix C.
8.4. Knowledge-Links to enable Information Model Relationships

8.4.1. General Aspects

The capture of the knowledge that represents the design criteria for application of each design solution, within a design situation, was highlighted in Chapter 7, where the definition of interaction elements was discussed.

Particularly in the case of simple interactions, the testing process is performed against the actual product information, which must be retrieved from the Product Model. The information required can be the value of a specific attribute, in the case of numerical interaction, or the verification of the existence of a particular element, in the case of the existence interaction. Thus, for the retrieval of any information from the Product Model, it is necessary to know, besides its information structure, the specific instance that holds the required information.

For example, if the information about type of mould configuration is required, it is necessary to know also from which injection mould instance such information must be retrieved, as more than one injection mould instance is likely to be associated with the class Injection_Mould.

To enable this information retrieving process and, hence to support the interaction testing process another element constituent of the Product Range Model, termed Knowledge Link, has been defined in this work.

8.4.2. Knowledge-Link Elements Definition

The Knowledge-Link elements store the information and knowledge about specific pieces of the product data model, providing the means to retrieve the value of the correct information for comparison by the simple interaction elements. These elements define a contribution of this research in identifying the relationships between the Product Range Model and the Product Model.

The creation of the Knowledge-Link elements provides an interface between the interaction elements, in the Product Range Model, and the Product Model, which provides independence and flexibility between both information models. This is because the information stored in the interaction elements becomes more independent of the Product Model information structure.
The Knowledge Link elements also provide reusability of these product data model paths, since different interaction elements, which compare information related to the same class attribute, can make use of the same defined Knowledge Link element.

The association between simple interactions and the Knowledge Link element was introduced in Chapter 7. This association has been defined as a many to one type, where each simple interaction is associated with only one Knowledge-Link element, but each Knowledge-Link element can be used by one or more simple interaction elements (Figure 8-5). For example, when dealing with the number of impressions, different interaction elements can be defined for capturing parts of the different design criteria from different design solutions. However all of them will require the same information from the Product Model, i.e. number of impression, in order to perform the final comparison, and hence they will make use of the same Knowledge-Link element.

![Figure 8-5 - Relationship between Simple Interaction and Knowledge-Link](image)

8.4.3. Retrieving Product Information from Product Model

8.4.3.1. Information Retrieval

As addressed in the previous section, the aim of the Knowledge Link is to provide the means to return the correct information value, from the Product Model, to the simple interaction elements. This information can be related to a specific number, for the case of the numerical interactions, or could be the result of the search for the specific element, in the case of existence interactions.

Figure 8-6 shows an example of both the situations mentioned above. In the numerical interaction case, a number is retrieved from a particular attribute of an object within the Product Model, e.g. *Number of Impression*, from a particular Injection Mould
object. This number is returned to the interaction element, where the final comparison is performed.

In contrast, in the existence interaction case, the existence of the object in the Product Model is verified, by the comparison of all objects related to a specific class from which the object is derived. In this case, the attribute is represented by the identification of the object. The returned result is the existence, or absence, of such object, which will be compared by the interaction element.

Figure 8-6 - Retrieving information from the Product Model

As a result, two elements are necessary for retrieving the product information in the Knowledge-Link definition, and these are related to the location of the class of object searched and the specific attribute required.

8.4.3.2. Information Location

To perform the information retrieving process presented above, the correct object needs to be located. The process of locating specific information about a particular object within an object-oriented information structure can be very complex, as there are different kinds of relationships between classes objects, i.e. inheritance, aggregation, association.

In the case of numerical interaction, as presented in the above example (Figure 8-6), the process of locating the information was straightforward, since the attribute *Number of Impression* is part of the root class of the product data model, i.e.
Injection_Mould. However, for the existence interaction example, the product hierarchy structure has to be followed in order to find the right object (Figure 8-7).

Figure 8-7 - Information structure path in Knowledge-Links

In the case above, the process of locating information consists of identifying the sequence of objects that may provide the correct information. For instance, to retrieve any information from the injection mould object, the specific injection mould that is being designed must be selected. Also, to retrieve any dimensional information from the plastic component, in addition to the actual plastic component in design, the specific geometric characteristic associated with such a dimension is required. As result, a "data model path" to locate and achieve the information required by an interaction element must also be considered in the definition of the Knowledge-Link element.

This work has defined this "data model path" as part of the Knowledge-Link information, through a sequence of class names. Besides detailed information about the product data model classes, such an approach demands a strong understanding of the knowledge related to how these objects are searched, and this is also included in the behaviour of the class Knowledge-Link. For example, the associations, aggregation and inheritance relationships between the Product Model objects requires different ways of searching for the information. These ways, in turn, are significantly dependent of the technology chosen for the computational implementation, e.g. relational database, object-oriented database, or alternative file structure, and even the
supplier of the technology used, resulting in a great variation in its implementation from case to case.

8.4.3.3. Knowledge-Link Information Structure

Figure 8-8 expands the diagram depicted in Figure 8-5 highlighting the attributes defined for the Knowledge-Link element. The attribute Knowledge_ID provides the individuality of each Knowledge Link element and the Data_Member_Name defines which attribute holds the value of the information that must be retrieved from the object. The attribute Main_Class defines the class that holds all data structure related with the information required. For example, in this work two main kind of products have been used, the injection mould and the plastic component, which hold all information associated with them. The attributes Sub_Class_xx define the path sequence for locating the required class. The experimental implementation tested these concepts using five (5) class levels as shown in Figure 8-8.

![Diagram](image)

**Figure 8-8 - Knowledge-Link class attributes**

Figure 8-9 depicts a general representation, based on the UML Activity diagram, of the general process of locating and retrieving the attribute value through the Knowledge Link element. The process starts with the selection of the main class, e.g. injection mould or plastic component, and is followed by the selection of the subclasses. As the last valid sub-class is identified the value of the data member variable is retrieved and returned to the interaction element.
Figure 8-9 - General Knowledge-Link information retrieve process

Figure 8-10 and Figure 8-11 depict examples of how Knowledge-Link advances through the Product Model information structure. In Figure 8-10 only one class has been defined in the Knowledge-Link element, i.e. Injection_Mould. Thus, the value of the attribute defined in the Data_Member_Name, i.e. Number_of_Impression is retrieved from this class, and returned to the interaction element.

In Figure 8-11, in addition to the main class, another class has been defined, named Solutions_Techniques. However, Solutions_Techniques is a parent class in the Product Model information structure defined (Figure 8-4), and hence the specific type of class object that is required to be compared is provided by the information of the existence interaction element, which in this case is related to the Gate_Solutions. Only after achieving this last class, i.e. Gate_Solutions, can the value of the attribute defined in the Data_Member_Name, i.e. Technique_ID, be retrieved and returned to the interaction element in the case of numerical interaction, or compared, in the case of existence interaction.

These simple examples illustrate how complex the knowledge necessary to represent the behaviour of the Knowledge Link elements can be.
8.5. Requirements for Storing Design Decision Information in the Product Model

8.5.1. General Aspects

Based on a functional enquiry, the Product Range Model offers valid information options to the designer, who will make the final choices. Thus, the information chosen will be part of the Product Model and for this reason, some level of compatibility, between the Product Range Model and Product Model, is required to enable this information transferring process.
For example, if an ejection pins design solution has been selected by the designer, from the Product Range Model, as a final decision about how to eject the plastic component, the information related to such a design solution should be stored in a compatible class in the Product Model. This will allow any additional enquiries about this design solution to be properly retrieved from the Product Model in addition to complementary information that may need to be developed about ejection pins, in the subsequent phases of the design.

This research has focused on the general/conceptual definition of the design solutions in the Product Range Model and for this reason the compatibility with the Product Model solution techniques has looked at the level of class identification attributes. Thus, each design solution defined in the Product Range Model keeps the link with its respective solution techniques class structure in the Product Model, enabling the initial information to be stored in the right place in the Product Model whenever it is chosen. This link is kept through the name of the Solution_Techniques class in the product data model. In this research the design solution name and identification, along with the association with the function, have been identified as the compatible information in both information models.

8.5.2. Product Data Information Relationships

Figure 8-12 depicts a general representation of the Product Range Model and Product Model information structures, highlighting the required compatibility between the Design Solutions and Functions in the Product Range Model and the Solution Techniques and Technique Functions in the Product Model.

In the Product Range Model the design solutions instances are derived from the same class, e.g. Ejection_DS. However, in the Product Model these instances can be derived from different classes, e.g. Ejection_Pins, Stripper_Plates, Ejection_Valves, etc. (Figure 8-13).

To support this relationship, each design solution defined in the Product Range Model has an additional attribute that defines where such design solutions must be created in the Product Model, if they are chosen. This attribute has been defined as PM_Class_Name.
Figure 8-12 - Information compatibility between Product Range Model and Product Model

Figure 8-13 - Relationships between PRM Design_Solution instance and Product Model Solution_Technique classes
8.6. Summary

This chapter has identified the need for structured information model relationships, between the Product Range Model and the Product Model, to support design decisions. Two different types of relationships have been identified, which are concerned with both the interaction testing processes and the transfer of information from the Product Range Model and Product Model. Knowledge-Link elements and the relationship between Product Range Model design solutions and Product Model classes have been proposed to overcome these issues. Such relationships are closely related to the data management field, identifying a potential area for further developments for application of additional information models, such as the case of the Product Range Model.
9. Experimental System Implementation

9.1. Introduction

This chapter describes the experiments conducted to explore the representation of the Product Range Model and to demonstrate the extent to which it can provide support to the design process by offering valid design information. Section 9.2 presents the objectives and scope of the experimental software system developed as well as a description of the general aspects considered for its design and implementation. Section 9.3 presents the exploitation of the Product Range Model structure, where the associations between injection functions, design solutions, interactions and knowledge-Links are demonstrated. Section 9.4 focuses on a functional design application to support the design process and make use of the Product Range Model, and is followed by the summary of the chapter.

9.2. Design and Implementation of the Experimental System

9.2.1. Objectives and Scope

The experiments conducted during this work have been focused on the exploration and definition of the Product Range Model structure, and how this information model can provide support for the design decision process.

The main objectives of the experimental system are listed as:

I.) To explore the information structure defined to the Product Range Model, which is composed by the functions, design solutions, Interactions and Knowledge-Links class elements;

II.) To explore the relationships between the class elements, addressed in the objective I.), in the Product Range Model;

III.) To explore how the Product Range Model, in combination with the product model, can provide support for design decisions, through a computational application.
How objectives I) and II) are realised is described in section 9.3, where an injection mould Product Range Model is demonstrated with some examples of actual information. The realisation of the objective III) is described in section 9.4, where a software application to support functional design has been developed. While the realisation of the first two objectives are related to static aspects of the Product Range Model, the last one is related to the dynamic aspects of the Product Range Model, i.e. how interactions are evaluated and valid design solutions are offered to the designer.

To test the ideas proposed in this work a set of information and knowledge related to injection mould product design has been defined for implementation purposes. Because of the practicalities of the implementation a basic set of functions and their associated design solutions was selected, which were related to the main systems of the injection mould, as well as a set of interactions that take place between the design solutions defined and the product information. This information has been captured mainly from technical literature and from the industrial collaborator, and demonstrates the concepts of the research comprehensively.

9.2.2. Experimental System Environment and Description

Chapter 3 has presented the tools applied to support the design and implementation of the experimental system. Following the MOSES approach, two main elements were pursued in the implementation, namely the information structure, which has been realised by the ObjectStore database, and the software applications, which have been realised by the programming environment Visual C++.

The design of the system has been supported by the use of UML notation and was initiated by the definition of the main functionality of the system. This definition has been used throughout the concept of Use Cases. Based on the Use Cases defined and the general description of their scenarios, the main categories of the system have been identified, i.e. functions, design solutions, Interactions and Knowledge-Links. These categories are part of a major category called Product Range Model and have been used for the development and detailing of the information structure achieved, which has been presented in Chapter 7 and Chapter 8. General relationships have been constructed between both Use Cases and categories guiding the dynamic aspects of the system, i.e. the behaviour of each object.
Appendix B presents the Requirement Traceability Matrix developed in the initial phases of analysis and design of the experimental system and describes the development process applied, using a Use Case as an example. Appendix C shows the UML class representation for the experimental system developed.

Two main groups of functions related with the Product Range Model have been identified in the experimental system development (Figure 9-1). The first one, named PRM Application Interface, is concerned with aspects related to the population and management of the Product Range Model. The second one, named DFF Application or Injection Mould Support System is related to how the information stored in the Product Range Model can effectively support the design process and relate to the product model. Such functions have guided the general structure of the experimental system, depicted in Figure 9-1.

For the development of the experimental system the main functionality of the PRM has been captured in terms of it providing design support through valid information. Thus, the choice of which injection mould functions to check for potential design solutions was left to the user. No restriction has been put in relation to the design process sequence. This approach provides more flexibility in terms of enquiring the Product Range Model, being independent of an injection mould formal design sequence process.
9.2.3. Visualisation of the Results

The visualisation of the results has been realised in two ways: I) by the output dialogues of the experimental system application developed, and II) by the Object Inspector interface, which allows visualising the data stored in the ObjectStore database structure (Appendix A).

9.2.4. Injection Mould Information

The definition of the information stored in the Injection Mould Product Range Model has been based initially on the general bibliography in the area (Pye 1989; Menges and Mohren 1993; Rees 1995; Rosato and Rosato 1995). This provided an initial set of information to explore some of the basic concepts of the Product Range Model. Part of this information has been discussed in chapter 5. In order to ground this set of information upon some more practical aspects, discussions about real case studies have been made with the industrial collaborator, i.e. Moss Plastic Parts. These discussions provided a more realistic "flavour" for the definition of the experimental system, as well as, the capture of a more focused set of information.

Thus, from the set of information initially collected, a more restricted set of injection mould design solutions has been defined in conjunction with the industrial collaborator engineers. This set of information represents some of the most common design solutions applied in the company, as well as part of the knowledge required to their application.

This information and knowledge have been used in this research work to support the experiments in two ways. First providing actual information to be stored in the Product Range Model, and hence evaluating how the information structure defined can capture design information and knowledge. This is explored in section 9.3, and the results are presented in the Appendix D. Second, once this information and knowledge are stored in the Product Range Model, they can be used to evaluate how the Product Range Model can provide support to design reuse. This is explored in section 9.4, through the combined use of the Product Range Model and Product Model developed.

A significant part of the collected information was focused on the main specification of the mould and how it influences the impression distribution and feeding system.
design. This was because such design solutions are common for all moulds and less dependent on the detailed aspects related to the plastic component geometry. Thus, as a result, the set of interactions used in the experimental system has been mainly based on these kinds of design solutions.

Different plastic components have been used as basis to the discussions with the industrial collaborator to gather information about injection mould functions, design solutions and interaction elements. From these, two components have been focused on for the experimental implementation (Figure 9-2), due to their properties and influences on the mould configuration, which allowed to explore different injection mould functions, design solution results and consequently interaction elements.

Figure 9-2 - Plastic components used for exploring Product Range Model information

One of the main outputs of the process of defining/selling a plastic component product, is a specification sheet of the mould, named "Feasibility Review Sheet", which is used as specifications to the design of the mould. This sheet has been used to capture the initial specification of the mould. A representation of the significant

- 146 -
information captured within this is depicted in Figure 9-3. These pieces of
information are complemented in the initial stages of design with other information,
not previewed in the feasibility sheet, such as mould configuration and parting line,
for example. This information has been defined, therefore, as the main initial
interactions modelled in the experimental system.

<table>
<thead>
<tr>
<th>FEASIBILITY REVIEW SHEET</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GENERAL DATA</strong></td>
</tr>
<tr>
<td>Enquiry number</td>
</tr>
<tr>
<td>Mould number</td>
</tr>
<tr>
<td>WOT number</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Dimensions</td>
</tr>
<tr>
<td>Drawing supplied ?</td>
</tr>
<tr>
<td>Originator</td>
</tr>
<tr>
<td>Run Location Plant 1</td>
</tr>
<tr>
<td>Plant 2</td>
</tr>
<tr>
<td>Plant 3</td>
</tr>
<tr>
<td>Annual Volume</td>
</tr>
<tr>
<td>Existing part or similar available ?</td>
</tr>
<tr>
<td>Tolerances</td>
</tr>
<tr>
<td>Drawing number &amp; Issue</td>
</tr>
</tbody>
</table>

| **MATERIAL**             |
| Type                     |
| Material colour (s)      |
| Grade                    |
| Specification            |
| Material / masterbatch code |
| Volume                   |

| **TECHNICAL INFORMATION** |
| PROTOTYPE OR PRODUCTION   |
| Number of cavities        |
| Est. shot weight          |
| Est. cycle time           |
| Machine design constraints|
| Sprue picker              |
| Machine type              |
| Critical areas            |
| Ancillary equipment       |
| Manifold YES NO          |
| Controller YES NO         |
| Feed point location       |
| Mould base YES NO         |
| Tool temp YES NO          |
| Spec. hopper YES NO       |
| Hot Runner                |
| Clamp force required      |
| Surface finish            |
| Logo required Material Recyclable |
| Load details              |
| In use temperatures       |

Figure 9-3 - Feasibility Review Sheet
9.3. Product Range Model Structure

9.3.1. General Description of the Experiment

This experiment aims to confirm that the information structure defined to the Product Range Model can support the representation information and knowledge through injection mould functions, design solutions, interaction elements and knowledge link elements, i.e. objectives I) defined in section 9.2.1.

