Integrated Information Model For Managing The Product Introduction Process

DEVI THIRUPATHI M.C.A., M.Phil.

Submitted for the Degree of Doctor of Philosophy

Department of Engineering

University of Warwick

Sep 1998

To

my husband *Ravi*, my daughter *Deepiga*,
my mother *Suseela*, fond memories of my father *Thirupathi*,
my *Teachers* and Lord *Muruga*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The interest and support of all those who have helped me in making my dream come true is highly appreciated. It has been a pleasure to have had the opportunity to study research at University of Warwick. The Warwick Manufacturing Group has been most supportive and it has been a privilege to work in such a stimulating environment.

I sincerely wish to express my deep sense of gratitude to Mr.R.Roy, my research supervisor, who has given me immense guidance and inspiration for the fruitful completion of my research work.

I also wish to acknowledge the Commonwealth Scholarships Commission in the United Kingdom, British Council and Warwick Manufacturing Group for financial support to carry out this research. Thanks to the Ministry of Human Resource Development and Bharathiar University, India for providing the necessary study leave.

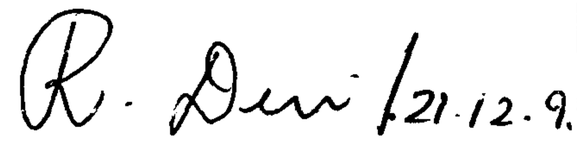
Thanks are due to Mr.P.Riley for the discussions on the product introduction process. The help and encouragement from Prof.K.Sarukesi and his family are gratefully acknowledged. Thanks to the Simulation team for their help and many useful discussions. Thanks are due to Dr.K.Young and Dr.B.Wu for examining this research work. Thanks to my friends for their help.

Special thanks to my in-laws, sister and brothers for all their love and support. Finally, I thank my husband Ravi and daughter Deepiga, for their patience, personal help and encouragement throughout the whole venture.

DECLARATION

I declare that the work described in this Ph.D. thesis, unless otherwise acknowledged in the text, is my own work and has not been previously submitted for any academic degree.

Signed



Devi Thirupathi

SUMMARY

The thesis proposes an integrated product introduction (PI) information model for managing the product introduction process in an efficient manner. Through the process of product introduction, ideas and needs are converted to the information from which technical systems and products can be made and sold. Two critical factors for its success are the management of the product introduction activities, and the quality and functionality of its output (i.e. the product) which itself depends on the quality of the PI process. The process is as effective as