In addition to these representations, this experiment explores also the relationships between injection mould functions and sub-functions; functions and design solutions; design solutions and interactions elements; composite and simple interaction elements; and simple interactions and knowledge link elements, i.e. objectives II) defined in section 9.2.1.

The relationships that support the reusability of the interaction elements by design solutions and composite interactions in different scenarios, and the reusability of the knowledge link elements by different interaction elements is also explored.

As addressed in section 9.2.4, the information used in this experiment has been collected from technical literature, discussed in chapter 5, and complemented by discussions with the industrial collaborator.

The results of this experiment are presented in the Appendix D, which contains the information stored in the Product Range Model in terms of function, design solutions, interactions and knowledge links, as well as the relationships between them.

9.3.2. Injection Mould Functions

Chapter 6 has addressed the information structure required to capture injection mould functions in the Product Range Model. This sub-section explores how instances of injection mould functions can be stored in such an information structure. Feed Function class is used as an example to explore how the structure has been realised.

Figure 9-4 shows a window of the injection mould functions instances stored in the Product Range Model. The functions on the left represent the five main root functions of the injection mould explored in this work. Moving from the left to right, the functions are decomposed in more specific sub-function, which in turn are also
decomposed in other sub-functions. The Function Level specifies the level of the functional decomposition. Each function is an object and hence holds its own information.

Figure 9-4 - Functional structure instances (ObjectStore Inspector)

Figure 9-5 details the contents of the previous figure, highlighting the Feed Function instance and its related sub-functions. Thus, the Feed Function is decomposed in three sub-functions named Feed Impression, Feed Mould and Distribute Feeding. This latter sub-function is still decomposed in two other sub-functions called Distribute Runner Layout and Control Runner Flow.

Figure 9-5 - Feed Function instanciation functional tree (ObjectStore Inspector)
Figure 9-6 shows the properties, attributes and relationships, of a specific sub-function instance, i.e. Distribute Feeding. The following attributes (*), and their values, are presented: Function_ID, Function_Status, Function_Name, Function_Level and Function_Type. The three main relationships with this sub-function instance are highlighted. The first two relationships are related with the sub-functions collection (Sub-Function_Coll) and the parent function (Function_Root). The latter one is related with all possible design solutions that can be applied to fulfil this function (Design_Solutions_Coll).

Figure 9-6 - Sub-Function instance Distribute Feeding (ObjectStore Inspector)

Figure 9-7 shows a representation of the relationship between the function previously described, i.e. Distribute Feeding and some of the design solutions associated with it. Because this is the parent function of two other sub-functions, namely Distribute_Runner_Layout and Control_Runner_Vlow (Figure 9-5), it encompasses all design solutions associated with both sub-functions. However, each of these later sub-functions will maintain an association with its particular sub-sets of these design solutions, i.e. Cross_Section_Runner_DS and Layout_Runner_DS (represented in the top of each design solution instance in Figure 9-7).
Chapter 9

9.3.3. Injection Mould Design Solutions

The design solutions class structure has been designed based on the main systems of the injection mould, i.e. ejection, cooling, feeding, venting and impression discussed in Chapter 5 and Chapter 6.

Figure 9-8 shows sets of ejection and runner layout design solutions instances.

---

Figure 9-7 - Design solutions associated with *Distribute Feeding* function  
(ObjectStore Inspector)

---

Figure 9-8 - Ejection and Runner Layout Design solution instances (ObjectStore Inspector)

- 151 -
Figure 9-9 shows the attributes and relationships of a design solution instance, named *Rectangular_Y_Layout*. This design solution was chosen to maintain the "compatibility" with the function instances explored in the previous sub-section. The following design solution attributes are depicted *DS_Name*, *DS_ID*, *PM_Class_Name* and *DS_Status*. Besides the relationship with the standard solutions, in this case *NULL*, the design solutions keep other relationships with the injection mould functions, represented by *Function_Coll*, and with the interaction elements, represented by *Interaction_Collection*. The relationships with approved and reproved interactions, i.e. *Approved_Interactions* and *Reproved_Interactions*, are used as elements to support the decision process during the design process and store sets of approved and reproved interactions, respectively. These relationships are discussed in section 9.4.

![Diagram of design solution instance](image)

Figure 9-9 - *Rectangular Y Layout* design solution instance (ObjectStore Inspector)

Figure 9-10 depicts an explicit representation of interactions and functions relationships with this particular design solution. Particularly for the relationship with functions, it can be noted that the two functions associated with this design solution, i.e. *Distribute_Feeding* and *Distribute_Runner/Layout*, are parent and sub-function...
respectively, showing how different levels of functions decomposition can be associated with the same design solution.

Figure 9-10 - Rectangular Y-Layout design solution associated with its functions and interactions (ObjectStore Inspector)

Figure 9-11 expands the ejection design solutions, highlighting their association with examples of commercial standard solutions available. Besides the name of the commercial standard solutions, are also shown their code, supplier and main dimensions. Figure 9-12 shows the navigation tree to an ejection design solution instance, named Normal_Ejection_Pins, where two kind of commercial standard solutions are available in different dimensions.

Figure 9-11 - Ejection standard design solutions (ObjectStore Inspector)
Chapter 9

9.3.4. Injection Mould Interaction Elements

The interaction elements define the knowledge required to represent the design criteria of each design solution. Figure 9-13 shows part of the interactions instances defined in the experimental system. The interactions are represented by their name, Interaction_Name, ID, Interaction_ID and type, Interaction_Type, and are mainly associated with the design solutions. Depending on the type of the interactions, i.e. numerical, existence, OR or AND interactions types, additional attributes and relationships are defined. Each interaction instance has been defined in this work as an individual object, and hence can be used by one or more design solutions and/or also be part of another interaction, i.e. composite interaction.

Figure 9-14 depicts an example of an interaction instance Impression = 1, which is associated with two design solution, One Impression Layout and Sprue Gate, and with the composite interactions, named Mould Configuration = 2-P && Impression = 1 and Impression = 1 && Feeding_Point == P/L. These relationships are highlighted in Figure 9-15.

Specifically in the case shown in Figure 9-14, because such an interaction is a simple interaction type, the relationship with the Knowledge-Link, named Number_of_Impression is also presented.
Figure 9-13 - Injection mould general interactions (ObjectStore Inspector)

Figure 9-14 - Interaction instance Impression = 1 (ObjectStore Inspector)
9.3.4.1. Simple Interactions

Figure 9-16 depicts some of the simple interactions instances stored in the Product Range Model. A total of 77 simple interactions have been included in the experimental system, being 26 existence interaction type and 51 numerical interaction type (Appendix D - Figure D 8 and Figure D 9). Besides the Interaction_Name and Interaction_Type, the attribute Comparator and the relationship with the Knowledge Link Data_Member_Name are also displayed. As already explained in Chapter 7, the Comparator attribute holds the code of the mathematical relational operators for the comparison process, e.g. 1 means equal; 3 means larger or equal; 6 means not equal, etc.. The Knowledge_Link stores the data model address from where the information value to be compared with the simple interaction is retrieved.
Figure 9-17 shows an instance of a numerical interaction related to the mould configuration, i.e. *Mould_Configuration* = 2-P. Besides being part of an AND composite interaction type, this interaction element is also used by different design solutions, which require this mould configuration, in addition with other interactions, for their validity. An additional attribute for numerical interaction instances is also displayed. The *Reference_Value* for this type mould configuration (enumeration type), i.e. 2-Plates, is "1", which must be compared with the actual value retrieved from the product model, through the Knowledge-Link instance named *Mould_Configuration*.

Figure 9-17 - Numerical interaction instance *Mould_Configuration* = 2-P
(ObjectStore Inspector)

Figure 9-18 depicts an instance of an existence interaction related to the type of runner layout. The main difference of this example in relation to the previous figure is related with the attributes *Element_Type* and *Element_Name*, which refer to an object that is being compared, rather than to a number.
9.3.4.2. Composite Interactions

Figure 9-19 depicts a set of composite interactions defined in the Product Range Model. Two types of composite interactions have been defined in the work namely AND and OR types. Such differentiation is provided by the attribute Interaction_Type. Either design solutions or another composite interaction can use the composite interactions. A total of 44 composite interactions have been included in the Product Range Model, being 20 OR composite interaction and 22 AND composite interaction type (Appendix D - Figure D 10 and Figure D 11).

Figure 9-20 shows an instance of an OR composite interaction type, named Impressions = 4, 6 or 8. Different to the simple interactions, these kinds of interactions do not have specific comparison attributes or relationship with the Knowledge-Links. The main characteristics of this kind of interaction are the type of Boolean comparison that they perform and the association to a set of interactions that are part of it. In the example depicted, the composite interaction is part of the design criteria that define the eligibility of an impression distribution type of design solution and is composed by three simple interactions (Interactions_Set), namely Impressions = 4, Impressions = 6 and Impressions = 8. These relationships are highlighted in Figure 9-21.
Figure 9-19 - Composite interaction instances (ObjectStore Inspector)

<table>
<thead>
<tr>
<th>Interaction Name</th>
<th>Interaction ID</th>
<th>Interaction Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Impressions = 3, 5, or &gt;=7)</td>
<td>51000</td>
<td>4</td>
</tr>
<tr>
<td>2 Runner * Circular</td>
<td>51001</td>
<td>4</td>
</tr>
<tr>
<td>3 (Impressions = 3, 5, or &gt;=7) OR (Runner = Circular)</td>
<td>51002</td>
<td>4</td>
</tr>
<tr>
<td>4 Impressions &gt;=2 &amp;&amp; &lt;=8</td>
<td>52000</td>
<td>3</td>
</tr>
<tr>
<td>5 Runner = In_Line</td>
<td>52001</td>
<td>4</td>
</tr>
<tr>
<td>6 Impressions = 4, 6 or 8</td>
<td>55000</td>
<td>4</td>
</tr>
<tr>
<td>7 Runner # Circular</td>
<td>55001</td>
<td>4</td>
</tr>
<tr>
<td>8 Ejection # Stripper</td>
<td>40000</td>
<td>4</td>
</tr>
<tr>
<td>9 Mould_Config = 2-P &amp;&amp; Impression= 1</td>
<td>23000</td>
<td>3</td>
</tr>
<tr>
<td>10 Impressions &gt;=3 &amp;&amp; &lt;=7</td>
<td>30000</td>
<td>3</td>
</tr>
<tr>
<td>11 Impressions = 6,8 or 10</td>
<td>31000</td>
<td>4</td>
</tr>
<tr>
<td>12 Box Type Component &amp;&amp; Lenght/Width Ratio &gt;= 3.5</td>
<td>33000</td>
<td>3</td>
</tr>
<tr>
<td>13 Impressions = 4 or 8</td>
<td>36000</td>
<td>4</td>
</tr>
<tr>
<td>14 Impressions &gt;=2 &amp;&amp; &lt;=8</td>
<td>35000</td>
<td>3</td>
</tr>
<tr>
<td>15 Mould_Config = 3-P &amp;&amp; Impressions &gt;=2</td>
<td>23001</td>
<td>3</td>
</tr>
<tr>
<td>16 (2-P &amp;&amp; 1 Impression) OR (3-P &amp;&amp; Impressions&gt;=2)</td>
<td>23002</td>
<td>4</td>
</tr>
<tr>
<td>17 Runner Half Section = Movel Plate</td>
<td>15000</td>
<td>4</td>
</tr>
<tr>
<td>18 Impression = 1 &amp; Feeding Point = P/L</td>
<td>12000</td>
<td>3</td>
</tr>
<tr>
<td>19 (Impression = 1 &amp;&amp; Feeding Point = P/L) OR Impression &gt;=2</td>
<td>12001</td>
<td>4</td>
</tr>
<tr>
<td>20 Runner Half Section = Fixed Plate</td>
<td>14000</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 9-20 - Composite interaction instance Impressions = 4, 6 or 8 (ObjectStore Inspector)
Figure 9-21 - Example of the composite interactions relationships with other interactions and design solutions (ObjectStore Inspector)

Figure 9-22 shows another type of composite interaction, in this case an AND composite interaction type, which is composed by two interactions shown in the attribute Interactions_Set. In this example, the interaction is not associated with any design solution, but is only associated with other composite interaction, which is shown by the attribute Composite_Interaction_Coll. Figure 9-23 shows the relationships previously addressed.

Figure 9-22 - Composite interaction instance Impressions = 1 && Feeding_Point == P/L (ObjectStore Inspector)
Figure 9-23 - Example of the composite interactions relationships with other interactions (ObjectStore Inspector)

The representation of the interactions either composite or simple, described previously, shows how a particular interaction instance can be applied to different situations, i.e. different design solutions and/or other interactions. This highlights the potential of the approach taken together with the power of object oriented technology, in terms of reusing the knowledge already defined. However, to allow this, some sort of functionality must be provided by the application responsible for the management of this base of interactions.

Figure 9-24 shows a sequence of dialogues illustrating the process of adding a new interaction to a particular design solution. Particularly in this example, the option for using an interaction already created is made, however if no interaction satisfies the piece of knowledge required, a new interaction can be created. This same process can be used also for adding a new interaction into a composite interaction.
Figure 9-24 - Inserting an interaction already defined
Chapter 9

9.3.5. Knowledge-Links

The Knowledge-Links is the last element in the Product Range Model structure. It provides simple interactions with the actual information extracted from the product model.

Figure 9-25 shows some of the Knowledge-Link instances stored in the Product Range Model. Each Knowledge-Link has an unique ID (Knowledge_ID), which is responsible for its identification. The Main_Class determines the top class defined in the product model, in this case which product holds the information that is requested, i.e. injection mould or plastic component. The Sub_Class_01 is the following class defined in the product data model and related with the Main_Class. Finally the Data_Member_Name captures the data member of the final object that holds the information required.

In the example depicted in Figure 9-25, most of the Data_Member_Name are attributes of the injection mould and plastic component, and represent mainly the product specification.

<table>
<thead>
<tr>
<th>Main Class</th>
<th>Sub Class 01</th>
<th>Data Member Name</th>
<th>Knowledge ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injection_Mould</td>
<td>Number_of_Impression</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Injection_Mould</td>
<td>Mould_Configuration</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Injection_Mould</td>
<td>Degating_Type</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Injection_Mould</td>
<td>Estimate_Shot_Weight</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Injection_Mould</td>
<td>Type_of_Runner</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>Component_Volume</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>Component_Projected_Area</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>Feeding_Point</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>Component_Shot_Weight</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>General_Shape</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Injection_Mould</td>
<td>Solutions_Technique</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Injection_Mould</td>
<td>Technique_ID</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>Generic_Geometry</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>Geometry_Type</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>L_W_Ratio</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Plastic_Component</td>
<td>P_L_Position</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9-25 - Knowledge-Link instances (ObjectStore Inspector)

Figure 9-26 depicts an instance of the Knowledge-Links, named Number_of_Impression. Besides the attributes previously explained, it is focused on the association with several simple interactions, in this case, numerical interactions. This relationship is highlighted in Figure 9-27.
Figure 9-26 - Knowledge-Link instance *Number_of_Impression* (ObjectStore Inspector)

![Knowledge-Link instance diagram](image)

Figure 9-27 - Knowledge-Link relationships with Simple Interactions (ObjectStore Inspector)

![Knowledge-Link relationships diagram](image)
9.4. Product Range Model Supporting Functional Design Through the Reuse of Information and Knowledge

9.4.1. General description of the experiment

This experiment aims to explore how the Product Range Model, together with the Product Model, can support designer decisions, i.e. objective III) defined in section 9.2.1. This is realised through the reuse of previous information and knowledge stored in the Product Range Model, i.e. functions, design solutions, interactions and knowledge links. This information is presented in Appendix D.

An Injection Mould DFF application explores how the user can interface with the Product Model to define the basic specification of the injection mould and how he/she can evaluate design solution results from his/her functional enquiry to the Product Range Model (section 9.4.2).

This experiment explores how the number of valid design solution options can vary based on the level of specification of the injection mould, and how interaction elements can support the designer by justifying the reasons for design solution states (section 9.4.3). It also explores how non-evaluated interactions can support the designer highlighting open specifications, which must be resolved through the choices made. In addition to product specifications, it also explores, in section 9.4.4, how design solutions can interact each other through interaction elements. The knowledge link elements are explored through the results of the interaction evaluation process.

This experimental system also explores how the Product Range Model can store temporary design solutions, providing support in terms of which interactions are still pending for a set of temporary design solutions chosen; and how the Product Range Model can evaluate these design solutions against additional modifications in the product specifications (section 9.4.5).

The relationship between the two information models, i.e. Product Model and Product Range Model, is also explored by storing final design solutions in the Product Model (section 9.4.6).

Finally, the behaviour of composite interactions is explored through the interaction states propagation in section 9.4.7.
9.4.2. An Injection Mould DFF application - Basic Dialogues

To start the injection mould design process an injection mould must be selected as well as the Product Range Model that will support its design.

Figure 9-28 depicts a set of screens where a particular injection mould is selected and the basic specifications related to this injection mould and its related plastic component are displayed. Changes in the product model information are mainly made through these dialogues.

Figure 9-29 shows the general windows environment of the experimental system. A modeless dialogue Design Specifications/Decisions, which stays permanently open during the design section of a particular injection mould, is displayed in the right side of the screen. This dialogue shows a) the information about basic specifications of the injection mould; b) the temporary design solutions selected, which are stored in the Product Range Model; c) the design solution already decided and stored in the product model, and d) the pending interactions, in the case of selected design solution that could not have all its interactions evaluated. An arrow, in Figure 9-29, shows a popup menu option where the functional design can be started.

The dialogue responsible for offering the functional enquiry to the user is depicted in Figure 9-30. The five main functions explored in this work are presented in the left side of the dialogue and represent the parent functions of the 5 main systems of the injection mould. Depending on the function chosen different levels of decomposition may need to be selected. In this example the functions and sub-functions related with Feeding Function are highlighted.

Based on the choice of a specific function, a view of all design solutions that can possibly fulfil such a function is provided by the option "VIEW Design Solutions Associated" (Figure 9-31). This option is valid for any function selected and is a result of the general relationship between functions and design solutions defined in the Product Range Model.

This shows how previous information and knowledge related to injection mould functions and design solutions (chapter 6), and stored in the Product Range Model can be retrieved. However, in this case, with no consideration about a specific design situation. The next section explores the reuse of this information based on the previous knowledge related to its application within a design situation.
Figure 9-28 - Initial selection and specification of the injection mould product
Figure 9-29 - General design environment windows

Figure 9-30 - Function decomposition option
9.4.3. Design solutions interactions with initial product specifications

This experiment explores how previous knowledge related to design criteria for application of each design solutions, within a design situation, can be reused to support design decisions. In this experiment this knowledge has been focused on the injection mould initial specifications such as mould configuration, type of runner, number of impressions, properties of the plastic component. These, as identified in chapter 5, have a significant influence on which set of design solutions can achieve a particular injection mould function and hence be offered to the designer.

Figure 9-32 shows two different design solutions results (a and b) for the same functional enquiry depicted in Figure 9-30, i.e. Distribute Runner Layout. This functional enquiry is performed through the Search Design Solutions action button shown in Figure 9-30. The differences between both results are consequence of distinct injection mould specifications provided.
Chapter 6 has addressed that each design solution can assume three different states after having its design criteria evaluated, namely accepted, rejected and non-evaluated. The dialogues depicted in Figure 9-32 show two main list boxes, namely Accepted Design Solutions and Rejected Design Solutions. The non-evaluated design solution status is also shown in the Accepted Design Solutions list box however, the double asterisk sign differentiates this status.

The choice for the type of runner layout system is mainly dependent on the number of impressions, mould configuration, runner type and the impression distribution layout. In this example, purposely, two situations were performed.

In the first one (Figure 9-32-a) only the number of impressions has been specified resulting in antagonist/contradictory choices of runner layouts. For example, in the case of Hot Runner Manifold - HRSQ4 and Rectangular H-Layout layout types. The first one can only be applied in the case of hot runner mould type and the second only in cold runner situations. The same situation occurs between circular, in line or rectangular layout types, which is mainly dependent on the type of impression distribution in the mould. As a result of this lack of specifications, all Accepted Design Solutions have not been completely evaluated, and no helpful information is provided to the designer. This is depicted in Figure 9-33 and Figure 9-34, where some interactions related to two particular design solutions, i.e. "Rectangular H-Layout" and "Hot Runner Manifold - HRST4", are shown after the evaluation process as "Not Evaluated". These interactions, part of the conditions required to the application of these design solutions (Appendix D - Figure D 12) could not be evaluated since no additional specification of the injection mould has been provided, but number of impressions.

On the other hand, the number of impressions is the determinant reason for rejecting the other design solutions that fulfilled initially the function specified (Figure 9-35(a)). In this figure is also depicted the same design solution instance through the Object Inspector tool, highlighting the approved and reproved interactions temporarily associated with this design solutions stored in the Product Range Model (Figure 9-35(b)).
(a) - Open injection mould specifications

(b) Some injection mould specifications defined

Figure 9-32 - Design solutions result for *Distribute Runner Layout* function
Figure 9-33 - Interactions result for "Rectangular H-Layout"

Figure 9-34 - Interactions result for "Rectangular H-Layout"
In the second situation (Figure 9-32-b) in addition to the number of impressions, the injection mould specifications 2-Plates mould configuration and Cold runner type have been added. As consequence a more restricted set of Accepted Design Solutions
is displayed as result. For example, all Hot Runner Manifold solutions types have become rejected as result of the cold runner type, once one of the conditions associated with the reuse of these kinds of design is the runner system be hot type (Appendix D - Figure D 12).

Besides, of the two additional specifications, three different types of cold runner layout i.e., circular, in-line and rectangular (two configurations) have been considered accepted, but not completely evaluated. The reason is because no previous decision has been made about the type of impression distributions, as shown in Figure 9-36(a).

Thus, the Not-Evaluated interaction is not related to the product specifications any more, but to other design decisions that should, or might, be made. Therefore, if a decision about the type of impression distribution in the mould plates has been made previously, the set of accepted design solutions shown in Figure 9-36(a) would be reduced to a more restricted set of accepted design solutions. Figure 9-36(b) shows how the interactions associated with this design solution instance are kept in the

<table>
<thead>
<tr>
<th>DS Name</th>
<th>Interaction Name</th>
<th>Interaction Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Distribution = InLine</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Mould Configuration = 2-P</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Runner type = cold</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Impressions &gt;=2 &amp; &lt;= 8</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Product Range Model database during this process of checking the design solutions validity. The interaction states are highlighted.

As previously stated, the Not Evaluated interactions do not reject a particular design solution for its reuse and hence this design solution can be selected by the designer as a design choice. Figure 9-37 shows that one of the previous accepted design solution displayed in Figure 9-32, i.e. InLine Unbalanced T-Layout, has been selected temporarily. After the confirmation of the user, such a design solution appears as a temporary decision in the modeless dialogue Design Specifications/Decisions (Figure 9-38). However, because one of its interactions has been considered not evaluated, i.e. Distribution = InLine, (Figure 9-36), this interaction appears as a pending interaction, which must be resolved if such a design solution is going to be definitely used.

Figure 9-37 - Selecting a temporary design solution
9.4.4. Interactions with Design Solutions Decisions

The initial product specifications can provide a more restrict set of design solutions that fulfil the injection mould functions. However, as depicted in Figure 9-38, the choice for the first design solutions can result in pending interactions related with other design solutions. This experiment expands the one presented in section before, exploring how interactions with other design solutions can also provide concurrent support to design choices. This is also demonstrated through the reuse of previous information and knowledge stored in the interaction elements.

Figure 9-39 depicts that in order to resolve the pending interactions associated with the choice of the runner layout made in Figure 9-37, the distribution of the impressions in the mould plate must also be decided. In this case, when such function is enquired, the only possible design solution option offered as accepted is the InLine Impression Distribution. After the selection and confirmation of this second design solution, the pending interaction has been resolved and now the choices made do not
present any pending situation (Figure 9-40). However, as new design solutions are chosen to resolve pending interactions, other new pending interactions can eventually appear.

Figure 9-39 - Selecting a design solution related with the pending interaction

Figure 9-40 - No pending interactions based on the design solution choices
In the experimental system developed, the modeless dialogue *Design Specifications/Decisions*, has been created to provide a constant "status" of the general design decisions process of both the design solutions selected and the pending interactions. For this reason, because of the limitations of this dialogue, it does not make any distinction between which design solution is actually holding the pending interactions. Thus, to provide a clear view of the design decisions status in relation to the temporarily design solutions selected, the menu option *View Design Solutions* has been added. This option calls a new dialogue that besides offering a view of which design solutions have been temporarily selected during the design section, shows which of them are particularly associated with the unresolved interactions (Figure 9-41).

![Image of dialogue](image.png)

Figure 9-41 - View design status - temporarily design solutions

Figure 9-42 and Figure 9-43 highlight the unresolved interactions associated with particular design solutions chosen during the design section namely *Tunnel/Submarine Gate* and *Circular Runner Section*. In this example, the interaction related to the kind of ejection technique to be used, i.e. *Ejection different Stripper*, is a
common restriction for both design solutions temporarily selected and shall be avoided. It is important to state that at this level this interaction is being analysed based on the "conflicts" between design decisions chosen so far. Eventually, the stripper ejection solution could be rejected due to the geometric aspects of the plastic component. However, if not, and if this is apparently the best way of ejection the plastic component, than other options of design solutions should be found for the feeding impression and control runner flow functions, through the option of re-starting the design process.

![Diagram of design options and interactions](image)

Figure 9-42 - Unresolved interactions associated with *Tunnel/Submarine Gate*
9.4.5. Changing the Product Specification during the Design Process

During the process of choosing design solutions to fulfil the injection mould functions new specifications of the injection mould product can be added in order to provide a better satisfaction of the pending interactions. Eventually a key specification can be changed, and hence a new evaluation of the set of solutions selected is required.

In the experimental system developed, while the Product Range Model is active in a design section, any change in the product model, through the specification dialogues, will result in a re-evaluation of the design solutions chosen.

Figure 9-44 highlights the results of changing the mould configuration from 2-Plates to 3-Plates under the conditions depicted in Figure 9-41. The design solutions Tunnel/Submarine Gate and Circular Runner Section have changed their state to rejected once both design solutions are only applied to 2-Plates mould configuration.
9.4.6. Storing Final Design Solutions in the Product Model

At any stage of the design process, the temporary design solutions can be stored in the product model, where it will be developed in more detailed design aspects. The user can select which design solutions will be stored in the product model (Figure 9-45). As a general rule, a design solution can only be stored directly in the product model if it does not have any unresolved pending interaction. Otherwise, a message is sent to the user asking for his/her confirmation, as shown in Figure 9-46.

Figure 9-47 shows the changes in the modeless dialogue, where the design solutions selected to be stored in the product model have moved from the temporary Function / Design Solutions to the PM Function / Design Solutions list boxes.

Figure 9-48 shows the result of this design section so far related with the injection mould feeding system. Such a result is shown through the injection mould specifications, particularly in the button Mould Systems.
### Chapter 9

#### Figure 9-45 - Selecting temporary design solutions to be stored in the product model

<table>
<thead>
<tr>
<th>Design Solutions Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injection Mould Systems Temporarily Selected</td>
</tr>
<tr>
<td>Function - Feed Impression - Design Solution - Tunnel/Submarine Gate</td>
</tr>
<tr>
<td>Function - Control Runner Flow - Design Solution - Circular Section</td>
</tr>
<tr>
<td>Function - Distribute Runner Layout - Design Solution - Rectangular H/Layout</td>
</tr>
<tr>
<td>Function - Conform Solution - Design Solution - Multi-Interaction Distribution</td>
</tr>
</tbody>
</table>

#### Figure 9-46 - Confirmation for storing not completely evaluated design solutions in the product model
Figure 9-47 - Change of temporary design solutions from the Product Range Model to the product model

Figure 9-48 - Product Model view of the feeding system
9.4.7. Composite Interactions Behaviour

Composite interactions are used to provide a better representation of pieces of knowledge, which can not be captured by simple interactions individually. This experiment aims to explored the process of evaluation of composite interactions through the propagation of the simple interactions states. Appendix D, Figure D 13, presents a list of composite interactions used in this research, as well as the interactions associated with them.

Figure 9-49 highlights a composite interaction associated to the Matrix Impression Distribution. This interaction is related to the number of impressions and for its approval the number of impressions in the mould must be equal 4, 6 or 8. In the example depicted in Figure 9-49 this interaction has been approved and the reason for that in shown in Figure 9-50, i.e. Impression = 4. Due to the nature of this type of interaction, i.e. OR, the approval of only one of its elements is required for the approval of the whole interaction, in this case, Impression = 4.

The experimental system has been designed to allow composite interactions to be open and hence, its interaction elements can be analysed individually.
Figure 9-50 - Composite interaction - Type OR with three elements

A different kind of composite interaction is shown in Figure 9-51. In this case, the interaction, again related to the number of impressions of the mould, is associated to the *InLine Impression Distribution*, and it is an AND composite type. The number of impression must be between 2 and 8 for its approval. As in the previous example, this interaction has been approved and the reason is show in Figure 9-52, where all interaction elements have been approved; which is the condition required for AND type of interaction.
Figure 9-51 - AND Composite interaction associated with InLine Impression Distribution

Figure 9-52 - Composite interaction - Type AND with two elements
In these two examples discussed above the composite interactions were composed by simple interactions, which are lately checked against the product information. However, there are situations where a set of simple interaction is not enough to represent the piece of design criteria required, and the composite interaction can be composed of other composite interactions. This is exemplified in Figure 9-53, where besides the number of impressions, the type of runner layout is also a decisive aspect in the choice of Circular Impression Distribution. Thus, either situation can define the acceptance of such a design solution. In this case the number of impressions is associated with odd numbers, i.e. 3, 5 and 7, and the circular distribution of the impression reflects in a more effective way of balancing the mould forces. On the order hand, such situation can not be the only decisive factor, once the circular distribution can be eventually preferred even in the cases of different number of impressions. Hence, another decisive factor is related to the runner layout type to be chosen. Therefore, one of these two interaction elements must be approved for the acceptance of this design solution.

Figure 9-53 - Composite interaction composed by other composite interactions
Figure 9-54 shows that for the situation explored so far, i.e. number of impressions equals 4, the composite interaction related with number of impressions has been reproved and the reason is depicted in Figure 9-55. However, because the interaction related with type of runner layout could not be evaluated (Figure 9-56), for the reason no runner layout has been defined so far, the whole interaction has been considered not evaluated.

Figure 9-54 - Partially approved composite interaction - OR Type
Chapter 9

Accepted Design Solutions

Impression Distribution

Design Solutions not completely Evaluated

Figure 9-55 - Reproved elements of number of impressions composite interaction

Figure 9-56 - Not Evaluated elements of runner layout composite interaction
9.5. Evaluation of the Results

This chapter has explained the experiments conducted in this research to explore the information structure developed for the Product Range Model and how such an information model can provide valid information to support design decisions.

The experiments realised in section 9.3 demonstrated how the structure defined to the Product Range Model, in terms of functions, design solutions, interactions and knowledge-links, can capture and store injection mould product range information and knowledge. The result of this experiment is represented in the Appendix D.

The experiments realised in section 9.4 demonstrated how the Product Range Model, together with a Product Model, can support the design decisions, through the reuse of information related to injection mould functions, design solutions, interactions and knowledge links.

An experimental injection mould DFF software application (section 9.4.2) has explored the use of the Product Range Model to support the injection mould design. This has demonstrated how information and knowledge captured in the Product Range Model (presented in Appendix D) can be reused to support design choices through the offer of valid design solutions to the designer. The results provided by the experimental system, in terms of accepted and rejected design solutions, were expected based on the interactions stored in the Product Range Model and associated with each design solution checked.

The importance of the interactions elements has been highlighted in section 9.4.3, 9.4.4 and 9.4.7, not only guiding the designer through his/her decisions during the process of choosing design solutions, but also in how these interactions can be applied and reused to represent the knowledge associated to the application of different design solutions.

Finally, the required compatibility between information structures of the Product Range Model and the Product Model has been explored in section 9.4.6, where temporary design solutions in the Product Range Model have been stored in the Product Model.
10. Discussion, Conclusions and Further Work

10.1. Introduction

The research reported has explored the nature of Product Range Model, within an integrated CAE environment, and presented how such an information model can support reuse of information in product range design. The Injection Mould Product Range Model structure composed by Functions, Design Solutions, Interactions elements and Knowledge-Link elements has been investigated and an experimental system developed to verify the results.

This chapter presents a discussion of the approach taken in this research, followed by conclusions reached and recommendations for further work. Section 10.2. presents a discussion of the major issues explored in this thesis. Conclusions and further work are presented in sections 10.2 and 10.3, respectively.

10.2. Discussion

This research has shown how an information structure can be defined to support design reuse. The Product Range Model, which is separated from the product model, supports the design process providing the designer with valid ways of fulfilling a specific product function. The representation of Interaction elements and Knowledge Link elements have been defined to capture, respectively, knowledge associated with design criteria related to each design solution, and the relationship between the Product Range Model and the product model. This work provides a novel contribution to the design reuse research within its context of integrated information systems.

The approach taken differs from the traditional information model approaches to support product range definitions, where the reuse is mainly focused on the representation of the most appropriate product architecture defined in the product model (Baxter, Juster et al. 1994; McKay, Erens et al. 1996; Gorti, Gupta et al. 1998). These approaches can provide support for composing variants of existing designs based on sets of product specifications. The author of this thesis agrees with the need for a clear definition and understanding of such a product architecture definition when
dealing with product ranges. However, these models alone are insufficient to capture information and knowledge that support design reuse decisions.

The approach taken in this research excludes the inclusion of knowledge in the product model. In the author's view the product model is a central repository for information related to the product life cycle, and hence the inclusion of knowledge in such an information model can significantly increase its complexity (Lei, Taura et al. 1996). Therefore, additional product range information and knowledge, which supports design reuse, must be modelled in a separate information & knowledge model, i.e., Product Range Model.

The relationship between the Product Range Model and product model has been an important part of this research. The Product Range Model should not be defined completely independently of the product model because of the close relationships between aspects of their information content. Two kinds of relationships between the Product Range Model and product model have been identified in this work. The first relationship is related to the need for product information to support the design solutions evaluation within the Product Range Model, and has been achieved through the definition of Knowledge Link elements. The second relationship is related to the need for information structure compatibility between design solutions within the Product Range Model and the solutions techniques in the product model. This has been achieved by the definition of compatible attributes in both classes. Although this research has explored the application of the Product Range Model to support design decisions through the reuse of information in product ranges, these two relationships will be significant for any other information & knowledge models used to support different product life-cycle functions, e.g., manufacturing. This is because these models will require information from the product model to support decisions and will provide information that must be stored in the product model.

This work has explored the concept of a Product Range Model at the conceptual level of design decisions. However, to provide the designer with more comprehensive information decisions, the Product Range Model could be extended to include other information related to design solutions, such as geometric, cost and manufacturing information. This will require the capture of additional kinds of Interaction elements related to this design solution information, as well as an extension of the design solutions information structure compatibility between the two information models.
This research has defined the compatibility between information structures of design solutions in the Product Range Model and solutions techniques in the product model through a restricted set of attributes, which are related to the identification of such design solutions. However, further work will be required to explore this compatibility related to other design solution information, e.g. geometric information and standard components.

How to resolve the interactions between different injection mould design solutions at the beginning of the design process has been recognised as an issue to be resolved (Lee, Chen et al. 1997). The research explored in this thesis has made a contribution to tackling this issue. The knowledge behind each design solution has been captured through a set of independent objects, named Interaction elements. This provides an advantage in that these Interaction elements can be simplistically defined and then reused (Booch 1994; Wainwright, Leung et al. 1996) in different design criteria scenarios. This structure also meets the requirement of KBS definition provided in (Dixon 1995), where pieces of knowledge can be changed without hard code programming. However, the approach taken is limited to the initial interactions that take place in the design of an injection mould and it is recognised that knowledge about other specific aspects of the injection mould design solutions are required for further design phases (Lee, Li et al. 1997). This may require an extension of the Product Range Model information structure to represent other kinds of Interaction elements, e.g. mathematical comparisons.

The relationship between functions and design solutions provide a general association between injection mould functions and possible ways of achieving it. This relationship enables that one function be achieved by one or more design solutions and that one design solution can be used to fulfil one or more functions. This allows the reusability of the same design solution concept for different design functions. One example of this are ejection pins, which can be used to eject the plastic component in different ways, eject gate and eject runner system. However, in this case, there will be relationships between the design criteria associated with the design solution and the function, which is achieved through such a design solution. For example, the design criteria for applying ejection pins for ejecting the plastic component, the gate or the runner may be significantly different. Therefore, the information structure of Interactions elements can be further expanded to also capture this kind of relationship.
Chapter 10

The Interaction elements, defined in this research, provide information that can be used to explain the reasons for particular design solution states, i.e. accepted, rejected and not-evaluated. In the case of not-evaluated design solutions, the designer can select conflicting solutions due to the variety of conditions available in the pending interactions. Using a simple example, if no type of runner has been defined, the choice of a type of gate can be associated with the requirement of a hot runner system, while the impression distribution can be associated with the requirement of a cold runner configuration. This research has not highlighted the potential conflicts between pending interactions, requiring therefore, further investigation in how these pending interactions can be managed to avoid conflicting design solution choices.

The experimental system developed has been shown to be adequate for exploring the research ideas discussed in this thesis. It has demonstrated how the Product Range Model can support a functional enquiry, by offering valid design solutions options based on the use of Interaction elements and Knowledge Link elements. The application of the UML notation alongside Use Cases has provided effective support for the design and representation of the experimental system developed.

Particularly for the design and implementation of the Knowledge Link elements it has been found that in addition to attributes that define the data model path, there is a significant part of knowledge required to use these attributes to retrieve the right information from the product model. This is because of the close dependence of Knowledge Link elements on the internal relationships between classes within the product model information structure. While this has not been a problem in the implementation of the experimental system, there is a need to investigate more appropriate ways of implementing the knowledge required, if major commercial systems are developed.

The information and knowledge within the Injection Mould Product Range Model can be significantly dependent on the plastic component characteristics. There are functions that are common for all moulds, e.g. feed mould. On the other hand, there are also functions that can be specific for ranges of plastic components, e.g. eject by the walls in walled types of products. This research has investigated the Injection Mould Product Range Model based on a specific kind of plastic component, which has resulted in a limited set of functions and design solutions, as well as, reduced amount of knowledge modelled through the Interactions elements and Knowledge
Link elements. This is advantageous since it enables information and knowledge related to a kind of product range to be divided into more focused sets of Interaction elements, allowing a better and simpler definition of this information and knowledge (Harrington and Soltan 1995). However, as different Injection Mould Product Range Models are defined for different ranges of plastic component, e.g. rotational and non-rotational components, the reusability of common information, such as Interactions elements and Knowledge Link elements, should be investigated.

This research used injection moulds as a type of product range to explore the nature of Product Range Models. The information structure achieved is relatively generic since most of product ranges can be structured in main functional main systems, which can be design by using different types of design solutions. The author of this thesis has discussed the product range of generators designed and manufactured by Alsthom Electric Machines Ltd. These would appear to fit within a similar Product Range Model structure, in terms of functions, design solutions, interactions and knowledge links. However, there is the need to investigate how applicable the concept of this information model is to other kinds of product ranges. Also, there is a need to investigate how a Product Range Model could be extended to support other kinds of design, such as original or routine designs.

10.3. Conclusions

I.) It has been shown, in chapter 9; section 9.4.3, that a Product Range Model, operating in an information support environment, can support design reuse. The structure of the Product Range Model has been defined in terms of Functions, Design Solutions, Interaction elements and Knowledge Link elements, and has been discussed in chapters 6, 7 and 8;

II.) It has been shown, using injection mould as a kind of product range (chapter 5), that an effective Product Range Model structure must represent not only information related to functions and design solutions, but also the knowledge associated with the application of these solutions. This knowledge, captured through the Interactions elements, ensures intelligent reuse of design solutions, as demonstrated in section 9.4;
III.) The definition of the Interaction element structure provides an effective means of representing design criteria associated with the reuse of each design solution. This structure, discussed in chapter 7, allows complex conditions to be captured quickly and efficiently, and enables simultaneous reuse of Interaction elements in multiple design criteria alternatives, hence avoiding duplication of information;

IV.) It has been shown, in chapter 8, that while the structure of a Product Range Model can be separate from the product model structure, there are two significant interdependencies between their structures. Firstly, the Product Range Model requires knowledge about the product data model to retrieve product information and hence enable design reuse evaluation. Secondly, information structure compatibility is essential between design solutions options within the Product Range Model, and design solutions selected within the product model;

V.) The Knowledge Link elements provide an effective linking mechanism between the Product Range Model and the product model (section 8.4). This enables the retrieval of product information to support design reuse evaluation, and provides a flexible means of dealing with this kind of interdependency between both information model structures. This has been demonstrated through the experiments in chapter 9, where two separated information structure were actually used;

VI.) The use of the UML notation and Use Cases has proved to be an effective support in the design of the experimental system. Use Cases allow the capture of the Product Range Model and design for function application functionality. UML provides a consistent notation for the representation of the different phases of the design and development of the experimental system;

VII.) An experimental system has been implemented using the object-oriented database ObjectStore© and the Visual C++ programming environment. This system has been explored using real cases from the industrial collaborator to successfully demonstrate the feasibility of the Product Range Model concept to support design reuse.
10.4. Recommendation for Further Work

I.) This research has defined an information structure to represent a Product Range Model and has identified how such an information&knowledge model can support design reuse. The implementation of an Injection Mould Product Range Model has also been demonstrated. However, there is a need to investigate the applicability of the Product Range Model in other kind of product ranges which may be either injection moulding and non injection moulding related;

II.) Design solutions have been explored as concepts, which has required information structure compatibility with the product model at a general level. There is a need for further work to explore design solutions as physical entities, geometrically represented. This will require the definition of a more detailed level of information structure compatibility between the two information models, as well as the addition of geometric interactions;

III.) It has been shown that the Product Range Model can support the association of design solutions with manufacturing options, i.e. standard components. However, no investigation has been done into how this association can effectively provide support for design decisions. There is a need for further work in exploring how information about manufacturing options can provide additional support for offering valid design options;

IV.) In this research, the modelling of manufacturing options has been realised within the Product Range Model. However, considering an architecture for a more integrated CAE environment, there is the need to investigate where these manufacturing options components should be stored: Product Range Model, Manufacturing Model or other separated information model?

V.) The generalised representation of functions in the Product Range Model has been sufficient to capture the major Product Range Model relationships. However, where specific functions, which relate to particular product attributes, are required, there is a need to define these relationships. For instance, when enquiring for a "eject feature-Boss" function, there is a need to specify which "Boss" in the product is aimed;
VI.) There is a need to investigate relationships between pending interactions defined in the research, to avoid conflicting choices of temporary design solutions;

VII.) This research has explored the design solutions based mainly on two possible situations, i.e. accepted or rejected. However, in order to provide more intelligent reuse to the designer, fuzzy aspects could be considered within Interactions elements, hence providing a level of acceptance in which the design solution can be applied for a specific design condition;

VIII.) As new information can be associated with design solutions within the Product Range Model, there is a need to investigate how the interaction information structure can be expanded to represent additional kinds of interactions, such as mathematical comparisons. There is also the need to explore other kinds of design solution interactions, such as related to costs and manufacturing;

IX.) This work has explored the Knowledge-Link mechanisms in an explicit definition of the product information model. However, further work is required to explore how the Knowledge Link elements are developed in more general information model structures.
References


Data and Knowledge Systems for Manufacturing and Engineering, Hong Kong. 1: 174-179.


References


References


References


References


References


Appendix A - Tools and Method Used in the Research

A-1. Introduction

This appendix provides a description of the notation, tools and method used to support the representation of the information and the development of the experimental system realised in this research. Section A-2 describes the UML (Unified Modelling Language) notation and diagrams, which has been used to represent information structures developed and the results of the experimental system analysis and design. Section A-3 describes the Use Case driven process applied, and section A-4 provides a general description of the programming environment and the database utilised in this work.

Complementary information about UML notation and process used in this research can be found in (Texel and Williams 1997; Booch, Rumbaugh et al. 1999; Jacobson, Booch et al. 1999).

A-2. UML (Unified Modelling Language) Notation and Diagrams

A-2.1. Use Cases Representation

Use Cases represent the high-level functionality of the system in development, describing “what the system should do”. Although the Use Cases try to capture what users want from the system, they do not specify how the system should do it, what is left to the system design phases.

Each Use Case itself does not contain much information, and hence its description must be complemented by Use Case scenarios, which are composed of the flow of events such as details about user actions, software actions and reactions, constraints, needs for graphical interface, relationships with other Use Cases, etc. The Use Case description defines ideally what the Use Case should do to achieve their functionality, and provide helpful information when specifying the properties (attributes and methods) of the classes needed to perform such Use Case.

Three elements can be identified in each Use Case textual format:

- Actor: represents an external stimulus to the software system. (Texel and Williams 1997) expand this definition to internal stimulus as well,

- Action: represents a capability requested of the software, and
Subject: represents the item acted upon by an Action requested of the software.

As result of Use Cases identification, a Use Case diagram can be modelled representing the relationships between Actors and their Actions, represented by Use Cases. Generalisation and association represent the types of relationships used in this diagram.

Figure A 1 shows a simple Use Case diagram where an actor, represented by the designer, and four Use Cases, which represent main functions of a particular product design system, can be identified. The designer is responsible for starting the Use Cases “Start_Design_Process” and “Design_by_Function”. Two ways of starting the design process, named “Create_New_Product” and “Select_Existing_Product” can be identified, and maintain a relationship of the generalisation type with the "Start_Design_Process" Use Case. The Use Case “Check_Design_Solutions” is related, through an association relationship, with the Use Case “Design_by_Function”.

![UML Use Case Diagram](image)

Figure A 1 – UML Use Case Diagram

A-2.2. Classes Representation

A-2.2.1. Packages or Categories Diagrams

One of the main transitions using UML notation is from Use Cases to the representation of the system in terms of objects. Packages, or Categories, represent main parts of the system, which Use Cases interact with. These elements define a group of classes with common functionality, providing support to a modular design and implementation of the system.
Class diagrams are used to model the initial phases of system analysis and design, representing the high level associations between system packages, or categories. In this representation only dependency relationship is used, which indicates that one category depends upon, or requires the resources from, other category.

Figure A.2 depicts the class diagram applied to packages representation. This diagram allows an initial view of the system in terms of main parts that need to be developed. In this example “Functions” and “Design Solutions” are main categories, which can contain different levels of class diagrams. Also, “Functions” category references “Design Solutions” category showing that there will be relationships between the classes of these two categories. “Input” category represents all objects related to the user's input to the system, while “Display” category is responsible for displaying the results of the particular user enquires.

![Diagram of class category relationships](image)

**Figure A.2 – Class Category Diagram**

A-2.2.2. Class Diagrams

Classes describe a set of objects that share the same attributes, methods, relationships and semantics. The Class diagrams represent the internal structure of objects, which can be described by its name, attributes and methods (Figure A.3), and relationships with other objects, which are described through the inheritance, association or aggregation types of relationships.
Figure A 3 - Class representation

Figure A 4 depicts a Class diagram for two kinds of products, named Injection Mould and Plastic Component, where three main kinds of associations are highlighted:

I. Inheritance: both products are specialisation of Product class and hence inherit main attributes and behaviour from such class, such as product name and number. Product class, on the other hand, is a generalisation of Injection Mould and Plastic Component classes;

II. Association: the relationship between Injection Mould and Plastic Component classes is called association and represents a general relationship between them. Thus, either classes can reach each other, however keeping their individuality;

III. Aggregation: the relationship between Injection Mould and Mould Plate classes is called aggregation, where one class is “made up of” another class. Thus Injection mould (“whole”) has Mould Plate (“part”), and can be, or not, responsible for the creation of its “part”.

Figure A 4 – UML Class Diagram

Association and aggregation relationships have significant consequences in terms of hard coding of the program in a particular language (e.g. C++). For example, whilst a 1...1 relationship requires only a reference to an object, a 1...n relationship requires implementation of storage classes which is considerably more complex (management
of pointer or reference list, etc.). Therefore, there is a corresponding demand to use proper criteria for selection and applications of UML class diagram.

A-2.3. Sequence of Actions Representation

Sequence diagrams model the dynamic aspects of the system. They capture and represent the interactions required between objects, through their methods, emphasising the time ordering of messages.

These diagrams represent mainly the behavioural aspects of objects, showing what methods (functionality) are required to satisfy a specific Use Case. These diagrams can be also applied to model the initial phases of the system analysis, using categories, as objects, and Use Cases, as methods, providing a dynamic high level representation of the system for a good discussion and initial understanding of the system.

Figure A 5 shows an example of a general representation UML Sequence Diagram, where categories are arranged along X-axis and methods (Use Cases), ordered in a numerical sequence, are arranged along Y-axis. Thus, in order to check which design solutions can be eligible for attending a specific function, a Use Case named "check_valid_design_solution" is performed by the "Function" category. Similarly, the "Design_Solutions" category performs another Use Case, named "check_interactions", which is responsible for evaluating all possible interactions that such design solutions can have.

![UML Sequence Diagram](image)

Figure A 5 – UML Sequence Diagram
A-2.4. Activity Representation

The Activity diagrams model the sequential steps (activities) in a computational process, and are also related with the representation of the dynamic aspects of the system.

Each activity results in some action that, in turn results in changes in state of the system or return of a value. Actions can encompass calling another operation, sending a signal, creating an object, or some pure computational expression. Thus, activity diagrams are typically applied to model either a system workflow (between object) or an operation (computational steps to be performed by a specific method).

Figure A 6 depicts a general example of activity diagram. Different activities performed by the system are shown in a sequential route of action. Simple arrows represent transitions from one activity state to another. The diamond boxes are defined as sequential branches and different actions are taken depending on the results expected. For instance, if a design solution, after checked, is considered accepted, it must be stored in a collection of accepted design solutions. Otherwise, it is considered rejected and hence stored in a collection of rejected design solutions. Forks and joins (thick horizontal bars) represent the splitting and synchronisation of concurrent flows, respectively. For instance, in Figure A 6, independent of the test design solution result, a next design solution must be selected for the checking process.

A-2.5. State Transition Representation

State Transition diagrams represent the internal behaviour of a class during its lifetime, showing the sequence life cycle phases of its instances, and the events that cause such instances to change from one state to another. In order to comply with different states and transitions, a class must have some attribute that sets its state.

States are defined as situations during the life of an object in which it satisfies some condition, performs some activity, or waits for some event. For instance, in Figure A 7, the state of a specific design solution can be neutral, accepted, rejected or selected.

Transition is a relationship between two object states indicating that based on certain actions and satisfaction of specific conditions, an object state can be changed to a second state. In the example of Figure A 7, to change from a neutral condition to a rejected state, a design solution must be checked (action) by a particular method, e.g. "check_design_solution()" and have its result as "reject".
A-3. Use Case Driven Process

This research has followed a formal methodology, proposed by (Texel and Williams 1997), to guide the analysis, design and implementation of the experimental system. The process chosen supports the evolution of the UML notation representation.
throughout all phases of a system development. However, for the main proposal of this research only some of the initial phases of this process has been applied, and are described below.

Based on a textual description of what is expected from the system, made by professionals who will be involved with the use and development of the system, functional requirements ("shall statements") are identified. The functional requirements represent what a system should do (actions performed by the system) in the perspective of the final user, and are placed in a Requirement Traceability Matrix (RTM). Each functional requirement is addressed as a potential Use Case (Figure A 8), which is composed textually by actor, actions and subject (section A-2.1).

The Use Cases, in turn, are complemented by its description, which is composed by a scenario (sequence of actions, constraints, relationships with other Use Cases, etc.) and a graphical interfaces if necessary. This description provides an important source of information to support the subsequent phases of analysing and designing the software classes. Other additional pieces of information are assigned also to each Use Case such as type of requirements for its implementation (software, hardware, etc.) and, mainly, the system categories, which will define main modules of the system in development.

\[\text{Requirements Capturing} \rightarrow \text{Use Cases Identification}\]

\[\text{Description and agreement of what the system should do} \rightarrow \text{Capture "shall" statements from Description} \rightarrow \text{Identifier software "Use Cases"} \rightarrow \text{Actor - Action - Subject} \rightarrow \text{Establish Project "Categories"} \rightarrow \text{Category INPUT} \rightarrow \text{Requirement Traceability Matrix (RTM)} \rightarrow \text{Use Case identification process}\]
Once Use Cases and Categories have been identified and agreed upon users and developers, static and dynamic views of the system can be produced incrementally, supported by Use Case scenarios and graphical interface descriptions.

Figure A9 depicts the two following phases of the overall process, named system analysis and software analysis and design.

During the system analysis phase, the static and dynamic views of the system are produced through Class and Sequence diagrams at the category level. Class diagrams provide a validation of the categories and identification of the responsibility and associations of each category (static view). Sequence diagrams allow also a validation of the category responsibility besides their interactions to implement each Use Case (dynamic view).

A refinement of the system categories is realised, during the software analysis phase, focusing what classes are required into each category (static view) and the interactions between these classes to realise each Use Case (dynamic view). This refinement is further extended to identify the internal properties of each class and their dependencies with other classes (static view), and interactions between these classes in terms of specific methods and attributes (dynamic view).

Figure A9 – Categories, objects and relationships identification process
At this phase, the class structure description suffers significant influence of the programming language chosen for implementation. State Transition and Activity diagrams also support this phase.

Once finished the object-oriented analysis and design of the system, most of information is now available for the computational implementation.

A-4. ObjectStore/Visual ++ Implementation Process

A-4.1. ObjectStore Database Designer

As described in Chapter 3, the experimental system has been developed using Visual C++ as programming environment for capturing the main interfaces of the system and the ObjectStore as a database for capture the persistent information of the information models developed.

Thus, the class structures, defined for representing the information data models, have been used as input in the ObjectStore Database Designer, where attributes, methods and dependencies are assigned to each class (Figure A 10). One particularity of the ObjectStore diagram when compared with the UML Class diagrams is the fact how the dependencies, association and aggregations, are built. For example, the UML aggregation “has” relationship is now represented as “collections” or “pointers” in ObjectStore diagram.

A-4.2. Visual C++ Programming Environment

Based on the class diagram designed in the ObjectStore Database Designer, the database schema file is generated for hard code generation, which is performed by ObjectStore in the Visual C++ environment.
Two main visual C++ Projects are generated inside a major Visual C++ Workspace (Figure A11). The first project (PRMOI files) is related with all ObjectStore classes and database schema, which define the persistent data, while the second one (PRMOI_MFC) is related with MFC, which contains the application classes, and all classes related with user interfaces. Each class, in turn, is composed by two files, i.e. header (.h) and source code (.cpp) files.

Figure A12 depicts a header file for a Function class, while Figure A13 depicts the source code file for the same class.

A-4.3. ObjectStore Inspector

Finally, after the database has been populated, ObjectStore Inspector allows object attributes be inspected for each specific class. Figure A14 shows an example of the basic window of the ObjectStore Inspector, where Function objects are displayed by their specific attributes, e.g. Function_Name and Function_Type.

Three panes are displayed in Figure A14: (a) the list of database roots; (b) a tree schema representation and (c) the list of instances under consideration.
Appendix A

Each object in the list of instances can be open and hence its data contents (Figure A 15) and when an instance association with other instances is shown, it is possible navigate through the related instances. The results of the navigation process can also be displayed and saved (Figure A 16). Such resources have been used to show part of the results achieved by the implementation of the experimental system in this work.

Figure A 11 – Visual C++ Programming Environment (Workspace ‘PRM01’)

```cpp
#pragma once

#include "Show_DS_Result_Dlg.h"

class Injection_Moulding_Machine;
class Manufacturing_Model;
class Product_Model;
class Injection_Mould;
class Plastic_Component;
class Function;
class Design_Solutions;
class Ejection_RingStripper_DS;
class Material;
```
class Design_Solutions;  // Related Class
class OS_DB_EXT_CLASS Function  // Class Definition
{
public:
    Function(int bAddToRoot = 0);
    Function(int Function_type, os_Collection<Design_Solutions *> * _Design_Solutions, os_Collection<Design_Solutions *> * _Accepted_Design_Solutions, os_Collection<Design_Solutions *> * _Rejected_Design_Solutions, os_Collection<Design_Solutions *> * _Selected_Design_Solutions, char * Function_name, int bAddToRoot = 0);
    static void UpdateExtents(Function* pFunction, int add, os_database* pdb = NULL);
    virtual ~Function();
    static os_typespec* get_os_typespec();

    // Attributes
    public:
    void set_Functype(int value){ Function_type = value; };
    int Function_type;
    void set_Design_Sol_Collection(os_Collection<Design_Solutions *> * value){
        Design_Sol_Collection = value;
    }
    os_Collection<Design_Solutions *> * Design_Sol_Collection;
    void set_Accepted_Design_Solutions(os_Collection<Design_Solutions *> * value){
        Accepted_Design_Solutions = value;
    }
    os_Collection<Design_Solutions *> * Accepted_Design_Solutions;
    void set_Rejected_Design_Solutions(os_Collection<Design_Solutions *> * value){
        Rejected_Design_Solutions = value;
    }
    os_Collection<Design_Solutions *> * Rejected_Design_Solutions;
    void set_Selected_Design_Solutions(os_Collection<Design_Solutions *> * value){
        Selected_Design_Solutions = value;
    }
    os_Collection<Design_Solutions *> * Selected_Design_Solutions;
    void set_Function_name(char *value);
    char * Function_name;

    // Operations
    public:
    virtual int check_design_solutions(int );
    void set_design_solution(int );

    void SendDatabase (os_database* DB_passed);
    os_database* DB;

    // Additional Attributes (added manually)
    int Function_Status;
    os_Collection<Design_Solutions *> * check_design_solutions(int Function_ID);
};
#endif

Figure A 12 - ObjectStore Header file (Function.h)
Appendix A

//ObjectStore Database Class(es) created by ObjectStore Component Wizard

#include "stdafx.h"
#include "Function.h"
#include "Design_Solutions.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Function

// Default Constructor – No parameters (generated by ObjectStore Wizard)
Function::Function(int bAddToRoot /*=0*/) {
    Function_type = 0;
    Design_Sol_Collection = 0;
    Accepted_Design_Solutions = 0;
    Rejected_Design_Solutions = 0;
    Selected_Design_Solutions = 0;
    Function_name = 0;
    if(bAddToRoot)
        UpdateExtents(this, TRUE);
}

// Constructor – Class parameters (generated by ObjectStore Wizard)
Function::Function(int _Function_type, os_Collection<Design_Solutions *> _Design_Sol_Collection, os_Collection<Design_Solutions *> * _Accepted_Design_Solutions, os_Collection<Design_Solutions *> * _Rejected_Design_Solutions, os_Collection<Design_Solutions *> * _Selected_Design_Solutions, char * Junction_name, int bAddToRoot /*=0*/) {
    Function_type = _Function_type;
    setFunction_name(_Function_name);
    if(bAddToRoot)
        UpdateExtents(this, TRUE);
}

// Function to store object in the Database (generated by ObjectStore Wizard)
void Function::UpdateExtents( Function* pFunction, int add, os_database* pdb /*= NULL*/) {
    if (! pFunction && ! pdb)
        return;
    TIX_HANDLE(err_objectstore)
    os_database *db;
    if (pFunction && &objectstore::is_persistent(pFunction))
        db = os_database::off(pFunction);
    else
        db = pdb;
    char root_name[] = "Function";
    os_database_root *db_root_Function = db->find_root(root_name);
    os_Collection<Function*>* db_root_c_Function;
    if (!db_root_Function)
        db_root_Function = db->create_root(root_name);
    db_root_c_Function = &os_Collection<Function*>::create(db);
    db_root_Function->set_value(db_root_c_Function);
    else
        db_root_c_Function = (os_Collection<Function*>* *) db_root_Function->get_value();
    if (pFunction &&
objectstore::is_persistent(pFunction)
if (add)
    db_root_c_Function->insert(pFunction);
else
    db_root_c_Function->remove(pFunction);
TIX_EXCEPTON
TIX_END_HANDLE
}

// Destructor (generated by ObjectStore Wizard)
Function::~Function()
{
    if(Design_Sol_Collection)
        delete Design_Sol_Collection;
    if(Accepted_Design_Solutions)
        delete Accepted_Design_Solutions;
    if(Rejected_Design_Solutions)
        delete Rejected_Design_Solutions;
    if(Selected_Design_Solutions)
        delete Selected_Design_Solutions;
    if(Function_name)
        delete Function_name;
}
UpdateExtents(this, FALSE);

// Function to input the Function Name attribute in the Database
void Function::set_function_name(char *value)
{
    if(!value)
        return;
    int len = strlen(value);
    if(!len)
        return;
    if(Function_name)
        delete Function_name;
    Function_name = new (os_layout_service::GetDataSegment(this), os_typespec::get_char(), len+1) char[len+1];
    strncpy(Function_name, value, len);
}

// Function to Get the Database Name (added manually)
void Function::SendDatabase (os_database* DB_passed)
{
    DB = DB_passed;
}

// Function definition generated by ObjectStore, but not implemented yet
void Function::check_design_solutions(int Function_type)
{
    int ret;
    //TODO: Add your code here
    return ret;
}

// Function to Find Accepted Design Solutions (added manually)
os_Collection<Design_Solutions*>* Function::check_design_solutions(int Function_ID)
{
    int temp = Function_ID;
os_Collection<Design_Solutions*>* All_DS;
Design_Sol_Collection = &os_Collection<Design_Solutions*>::create(DB);
Accepted_Design_Solutions = &os_Collection<Design_Solutions*>::create(DB);
Rejected_Design_Solutions = &os_Collection<Design_Solutions*>::create(DB);
All_DS = &os_Collection<Design_Solutions*>::create(DB); // This is a temp variable
os_database_root* ptDS_root = DB->find_root("Design_Solutions");
if (! ptDS_root)
{
    ptDS_root = DB->create_root("Design_Solutions");
    ptDS_root->set_value (All_DS);
}
else
{
    All_DS = (os_collection<Design_Solutions*>*) ptDS_root->get_value();
}
const os_coll_query &valid_DS = os_coll_query::create("Design_Solutions*",
    "Function_type == *(int*)func_ID", DB); // this->Function_type
os_bound_query DS_Range (valid_DS, (os_keyword_arg("func_ID", &temp)));
Design_Sol_Collection = &(All_DS->query(DS_Range));
if (Design_Sol_Collection->empty())
{
    return Accepted_Design_Solutions;
}
os_Cursor<Design_Solutions*> DS_Check (*Design_Sol_Collection);
for (Design_Solutions* DS = (Design_Solutions*) DS_Check.first();
    (Design_Solutions*) DS_Check.more();
    DS = (Design_Solutions*) DS_Check.next())
{
    DS->SendDatabase(DB);
    DS->send_InjectionMould_ptr(pInjectionMould);
    if (DS->check_interactions())
    {
        Accepted_Design_Solutions->insert(DS);
    }
    else
    {
        Rejected_Design_Solutions->insert(DS);
    }
}

os_coll_query::destroy(valid_DS);
return Accepted_Design_Solutions;
}
// Function definition generated by ObjectStore, but not implemented yet
void Function::set_design_solution(int Function_type)
{
    //TODO: Add your code here
}
// Function to define types of collection references (generated by ObjectStore Wizard)
void Function_force_vfts (void*){
    os_Array<Function*> pos_Array_Function;
    os_Bag<Function*> pos_Bag_Function;
    os_Collection<Function*> pos_Collection_Function;
    os_List<Function*> pos_List_Function;
    os_Set<Function*> pos_Set_Function;
}

Figure A 13 - ObjectStore Source Code file (Function.cpp)
Figure A 14 – ObjectStore Inspector basic window- Collection of Function objects

Figure A 15 - Instance "Feed Impression" of Function class
Figure A 16 - Navigation path into the database: Feed Impression function and its associated design solutions
APPENDIX B - An Example of the Analysis and Design Process of the Experimental System

B-1. Initial System Description - Analysis Phase

B-1.1. Requirements Traceability Matrix

The Requirement Traceability Matrix (RTM) captures the key sentences of a system description, which identifies functions that the system should perform. These sentences are refined to avoid duplication, and re-written to provide a better representation of each function required for the system. The Use Cases and Packages are extracted from these final sentences.

The RTM is used also for identifying other information related to the Use Cases development, such as the means that they will be implemented; in which phase of the system development, which team will be responsible, etc. However, this has not been defined as a main issue to be explored in this research.

Table B 1 shows some of the Use Cases initially defined during the phase of system analysis, where is highlighted the Entry 5, which is used as an example to show the development of the process applied.
<table>
<thead>
<tr>
<th>Entry</th>
<th>Functional Sentences</th>
<th>Use Cases</th>
<th>Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>User starts a new Injection Mould design</td>
<td>UC1_User_Starts_Design</td>
<td>Input_CAT</td>
</tr>
<tr>
<td>02</td>
<td>The IMSS shall ask for the initial information and requirements of the injection mould and plastic component</td>
<td>UC2_IMSS_Request_Initial_Information</td>
<td>Input_CAT</td>
</tr>
<tr>
<td>03</td>
<td>The IMSS shall store the initial information and requirements of the injection mould in the PM</td>
<td>UC3_IMSS_Store_InitialInformation_PM</td>
<td>PM_CAT</td>
</tr>
<tr>
<td>04</td>
<td>The User shall choose an injection mould function.</td>
<td>UC4_User_Choose_FunctionDesign</td>
<td>Function_CAT</td>
</tr>
<tr>
<td>05</td>
<td><strong>IMSS shall check valid Design Solutions</strong></td>
<td>UC5_IMSS_Check_ValidDesignSolutions</td>
<td>Design_Solutions_CAT</td>
</tr>
<tr>
<td>06</td>
<td>IMSS shall check Design Solutions Interactions against PM initial information</td>
<td>UC6_IMSS_Check_PMInteractions</td>
<td>Interactions_CAT</td>
</tr>
<tr>
<td>07</td>
<td>PRM stores Valid Design Solutions</td>
<td>UC7_IMSS_Stores_AcceptedDesignSolutions</td>
<td>PRM_CAT</td>
</tr>
<tr>
<td>08</td>
<td>PRM stores Non-Valid Design Solutions</td>
<td>UC8_IMSS_Stores_RejectedDesignSolutions</td>
<td>PRM_CAT</td>
</tr>
<tr>
<td>09</td>
<td>IMSS presents a set of Valid Design Solutions</td>
<td>UC9_IMSS_Display_ValidDesignSolutions</td>
<td>Display_CAT</td>
</tr>
<tr>
<td>10</td>
<td>IMSS presents a set of Non Valid Design Solutions</td>
<td>UC10_IMSS_Display_NonValidDesignSolutions</td>
<td>Display_CAT</td>
</tr>
<tr>
<td>11</td>
<td>IMSS shall display reproved interactions for each non valid Design Solution</td>
<td>UC11_IMSS_Display_ReprovedInteractions</td>
<td>Display_CAT</td>
</tr>
<tr>
<td>12</td>
<td>The User shall choose from PRM options of solutions in each functional design stage</td>
<td>UC12_User_Choose_DesignSolution</td>
<td>Design_Solution_CAT</td>
</tr>
<tr>
<td>13</td>
<td>The IMSS shall check interactions with other Design Solutions</td>
<td>UC13_IMSS_Check_DesignSolutionsInteractions</td>
<td>Interaction_CAT</td>
</tr>
<tr>
<td>14</td>
<td>The IMSS shall store the decisions chosen in the PM</td>
<td>UC14_IMSS_Stores_FinalDesignSolution_PM</td>
<td>PM_CAT</td>
</tr>
<tr>
<td>15</td>
<td>The IMSS shall store temporary Design Solutions in the PRM</td>
<td>UC15_IMSS_Stores_Temporary_Design</td>
<td>PRM_CAT</td>
</tr>
<tr>
<td>16</td>
<td>The IMSS shall display possible standard solutions Valid Design Solution</td>
<td>UC16_IMSS_Display_StandardSolutions</td>
<td>Display_CAT</td>
</tr>
<tr>
<td>17</td>
<td>The IMSS shall retrieve product specification and previous decisions information to be compared with each interaction</td>
<td>UC18_IMSS_Retrieve_Stored_Information</td>
<td>Interactions_CAT</td>
</tr>
</tbody>
</table>

Table B 1 - Requirement Traceability Matrix
B-1.2. Use Case Scenarios

Use Case scenarios describe the aspects related to each Use Case, such as a general description; pre and post-conditions; actions and reactions required by the software and user; graphical interfaces needed; relationships with other Use Cases. This information is critical for the further development of the system classes, mainly related to their behaviour aspects. Table B 2 presents an example of a Use Case scenario related to the Use Case 5 (UC5) of the RTM depicted in Table B 2.

<table>
<thead>
<tr>
<th>Overview:</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Use Case evaluates, for each selected Function, the sets of valid Accepted and Rejected Design Solutions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preconditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• A Function Design must be chosen;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
</tr>
<tr>
<td>1. Check all possible Design Solutions to attend the Function specified</td>
</tr>
<tr>
<td>2. For each Design Solution check the Interactions;</td>
</tr>
<tr>
<td>3. If all interaction status are Approved, the Design Solution Status is Accepted;</td>
</tr>
<tr>
<td>4. If one or more interaction status are Reproved, the Design Solution Status is Rejected;</td>
</tr>
<tr>
<td>5. The Reproved Interactions are kept for each Rejected Design Solution</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario Notes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>This scenario doesn't have the interference of the operator. For each specific Design Solution the scenario must be repeated. The &quot;Rejected Design Solutions&quot; keep the interactions that have been not respected to show the User.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post Conditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• A set of Design Solutions is available to be selected by the User;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Required GUI (Graphical User Interface): None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Exceptions: None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Cases Utilised: UC6 and UC13</th>
</tr>
</thead>
</table>

Table B 2 - Scenario for UC5: IMSS_Check_ValidDesignSolutions

- 236 -
B-2. General System Diagrammatic Representation - Analysis and Design Phase

B-2.1. Use Cases Representation

The Use Cases diagram represents the functionality of the system in development. In the example shown below, some of the Use Cases identified in the RTM are represented in the Use Case diagram, as well as their relationships with other Use Cases and actors.

For the experimental system developed in this research two main functionality has been identified in relation to the Product Range Model. The first one is related to how this information model can support the design process and is represented in Figure B 1. The second one, not addressed in major details in this research, is related to the management aspects of the Product Range Model, such as storing new instances, changing the existent ones, and is depicted in Figure B 2.

![Use Case diagram for design support PRM functionality](image-url)

Figure B 1 - Use Case diagram for design support PRM functionality
B-2.2. Packages/Categories Representation

The representation of the system in terms of packages aims to identify the relationships between the main groups of objects, which share common functionality within the system. The packages are also identified from the RTM through an analysis of the textual description of each Use Case.

Figure B 3 describes the experimental system in terms of its main packages, where the Functions, Design Solutions and Interactions packages are highlighted. Although such packages are part of the PRM, a PRM_CAT package is also depicted. However, in this case, the meaning of this PRM_CAT is related to the database itself, i.e. information repository. The PM_CAT represents the product information structure and basically relates to the PRM_CAT and to the Interaction_CAT. The Display_CAT represents the graphical interface objects that support the interface of the user with the system.

The definition of packages is not compulsory in the development of the system. There will be cases where such definition is not required and the definition of a class can achieve the required system function.
B-2.3. **Sequence Diagrams - Use Cases and Packages**

So far, the diagrammatic representations provided have been related to the static aspects of the system. The Sequence diagrams capture the dynamic behaviour of the system through the relationships between system packages and the Use Cases. For each Use Case identified in the RTM, at least one Sequence diagram should be realised identifying its relationships with the system packages. Figure B 4 depicts the representation of the Use Case *IMSS_Check_ValidDesignSolutions*, as well as its relationships with the Function_CAT; Design_Solutions_CAT, and Use Cases 06 and 13 (Table B 1).

**Figure B 4 - Sequence diagram for Packages and Use Cases relationships**
B-3. System Classes Diagrammatic Representation - Design Phase

B-3.1. System Classes Representation

Each package defined for the system needs to be expanded in terms of the class structures that will represent the system in development. Class diagrams provide a static representation of the system in terms of its classes, their relationships and attributes.

For example, for the Design Solutions package different kinds of design solutions are identified in the injection mould design, such as related to cooling system, ejection system, feeding system, etc. Each of these design solutions will have particular characteristics, which justify the definition of these kinds of classes. This is valid for every package identified in the system.

Figure B 5 depicts a class diagram representation for types of injection mould design solutions focused on this research. Each of these design solutions can also be expanded in other sub-classes that provide a more defined classification of their representation.

![Design Solutions Class Diagram](image)

Figure B 5 - Design Solutions class representation

Figure B 6 expands the ejection design solution, highlighting different kinds of ejection pins. A good example of the identification process of the classes within a package, is demonstrated in Chapter 7, where an information structure is defined for representing the interaction elements.
B-3.2. System Class Behaviour Representation

To complement the description above, the relationships between classes objects can be captured through the Sequence diagrams. This diagram allows the representation of dynamic aspects of the system, and at this stage, the relationships are expanded in terms of the classes methods, rather than Use Cases. Figure B 7 highlights the relationships between the classes Functions and Design Solutions to implement the UC5: IMSS_Check_ValidDesignSolutions.

Figure B 7 - Sequence diagrams for class relationships
APPENDIX C - UML Representation of the Experimental System Classes

C-1. Product Range Model Information Structures Representations

Figure C1 - Product Range Model general information structure
Figure C2 - Product Range Model - Interactions information structure
Figure C3 - Product Range Model - Design Solutions information structure

C-2. Product Model Information Structure Representation

Figure C4 - Injection mould and plastic component product association
Figure C5 - Plastic component general geometry description
Figure C6 - Plastic product geometric features
Figure C7 - Injection mould plates and injection mould systems
Figure C8 - Injection mould design solutions in the product model
Figure C9 - Ejection design solutions
Figure C10 - Runner design solutions
Figure C11 - Gate design solutions
Figure C12 - Impression layout design solutions
Figure C13 - Cooling design solutions
Figure C.1: Product Range Model general information structure
Figure C2 - Product Range Model - Interactions information structure
Figure C 3 - Product Range Model - Design Solutions information structure
Figure C 4 - Injection mould and plastic component product association
Figure C 5 - Plastic component general geometry description
Figure C.6 - Plastic product geometric features
Figure C 7 - Injection mould plates and injection mould systems
Figure C 8 Injection mould design solutions in the product model
Figure C 9 - Ejection design solutions
Figure C 10 - Runner design solutions
Figure C 11 - Gate design solutions

Figure C 12 - Impression layout design solutions
Figure C 13 - Cooling design solutions
APPENDIX D - Information Stored in the Product Range Model

Figure D 1 - Injection mould functions
Figure D 2 - Runner layout design solutions
Figure D 3 - Runner cross section design solutions
Figure D 4 - Impression distribution design solutions
Figure D 5 - Gates design solutions
Figure D 6 - Ejection design solutions
Figure D 7 - Functions and design solutions association
Figure D 8 - Numerical interactions
Figure D 9 - Existence interactions
Figure D 10 - Composite "OR" interactions
Figure D 11 - Composite "AND" interactions
Figure D 12 - Interactions associated to design solutions
Figure D 13 - Interactions associated with composite interactions
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Name</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Eject Function</td>
<td>Eject Impression</td>
<td>Eject Impression - by Wall</td>
</tr>
<tr>
<td>2 Eject Impression</td>
<td>Eject Impression - by Base</td>
<td></td>
</tr>
<tr>
<td>3 Eject Feeding</td>
<td>Eject Sprue</td>
<td></td>
</tr>
<tr>
<td>4 Eject Runner</td>
<td>Eject Gate</td>
<td></td>
</tr>
<tr>
<td>5 Eject Feature</td>
<td>Eject Rib</td>
<td></td>
</tr>
<tr>
<td>6 Feed Function</td>
<td>Feed Impression</td>
<td></td>
</tr>
<tr>
<td>7 Distribute Feeding</td>
<td>Distribute Runner Layout</td>
<td></td>
</tr>
<tr>
<td>8 Cool Function</td>
<td>Cool Plate</td>
<td>Cool Cavity Plate</td>
</tr>
<tr>
<td>9 Cool Impression</td>
<td>Cool Core Plate</td>
<td></td>
</tr>
<tr>
<td>10 Cool Runner</td>
<td>Cool Cavity</td>
<td></td>
</tr>
<tr>
<td>11 Cool Ejection</td>
<td>Cool Core</td>
<td></td>
</tr>
<tr>
<td>12 Cool Feature</td>
<td>Cool Boster</td>
<td></td>
</tr>
<tr>
<td>13 Cool Rib</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 Cool Runner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 Cool Ejection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 Cool Feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 Cool Rib</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 Cool Runner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 Cool Ejection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 Cool Feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 Cool Rib</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure D1 - Injection mould functions

<table>
<thead>
<tr>
<th>Layout_Runner_DS: 14 Items</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DS Name</strong></td>
</tr>
<tr>
<td>1 Hot Runner Manifold - HR2</td>
</tr>
<tr>
<td>2 Circular Spoke Layout</td>
</tr>
<tr>
<td>3 Circular Y-Layout</td>
</tr>
<tr>
<td>4 Rectangular S-Layout</td>
</tr>
<tr>
<td>5 InLine Unbalanced - T-Layout</td>
</tr>
<tr>
<td>6 Rectangular H-Layout</td>
</tr>
<tr>
<td>7 Rectangular X-Layout</td>
</tr>
<tr>
<td>8 Rectangular Y-Layout</td>
</tr>
<tr>
<td>9 Rectangular Unbalanced - H-Layout</td>
</tr>
<tr>
<td>10 Hot Runner Manifold - HRST4</td>
</tr>
<tr>
<td>11 Hot Runner Manifold - HRSQ4</td>
</tr>
<tr>
<td>12 Hot Runner Manifold - HR4M</td>
</tr>
<tr>
<td>13 Hot Runner Manifold - HR6</td>
</tr>
<tr>
<td>14 Hot Runner Manifold - HR6M</td>
</tr>
<tr>
<td>15 Hot Runner Manifold - HR7</td>
</tr>
<tr>
<td>16 Hot Runner Manifold - HR7M</td>
</tr>
<tr>
<td>17 Hot Runner Manifold - HR8</td>
</tr>
<tr>
<td>18 Hot Runner Manifold - HR8M</td>
</tr>
<tr>
<td>19 Hot Runner Manifold - HR9</td>
</tr>
<tr>
<td>20 Hot Runner Manifold - HR9M</td>
</tr>
<tr>
<td>21 Hot Runner Manifold - HR10</td>
</tr>
<tr>
<td>22 Hot Runner Manifold - HR10M</td>
</tr>
</tbody>
</table>

Figure D2 - Runner layout design solutions
### Cross_Section_Runner_DS: 4 Items

<table>
<thead>
<tr>
<th>DS_Name</th>
<th>DS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Circular Section</td>
<td>40</td>
</tr>
<tr>
<td>2 Hexagonal Section</td>
<td>44</td>
</tr>
<tr>
<td>3 Modified Trapezoidal Section - Moveable Plate</td>
<td>42</td>
</tr>
<tr>
<td>4 Modified Trapezoidal Section - Fixed Plate</td>
<td>46</td>
</tr>
</tbody>
</table>

Figure D 3 - Runner cross section design solutions

### Impression_Distribution_DS: 4 Items

<table>
<thead>
<tr>
<th>DS_Name</th>
<th>DS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 One Impression Layout</td>
<td>50</td>
</tr>
<tr>
<td>2 Circular Impression Distribution</td>
<td>51</td>
</tr>
<tr>
<td>3 In Line Impression Distribution</td>
<td>52</td>
</tr>
<tr>
<td>4 Matrix Impression Distribution</td>
<td>55</td>
</tr>
</tbody>
</table>

Figure D 4 - Impression distribution design solutions

### Gate_DS: 6 Items

<table>
<thead>
<tr>
<th>DS_Name</th>
<th>DS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Sprue Gate</td>
<td>20</td>
</tr>
<tr>
<td>2 Rectangular Gate</td>
<td>21</td>
</tr>
<tr>
<td>3 Diaphragm Gate</td>
<td>23</td>
</tr>
<tr>
<td>4 Tunnel/Submarine Gate</td>
<td>25</td>
</tr>
<tr>
<td>5 Pin Point Gate</td>
<td>27</td>
</tr>
<tr>
<td>6 Hot Point Gate</td>
<td>28</td>
</tr>
</tbody>
</table>

Figure D 5 - Gates design solutions

### Ejection_DS: 11 Items

<table>
<thead>
<tr>
<th>DS_Name</th>
<th>DS_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Normal Ejector Pin</td>
<td>1</td>
</tr>
<tr>
<td>2 D-Shaped Ejector Pin</td>
<td>3</td>
</tr>
<tr>
<td>3 Sleeve Ejector Pin</td>
<td>4</td>
</tr>
<tr>
<td>4 Stripper Ring</td>
<td>10</td>
</tr>
<tr>
<td>5 Stripper Plate</td>
<td>11</td>
</tr>
<tr>
<td>6 Reverse Sprue Puller - below P/L</td>
<td>12</td>
</tr>
<tr>
<td>7 Z-Type Sprue Puller - below P/L</td>
<td>13</td>
</tr>
<tr>
<td>8 Mushroom Sprue Puller - above P/L</td>
<td>14</td>
</tr>
<tr>
<td>9 Normal Ejector Pin - Runner Puller</td>
<td>15</td>
</tr>
<tr>
<td>10 Reverse Runner Puller - below P/L</td>
<td>16</td>
</tr>
<tr>
<td>11 Mushroom Runner Puller - above P/L</td>
<td>17</td>
</tr>
</tbody>
</table>

Figure D 6 - Ejection design solutions
<table>
<thead>
<tr>
<th>Function_Name</th>
<th>DS_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed Impression</td>
<td>Sprue Gate</td>
</tr>
<tr>
<td>2</td>
<td>Rectangular Gate</td>
</tr>
<tr>
<td>3</td>
<td>Diaphragm Gate</td>
</tr>
<tr>
<td>4</td>
<td>Tunnel/Submarine Gate</td>
</tr>
<tr>
<td>5</td>
<td>Pin Point Gate</td>
</tr>
<tr>
<td>6</td>
<td>Hot Point Gate</td>
</tr>
<tr>
<td>7 Eject Rib</td>
<td>Normal Ejector Pin</td>
</tr>
<tr>
<td>8 Distribute Impressions</td>
<td>One Impression Layout</td>
</tr>
<tr>
<td>9</td>
<td>Circular Impression Distribution</td>
</tr>
<tr>
<td>10</td>
<td>In Line Impression Distribution</td>
</tr>
<tr>
<td>11</td>
<td>Matrix Impression Distribution</td>
</tr>
<tr>
<td>12 Eject Impression - by Wall</td>
<td>Normal Ejector Pin</td>
</tr>
<tr>
<td>13</td>
<td>D-Shaped Ejector Pin</td>
</tr>
<tr>
<td>14</td>
<td>Sleeve Ejector Pin</td>
</tr>
<tr>
<td>15</td>
<td>Stripper Ring</td>
</tr>
<tr>
<td>16</td>
<td>Stripper Plate</td>
</tr>
<tr>
<td>17 Eject Impression - by Base</td>
<td>Normal Ejector Pin</td>
</tr>
<tr>
<td>18 Eject Sprue</td>
<td>Reverse Sprue Puller - bellow P/L</td>
</tr>
<tr>
<td>19</td>
<td>Z-Type Sprue Puller - bellow P/L</td>
</tr>
<tr>
<td>20</td>
<td>Mushroom Sprue Puller - above P/L</td>
</tr>
<tr>
<td>21 Eject Runner</td>
<td>Normal Ejector Pin - Runner Puller</td>
</tr>
<tr>
<td>22</td>
<td>Reverse Runner Puller - bellow P/L</td>
</tr>
<tr>
<td>23</td>
<td>Mushroom Runner Puller - above P/L</td>
</tr>
<tr>
<td>24 Eject Gate</td>
<td>Normal Ejector Pin - Runner Puller</td>
</tr>
<tr>
<td>25</td>
<td>Reverse Runner Puller - bellow P/L</td>
</tr>
<tr>
<td>26</td>
<td>Mushroom Runner Puller - above P/L</td>
</tr>
<tr>
<td>27 Distribute Runner Layout</td>
<td>Circular Spoke Layout</td>
</tr>
<tr>
<td>28</td>
<td>Circular Y-Layout</td>
</tr>
<tr>
<td>29</td>
<td>Rectangular S-Layout</td>
</tr>
<tr>
<td>30 InLine Unbalanced - T-Layout</td>
<td>Rectangular H-Layout</td>
</tr>
<tr>
<td>31</td>
<td>Rectangular X-Layout</td>
</tr>
<tr>
<td>32</td>
<td>Rectangular Y-Layout</td>
</tr>
<tr>
<td>33</td>
<td>Rectangular Unbalanced - H-Layout</td>
</tr>
<tr>
<td>34 Hot Runner Manifold</td>
<td>HR2</td>
</tr>
<tr>
<td>35 Hot Runner Manifold</td>
<td>HRST4</td>
</tr>
<tr>
<td>36 Hot Runner Manifold</td>
<td>HRSO4</td>
</tr>
<tr>
<td>37 Hot Runner Manifold</td>
<td>HR4M</td>
</tr>
<tr>
<td>38 Hot Runner Manifold</td>
<td>HR6</td>
</tr>
<tr>
<td>39 Hot Runner Manifold</td>
<td>HR6M</td>
</tr>
<tr>
<td>40 Control Runner Flow</td>
<td>Circular Section</td>
</tr>
<tr>
<td>41 Modified Trapezoidal Section - Move Plate</td>
<td>Modified Trapezoidal Section - Fixed Plate</td>
</tr>
<tr>
<td>42 Hexagonal Section</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td></td>
</tr>
</tbody>
</table>

Figure D 7 - Functions and design solutions association
<table>
<thead>
<tr>
<th>Interaction Name</th>
<th>Interaction Type</th>
<th>Interaction ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impressions = 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Impressions = 3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Impressions = 5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &gt;= 7</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Impressions = 6</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Type of Runner # Hot</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Impressions &gt;= 3</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>Max. Diam &lt;= 90</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>Impressions &lt;= 5</td>
<td>1</td>
<td>71</td>
</tr>
<tr>
<td>Max. Diam &lt;= 50</td>
<td>1</td>
<td>72</td>
</tr>
<tr>
<td>Impressions &lt;= 7</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>Max. Diam &lt;= 40</td>
<td>1</td>
<td>73</td>
</tr>
<tr>
<td>Impressions &gt;= 2</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Impressions &lt;= 8</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Impressions = 4</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Impressions = 2</td>
<td>1</td>
<td>35</td>
</tr>
<tr>
<td>Impressions &lt;= 3</td>
<td>1</td>
<td>74</td>
</tr>
<tr>
<td>Max. Diam &lt;= 60</td>
<td>1</td>
<td>75</td>
</tr>
<tr>
<td>Impressions &lt;= 4</td>
<td>1</td>
<td>76</td>
</tr>
<tr>
<td>General Shape = Non Rotational</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>Impressions &lt;= 6</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>Max. Diam &lt;= 20</td>
<td>1</td>
<td>77</td>
</tr>
<tr>
<td>Impressions = 10</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>Max. Diam &lt;= 10</td>
<td>1</td>
<td>78</td>
</tr>
<tr>
<td>Length/Width Ratio &gt;= 3.5</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>Impressions = 8</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Mould Configuration = 2-P</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Runner type = cold</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>Material = PVC</td>
<td>1</td>
<td>83</td>
</tr>
<tr>
<td>P/L = Base</td>
<td>1</td>
<td>48</td>
</tr>
<tr>
<td>P/L # Base</td>
<td>1</td>
<td>49</td>
</tr>
<tr>
<td>Max. Diam &gt; 60</td>
<td>1</td>
<td>90</td>
</tr>
<tr>
<td>Impressions &gt;= 6</td>
<td>1</td>
<td>58</td>
</tr>
<tr>
<td>Feed Point = Top</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>Degating Type = Manual</td>
<td>1</td>
<td>28</td>
</tr>
<tr>
<td>Feed Point = P/L</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>Mould Configuration = 3-P</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>Degating Type = Auto-Degating</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>General Shape = Tubular</td>
<td>1</td>
<td>31</td>
</tr>
<tr>
<td>Impressions &gt;= 1</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>Runner Type = Hot</td>
<td>1</td>
<td>36</td>
</tr>
<tr>
<td>Mould Type = Moss Standard</td>
<td>1</td>
<td>37</td>
</tr>
<tr>
<td>Base Wall Thickness &gt;= 2</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>Base Wall Thickness &gt;= 3</td>
<td>1</td>
<td>81</td>
</tr>
<tr>
<td>Height/Radius Ratio &lt;= 1.2</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td>Material # PVC</td>
<td>1</td>
<td>84</td>
</tr>
<tr>
<td>General Shape = Rotational</td>
<td>1</td>
<td>88</td>
</tr>
<tr>
<td>Base Wall Thickness =&gt; 1.5</td>
<td>1</td>
<td>85</td>
</tr>
<tr>
<td>Base Diameter &lt;= 20</td>
<td>1</td>
<td>86</td>
</tr>
<tr>
<td>Base Diameter &lt;= 6</td>
<td>1</td>
<td>95</td>
</tr>
<tr>
<td>Impression &gt; 4</td>
<td>1</td>
<td>91</td>
</tr>
</tbody>
</table>

Figure D 8 - Numerical interactions
### Appendix D

#### Existence_Interaction: 26 Items

<table>
<thead>
<tr>
<th>Interaction Name</th>
<th>Interaction ID</th>
<th>Interaction Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Runner Layout = Circular-Spoke</td>
<td>46</td>
<td>2</td>
</tr>
<tr>
<td>2 Runner Layout = Circular-T</td>
<td>47</td>
<td>2</td>
</tr>
<tr>
<td>3 Runner Layout = Circular</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>4 Runner Layout = S-Type</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>5 Runner Layout = T-Unbalanced</td>
<td>42</td>
<td>2</td>
</tr>
<tr>
<td>6 Runner Layout = HR2</td>
<td>43</td>
<td>2</td>
</tr>
<tr>
<td>7 Runner Layout = HR4ST</td>
<td>44</td>
<td>2</td>
</tr>
<tr>
<td>8 Runner Layout # S Type</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>9 Runner Layout # Circular-Spoke</td>
<td>38</td>
<td>2</td>
</tr>
<tr>
<td>10 Runner Layout # Circular-T</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td>11 Ejection neq. Stripper Plate</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>12 Ejection neq. Stripper Ring</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>13 Distribution = Circular</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>14 Cross Section # Trapezoidal - Movell Plate</td>
<td>53</td>
<td>2</td>
</tr>
<tr>
<td>15 Cross Section # Trapezoidal Modified - Movell Plate</td>
<td>55</td>
<td>2</td>
</tr>
<tr>
<td>16 Cross Section # Semi-Circular - Movell Plate</td>
<td>56</td>
<td>2</td>
</tr>
<tr>
<td>17 Distribution = InLine</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>18 Distribution = Matrix</td>
<td>45</td>
<td>2</td>
</tr>
<tr>
<td>19 Cross Section == Trapezoidal - Movell Plate</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>20 Cross Section == Trapezoidal Modified - Movell Plate</td>
<td>51</td>
<td>2</td>
</tr>
<tr>
<td>21 Cross Section == Semi-Circular - Movell Plate</td>
<td>52</td>
<td>2</td>
</tr>
<tr>
<td>22 Cross Section == Trapezoidal - Fixed Plate</td>
<td>61</td>
<td>2</td>
</tr>
<tr>
<td>23 Cross Section == Trapezoidal Modified - Fixed Plate</td>
<td>59</td>
<td>2</td>
</tr>
<tr>
<td>24 Cross Section == Semi-Circular - Fixed Plate</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>25 Ejection == Stripper Plate</td>
<td>54</td>
<td>2</td>
</tr>
<tr>
<td>26 No Internal Cooling</td>
<td>87</td>
<td>2</td>
</tr>
</tbody>
</table>

#### OR_Interaction: 20 Items

<table>
<thead>
<tr>
<th>Interaction Name</th>
<th>Interaction ID</th>
<th>Interaction Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Impressions = 3, 5, or &gt;=7)</td>
<td>51000</td>
<td>4</td>
</tr>
<tr>
<td>2 Runner = Circular</td>
<td>51001</td>
<td>4</td>
</tr>
<tr>
<td>3 (Impressions = 3, 5, or &gt;=7) OR (Runner = Circular)</td>
<td>51002</td>
<td>4</td>
</tr>
<tr>
<td>4 (Number Impressions) OR (Max Diameter)</td>
<td>51003</td>
<td>4</td>
</tr>
<tr>
<td>5 Runner == InLine</td>
<td>52001</td>
<td>4</td>
</tr>
<tr>
<td>6 (Number Impressions) OR (Max Diameter)</td>
<td>52003</td>
<td>4</td>
</tr>
<tr>
<td>7 Impressions = 6,8 or 10</td>
<td>31000</td>
<td>4</td>
</tr>
<tr>
<td>8 Impressions = 4,6 or 8</td>
<td>55000</td>
<td>4</td>
</tr>
<tr>
<td>9 Runner # Circular</td>
<td>55001</td>
<td>4</td>
</tr>
<tr>
<td>10 (Number Impressions) OR (Max Diameter)</td>
<td>55003</td>
<td>4</td>
</tr>
<tr>
<td>11 PVC Material &amp; Wall Thickness</td>
<td>10000</td>
<td>4</td>
</tr>
<tr>
<td>12 Ejection # Stripper</td>
<td>40000</td>
<td>4</td>
</tr>
<tr>
<td>13 Impressions = 4 or 8</td>
<td>36000</td>
<td>4</td>
</tr>
<tr>
<td>14 (2-P &amp; 1 Impression) OR (3-P &amp; Impressions&gt;2)</td>
<td>23002</td>
<td>4</td>
</tr>
<tr>
<td>15 Runner Half Section == Movell Plate</td>
<td>15000</td>
<td>4</td>
</tr>
<tr>
<td>16 (Impression = 1 &amp; Feeding Point == P/L) OR Impression &gt;2</td>
<td>12001</td>
<td>4</td>
</tr>
<tr>
<td>17 Runner Half Section == Fixed Plate</td>
<td>14000</td>
<td>4</td>
</tr>
<tr>
<td>18 Ejection == Stripper Plate OR Mould Config. == 3-P</td>
<td>14001</td>
<td>4</td>
</tr>
<tr>
<td>19 Runner Half Section # Movell Plate</td>
<td>16000</td>
<td>4</td>
</tr>
<tr>
<td>20 (Number Impressions) &amp;&amp; (Max Diameter)</td>
<td>11000</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure D 9 - Existence interactions

Figure D 10 - Composite "OR" interactions
### Appendix D

#### AND Interaction: 22 Items

<table>
<thead>
<tr>
<th>Interaction Name</th>
<th>Interaction ID</th>
<th>Interaction Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impressions = 3 &amp; Max Diameter &lt;= 80</td>
<td>51004</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 5 &amp; Max Diameter &lt;= 50</td>
<td>51005</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 7 &amp; Max Diameter &lt;= 40</td>
<td>51006</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &gt;= 2 &amp; &lt; 8</td>
<td>52000</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 6 &amp; Max Diameter &lt;= 20</td>
<td>52007</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 8 &amp; Max Diameter &lt;= 10</td>
<td>52008</td>
<td>3</td>
</tr>
<tr>
<td>Impressions = 2 &amp; Max Diameter &lt;= 90</td>
<td>52004</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 3 &amp; Max Diameter &lt;= 60</td>
<td>52005</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 4 &amp; Max Diameter &lt;= 40</td>
<td>52006</td>
<td>3</td>
</tr>
<tr>
<td>Box Type Component &amp; Lenght/Width Ratio &gt;= 3.5</td>
<td>33000</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 4 &amp; Max Diameter &lt;= 40</td>
<td>55006</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 6 &amp; Max Diameter &lt;= 60</td>
<td>55007</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 8 &amp; Max Diameter &lt;= 40</td>
<td>55008</td>
<td>3</td>
</tr>
<tr>
<td>(Base Wall Thickness &gt;= 3 &amp; Height/Radius Ratio &lt;= 1.2)</td>
<td>10001</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &gt;= 3 &amp; &lt; 7</td>
<td>30000</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &gt; 4 &amp; Max Diameter &lt;= 60</td>
<td>11002</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &gt;= 2 &amp; &lt;= 8</td>
<td>35000</td>
<td>3</td>
</tr>
<tr>
<td>Mould_Config 2-P &amp; Impressions &gt;= 1</td>
<td>23000</td>
<td>3</td>
</tr>
<tr>
<td>Mould_Config 3-P &amp; Impressions &gt;= 2</td>
<td>23001</td>
<td>3</td>
</tr>
<tr>
<td>Impression = 1 &amp; Feeding Point == P/L</td>
<td>12000</td>
<td>3</td>
</tr>
<tr>
<td>(Base Diameter &gt;= 6 &amp; &lt;= 20)</td>
<td>40020</td>
<td>3</td>
</tr>
<tr>
<td>Impressions &lt;= 4 &amp; Max Diameter &gt; 60</td>
<td>11001</td>
<td>3</td>
</tr>
</tbody>
</table>

---

**Figure D 11 - Composite "AND" interactions**
<table>
<thead>
<tr>
<th>DS_Name</th>
<th>Interaction Name</th>
<th>Interaction ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Ejector Pin</td>
<td>Base Wall Thickness &gt; 2</td>
<td>80</td>
</tr>
<tr>
<td>PVC Material &amp; Wall Thickness</td>
<td>P/L = Base</td>
<td>48</td>
</tr>
<tr>
<td>D-Shaped Ejector Pin</td>
<td>General Shape = Non Rotational</td>
<td>22</td>
</tr>
<tr>
<td>Sleeve Ejector Pin</td>
<td>General Shape = Rotational</td>
<td>88</td>
</tr>
<tr>
<td>Base Wall Thickness &gt; 1.5</td>
<td>(Base Diameter &gt; 6 &amp; &lt;= 29)</td>
<td>85</td>
</tr>
<tr>
<td>PVC Material &amp; Well Thickness</td>
<td>P/L = Base</td>
<td>49</td>
</tr>
<tr>
<td>Stripper Plate</td>
<td>P/L = Base</td>
<td>48</td>
</tr>
<tr>
<td>Reverse Sprue Puller - bellow P/L</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>Impression = 1 &amp; Feeding Point = P/L</td>
<td>OR Impression &gt;= 2</td>
<td>12001</td>
</tr>
<tr>
<td>Ejection neg. Stripper Plate</td>
<td>Ejection = P/L</td>
<td>17</td>
</tr>
<tr>
<td>2-Type Sprue Puller - bellow P/L</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>Impression = 1 &amp; Feeding Point = P/L</td>
<td>OR Impression &gt;= 2</td>
<td>12001</td>
</tr>
<tr>
<td>Mushroom Sprue Puller - above P/L</td>
<td>Ejection neg. Stripper Plate = cold</td>
<td>17</td>
</tr>
<tr>
<td>Normal Ejector Pin - Runner Puller</td>
<td>Runner Half Section = Fixed Plate</td>
<td>14000</td>
</tr>
<tr>
<td>Impression = 1 &amp; Feeding Point = P/L</td>
<td>OR Impression &gt;= 2</td>
<td>12001</td>
</tr>
<tr>
<td>Ejection neg. Stripper Plate</td>
<td>Ejection = P/L</td>
<td>17</td>
</tr>
<tr>
<td>Reverse Runner Puller - bellow P/L</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>Impression = 1 &amp; Feeding Point = P/L</td>
<td>OR Impression &gt;= 2</td>
<td>12001</td>
</tr>
<tr>
<td>Ejection neg. Stripper Plate</td>
<td>Ejection = P/L</td>
<td>17</td>
</tr>
<tr>
<td>Mushroom Runner Puller - above P/L</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>Impression = 1 &amp; Feeding Point = P/L</td>
<td>OR Impression &gt;= 2</td>
<td>12001</td>
</tr>
<tr>
<td>Ejection neg. Stripper Plate</td>
<td>Ejection = P/L</td>
<td>17</td>
</tr>
<tr>
<td>Sprue Gate</td>
<td>Impressions &gt;= 1</td>
<td>1</td>
</tr>
<tr>
<td>Mould Configuration = 2-P</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Feed Point = Top</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>Degating Type = Manual</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Rectangular Gate</td>
<td>Mould Configuration = 2-P</td>
<td>14</td>
</tr>
<tr>
<td>Runner type = cold</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Feed Point = P/L</td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>Diaphragm Gate</td>
<td>(2-P &amp; 1 Impression) OR (3-P &amp; 2 Impressions)</td>
<td>23002</td>
</tr>
<tr>
<td>Feed Point = Top</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>Runner type = cold</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>General Shape = Tubular</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>Tunnel/Submarine Gate</td>
<td>Impressions &gt; 1</td>
<td>33</td>
</tr>
<tr>
<td>Runner type = cold</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Mould Configuration = 2-P</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Feed Point = Top</td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>Degating Type = Auto-Degating</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Pin Point Gate</td>
<td>Ejection = # Stripper</td>
<td>40000</td>
</tr>
<tr>
<td>Runner type = cold</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Mould Configuration = 3-P</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>Feed Point = Top</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>Degating Type = Auto-Degating</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Hot Runner Manifold - HR2</td>
<td>Runner Type = Hot</td>
<td>36</td>
</tr>
<tr>
<td>Impressions = 2</td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>Hot Point Gate</td>
<td>Runner Type = Hot</td>
<td>36</td>
</tr>
<tr>
<td>Feed Point = Top</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>Degating Type = Auto-Degating</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Circular Spoke Layout</td>
<td>Distribution = Circular</td>
<td>25</td>
</tr>
<tr>
<td>Mould Configuration = 2-P</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Runner type = cold</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Impressions &gt; 3 &amp; &lt;= 7</td>
<td></td>
<td>30000</td>
</tr>
</tbody>
</table>

Figure D 12 - Interactions associated to design solutions
<table>
<thead>
<tr>
<th>DS_Name</th>
<th>Interaction_Name</th>
<th>Interaction_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>62 Circular Y-Layout</td>
<td>Distribution = Circular</td>
<td>25</td>
</tr>
<tr>
<td>63</td>
<td>Impressions = 6,8 or 10</td>
<td>31000</td>
</tr>
<tr>
<td>64</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>65 Rectangular S-Layout</td>
<td>Distribution = 2-P</td>
<td>35</td>
</tr>
<tr>
<td>66</td>
<td>Runner type = cold</td>
<td>14</td>
</tr>
<tr>
<td>67</td>
<td>Mould Configuration = 2-P</td>
<td>14</td>
</tr>
<tr>
<td>68 InLine Unbalanced - T-Layout</td>
<td>Box Type Component &amp; Length/Width Ratio &gt;&gt; 3.5</td>
<td>33000</td>
</tr>
<tr>
<td>69</td>
<td>Impressions &gt;=2 &amp; &lt;= 8</td>
<td>35000</td>
</tr>
<tr>
<td>70</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>71 Rectangular X-Layout</td>
<td>Distribution = Matrix</td>
<td>45</td>
</tr>
<tr>
<td>72</td>
<td>Runner type = cold</td>
<td>14</td>
</tr>
<tr>
<td>73 Rectangular H-Layout</td>
<td>Distribution = Matrix</td>
<td>45</td>
</tr>
<tr>
<td>74</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>75 Rectangular Y-Layout</td>
<td>Distribution = Matrix</td>
<td>45</td>
</tr>
<tr>
<td>76</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>77 Rectangular Unbalanced - H-Layout</td>
<td>Distribution = Matrix</td>
<td>45</td>
</tr>
<tr>
<td>78</td>
<td>Runner type = cold</td>
<td>14</td>
</tr>
<tr>
<td>79 Hot Runner Manifold - HRST4</td>
<td>Impressions = 6</td>
<td>10</td>
</tr>
<tr>
<td>80 Hot Runner Manifold - HRS04</td>
<td>Runner Type = Hot</td>
<td>36</td>
</tr>
<tr>
<td>81 Hot Runner Manifold - HR4M</td>
<td>Distribution = In-Line</td>
<td>40</td>
</tr>
<tr>
<td>82 Hot Runner Manifold - HR6</td>
<td>Impressions = 6</td>
<td>10</td>
</tr>
<tr>
<td>83 Hot Runner Manifold - HR6M</td>
<td>Runner Type = Hot</td>
<td>36</td>
</tr>
<tr>
<td>84 Hot Runner Manifold - HR6</td>
<td>Distribution = Matrix</td>
<td>45</td>
</tr>
<tr>
<td>85 Hot Runner Manifold - HR6M</td>
<td>Mould Type = Moss Standard</td>
<td>37</td>
</tr>
<tr>
<td>86 Circular Section</td>
<td>Impressions = 6</td>
<td>5</td>
</tr>
<tr>
<td>87 Hot Runner Manifold - HR6</td>
<td>Runner Type = Hot</td>
<td>36</td>
</tr>
<tr>
<td>88 Hot Runner Manifold - HR6M</td>
<td>Distribution = Matrix</td>
<td>45</td>
</tr>
<tr>
<td>89 Hot Runner Manifold - HR6</td>
<td>Mould Type = Moss Standard</td>
<td>37</td>
</tr>
<tr>
<td>90 Hot Runner Manifold - HR6M</td>
<td>Impressions &lt;= 6</td>
<td>15</td>
</tr>
<tr>
<td>91 Hot Runner Manifold - HR6</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>92 Hot Runner Manifold - HR6M</td>
<td>Runner Layout # S Type</td>
<td>21</td>
</tr>
<tr>
<td>93 Hot Runner Manifold - HR6</td>
<td>Ejection # / Striper</td>
<td>40000</td>
</tr>
<tr>
<td>94 Hot Runner Manifold - HR6M</td>
<td>P/L # Base</td>
<td>49</td>
</tr>
<tr>
<td>95 Hexagonal Section</td>
<td>Impressions &gt;= 2</td>
<td>7</td>
</tr>
<tr>
<td>96 Modified Trapezoidal Section - Movable Plate</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>97</td>
<td>Mould Configuration = 2-P</td>
<td>14</td>
</tr>
<tr>
<td>98 Modified Trapezoidal Section - Movable Plate</td>
<td>Runner Layout # S Type</td>
<td>21</td>
</tr>
<tr>
<td>99 Modified Trapezoidal Section - Fixed Plate</td>
<td>Ejection # / Striper</td>
<td>40000</td>
</tr>
<tr>
<td>100</td>
<td>Builder # Base</td>
<td>49</td>
</tr>
<tr>
<td>101</td>
<td>Impressions &gt;= 2</td>
<td>7</td>
</tr>
<tr>
<td>102</td>
<td>Runner type = cold</td>
<td>16</td>
</tr>
<tr>
<td>103</td>
<td>Mould Configuration = 2-P</td>
<td>14</td>
</tr>
<tr>
<td>104 Mobile Plate</td>
<td>Runner Layout # S Type</td>
<td>21</td>
</tr>
<tr>
<td>105 Mobile Plate</td>
<td>Ejection # / Striper</td>
<td>40000</td>
</tr>
<tr>
<td>106 Mobile Plate</td>
<td>P/L # Base</td>
<td>49</td>
</tr>
<tr>
<td>107 Mobile Plate</td>
<td>Impressions = 1</td>
<td>1</td>
</tr>
<tr>
<td>108 Mobile Plate</td>
<td>(Impressions &gt;= 3, 5, or &gt;6) OR (Runner = Circular)</td>
<td>51002</td>
</tr>
<tr>
<td>109 Mobile Plate</td>
<td>Type_of_Runner # Hot</td>
<td>11</td>
</tr>
<tr>
<td>110 Mobile Plate</td>
<td>Impressions = 3</td>
<td>26</td>
</tr>
<tr>
<td>111 Mobile Plate</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>51003</td>
</tr>
<tr>
<td>112 Mobile Plate</td>
<td>Impressions &gt;=2 &amp; &lt;= 8</td>
<td>52000</td>
</tr>
<tr>
<td>113 Mobile Plate</td>
<td>Runner = In_Line</td>
<td>52001</td>
</tr>
<tr>
<td>114 Mobile Plate</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>52003</td>
</tr>
<tr>
<td>115 Mobile Plate</td>
<td>Runner # Circular</td>
<td>55000</td>
</tr>
<tr>
<td>116 Mobile Plate</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>55003</td>
</tr>
<tr>
<td>117 Mobile Plate</td>
<td>Impressions = 3, 5, or =&gt;7 OR (Runner = Circular)</td>
<td>51002</td>
</tr>
<tr>
<td>118 Mobile Plate</td>
<td>Type_of_Runner # Hot</td>
<td>11</td>
</tr>
<tr>
<td>119 Mobile Plate</td>
<td>Impressions = 3</td>
<td>26</td>
</tr>
<tr>
<td>120 Mobile Plate</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>51003</td>
</tr>
<tr>
<td>121 Mobile Plate</td>
<td>Impressions &gt;=2 &amp; &lt;= 8</td>
<td>52000</td>
</tr>
<tr>
<td>122 Mobile Plate</td>
<td>Runner = In_Line</td>
<td>52001</td>
</tr>
<tr>
<td>123 Mobile Plate</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>52003</td>
</tr>
<tr>
<td>124 Mobile Plate</td>
<td>Runner # Circular</td>
<td>55000</td>
</tr>
<tr>
<td>125 Mobile Plate</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>55003</td>
</tr>
<tr>
<td>126 Mobile Plate</td>
<td>Ejection # / Striper</td>
<td>40000</td>
</tr>
<tr>
<td>127 Mobile Plate</td>
<td>P/L # Base</td>
<td>49</td>
</tr>
<tr>
<td>128 Mobile Plate</td>
<td>Impressions = 1</td>
<td>1</td>
</tr>
<tr>
<td>129 Mobile Plate</td>
<td>(Impressions &gt;= 3, 5, or &gt;6) OR (Runner = Circular)</td>
<td>51002</td>
</tr>
<tr>
<td>130 Mobile Plate</td>
<td>Type_of_Runner # Hot</td>
<td>11</td>
</tr>
<tr>
<td>131 Mobile Plate</td>
<td>Impressions = 3</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>51003</td>
</tr>
<tr>
<td></td>
<td>Runner = In_Line</td>
<td>52001</td>
</tr>
<tr>
<td></td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>52003</td>
</tr>
<tr>
<td></td>
<td>Runner # Circular</td>
<td>55000</td>
</tr>
<tr>
<td></td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>55003</td>
</tr>
</tbody>
</table>

Figure D 12 - Interactions associated to design solutions (cont.)
<table>
<thead>
<tr>
<th>Interaction_Name</th>
<th>Interaction_Type</th>
<th>Interaction_Name</th>
<th>Interaction_Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Impressions = 3, 5, or &gt;= 7</td>
<td>4</td>
<td>Impressions = 3, 5, or &gt;= 7</td>
</tr>
<tr>
<td>2</td>
<td>Impressions = 5</td>
<td>1</td>
<td>Impressions = 5</td>
</tr>
<tr>
<td>3</td>
<td>Impressions =&gt; 7</td>
<td>1</td>
<td>Impressions =&gt; 7</td>
</tr>
<tr>
<td>4</td>
<td>Runner = Circular</td>
<td>4</td>
<td>Runner Layout = Circular-Spoke</td>
</tr>
<tr>
<td>5</td>
<td>Runner Layout = Circular-T</td>
<td>2</td>
<td>Runner Layout = Circular-T</td>
</tr>
<tr>
<td>6</td>
<td>(Impressions = 3, 5, or &gt;= 7) OR (Runner = Circular)</td>
<td>4</td>
<td>(Impressions = 3, 5, or &gt;= 7)</td>
</tr>
<tr>
<td>7</td>
<td>Runner = Circular</td>
<td>4</td>
<td>Runner = Circular</td>
</tr>
<tr>
<td>8</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>4</td>
<td>Impressions = 3 &amp; &amp; Max Diameter &lt;= 80</td>
</tr>
<tr>
<td>9</td>
<td>Impressions &lt;= 5 &amp; &amp; Max Diameter &lt;= 50</td>
<td>3</td>
<td>Impressions &lt;= 5 &amp; &amp; Max Diameter &lt;= 50</td>
</tr>
<tr>
<td>10</td>
<td>Impressions &lt;= 7 &amp; &amp; Max Diameter &lt;= 40</td>
<td>3</td>
<td>Impressions &lt;= 7 &amp; &amp; Max Diameter &lt;= 40</td>
</tr>
<tr>
<td>11</td>
<td>Impressions = 3 &amp; &amp; Max Diameter &lt;= 80</td>
<td>1</td>
<td>Impressions = 3 &amp; &amp; Max Diameter &lt;= 80</td>
</tr>
<tr>
<td>12</td>
<td>Max Diam &lt;= 90</td>
<td>1</td>
<td>Max Diam &lt;= 90</td>
</tr>
<tr>
<td>13</td>
<td>Impressions &lt;= 5 &amp; &amp; Max Diameter &lt;= 50</td>
<td>1</td>
<td>Impressions &lt;= 5 &amp; &amp; Max Diameter &lt;= 50</td>
</tr>
<tr>
<td>14</td>
<td>Max Diam &lt;= 50</td>
<td>1</td>
<td>Max Diam &lt;= 50</td>
</tr>
<tr>
<td>15</td>
<td>Runner =&gt; In_Line</td>
<td>4</td>
<td>Runner Layout = S-Type</td>
</tr>
<tr>
<td>16</td>
<td>Runner Layout = T-Unbalanced</td>
<td>2</td>
<td>Runner Layout = T-Unbalanced</td>
</tr>
<tr>
<td>17</td>
<td>Runner Layout = HR2</td>
<td>2</td>
<td>Runner Layout = HR2</td>
</tr>
<tr>
<td>18</td>
<td>Runner Layout = HR4ST</td>
<td>2</td>
<td>Runner Layout = HR4ST</td>
</tr>
<tr>
<td>19</td>
<td>Impressions &lt;= 7 &amp; &amp; Max Diameter &lt;= 40</td>
<td>3</td>
<td>Impressions &lt;= 7 &amp; &amp; Max Diameter &lt;= 40</td>
</tr>
<tr>
<td>20</td>
<td>Max Diam &lt;= 40</td>
<td>1</td>
<td>Max Diam &lt;= 40</td>
</tr>
<tr>
<td>21</td>
<td>Impressions =&gt; 2 &amp; &amp; &lt;= 8</td>
<td>1</td>
<td>Impressions =&gt; 2 &amp; &amp; &lt;= 8</td>
</tr>
<tr>
<td>22</td>
<td>Impressions &lt;= 8</td>
<td>1</td>
<td>Impressions &lt;= 8</td>
</tr>
<tr>
<td>23</td>
<td>Impressions &lt;= 6 &amp; &amp; Max Diameter &lt;= 20</td>
<td>3</td>
<td>Impressions &lt;= 6 &amp; &amp; Max Diameter &lt;= 20</td>
</tr>
<tr>
<td>24</td>
<td>Max Diam &lt;= 20</td>
<td>1</td>
<td>Max Diam &lt;= 20</td>
</tr>
<tr>
<td>25</td>
<td>Impressions &lt;= 8 &amp; &amp; Max Diameter &lt;= 10</td>
<td>3</td>
<td>Impressions &lt;= 8 &amp; &amp; Max Diameter &lt;= 10</td>
</tr>
<tr>
<td>26</td>
<td>Max Diam &lt;= 10</td>
<td>1</td>
<td>Max Diam &lt;= 10</td>
</tr>
<tr>
<td>27</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>4</td>
<td>Impressions &lt;= 2 &amp; &amp; Max Diameter &lt;= 90</td>
</tr>
<tr>
<td>28</td>
<td>Impressions &lt;= 3 &amp; &amp; Max Diameter &lt;= 60</td>
<td>3</td>
<td>Impressions &lt;= 3 &amp; &amp; Max Diameter &lt;= 60</td>
</tr>
<tr>
<td>29</td>
<td>Impressions &lt;= 4 &amp; &amp; Max Diameter &lt;= 40</td>
<td>3</td>
<td>Impressions &lt;= 4 &amp; &amp; Max Diameter &lt;= 40</td>
</tr>
<tr>
<td>30</td>
<td>Impressions &lt;= 6 &amp; &amp; Max Diameter &lt;= 20</td>
<td>3</td>
<td>Impressions &lt;= 6 &amp; &amp; Max Diameter &lt;= 20</td>
</tr>
<tr>
<td>31</td>
<td>Impressions &lt;= 8 &amp; &amp; Max Diameter &lt;= 10</td>
<td>3</td>
<td>Impressions &lt;= 8 &amp; &amp; Max Diameter &lt;= 10</td>
</tr>
</tbody>
</table>

Figure D 13 - Interactions associated with composite interactions
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Impressions = 2 &amp; Max Diameter &lt;= 90</td>
<td>3 Impressions = 2</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>Max Diam &lt;= 90</td>
</tr>
<tr>
<td>34</td>
<td>Impressions &lt;= 3 &amp; Max Diameter &lt;= 60</td>
<td>3 Impressions &lt;= 3</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>Max Diam &lt;= 60</td>
</tr>
<tr>
<td>36</td>
<td>Impressions &lt;= 4 &amp; Max Diameter &lt;= 40</td>
<td>3 Impressions &lt;= 4</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>Max Diam &lt;= 40</td>
</tr>
<tr>
<td>38</td>
<td>Box Type Component &amp; Length/Width Ratio &gt;= 3.5</td>
<td>3 General Shape = Non Rotational</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>Length/Width Ratio &gt;= 3.5</td>
</tr>
<tr>
<td>40</td>
<td>Impressions = 6, 8 or 10</td>
<td>4 Impressions = 6</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>Impressions = 8</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td>Impressions = 10</td>
</tr>
<tr>
<td>43</td>
<td>Impressions = 4, 6 or 8</td>
<td>4 Impressions = 4</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td>Impressions = 6</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td>Impressions = 8</td>
</tr>
<tr>
<td>46</td>
<td>Runner # Circular</td>
<td>4 Runner Layout # Circular-Spoke</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td>Runner Layout # Circular-T</td>
</tr>
<tr>
<td>48</td>
<td>(Number Impressions) OR (Max Diameter)</td>
<td>4 Impressions &lt;= 4 &amp; Max Diameter &lt;= 40</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td>Impressions &lt;= 6 &amp; Max Diameter &lt;= 60</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>Impressions &lt;= 8 &amp; Max Diameter &lt;= 60</td>
</tr>
<tr>
<td>51</td>
<td>Impressions &lt;= 4 &amp; Max Diameter &lt;= 40</td>
<td>3 Impressions &lt;= 4</td>
</tr>
<tr>
<td>52</td>
<td></td>
<td>Max Diam &lt;= 40</td>
</tr>
<tr>
<td>53</td>
<td>Impressions &lt;= 6 &amp; Max Diameter &lt;= 60</td>
<td>3 Impressions &lt;= 6</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td>Max Diam &lt;= 60</td>
</tr>
<tr>
<td>55</td>
<td>Impressions &lt;= 8 &amp; Max Diameter &lt;= 40</td>
<td>3 Impressions &lt;= 2</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td>Max Diam &lt;= 40</td>
</tr>
<tr>
<td>57</td>
<td>PVC Material &amp; Wall Thickness</td>
<td>4 (Base Wall Thickness &gt;= 3 &amp; Height/Radius Ratio &lt;= 1.2)</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td>Material # PVC</td>
</tr>
<tr>
<td>59</td>
<td>(Base Wall Thickness &gt;= 3 &amp; Height/Radius Ratio &lt;= 1.2)</td>
<td>3 Material == PVC</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>Base Wall Thickness &gt;= 3</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td>Height/Radius Ratio &lt;= 1.2</td>
</tr>
</tbody>
</table>

Figure D 13 - Interactions associated with composite interactions (Cont.)
<table>
<thead>
<tr>
<th>Ejection No</th>
<th>Impression &gt; 4</th>
<th>Impression &gt; 3 &amp; &lt; 4</th>
<th>Impression &gt; 2 &amp; &lt; 4</th>
<th>Impression &gt; 2 &amp; &lt; 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>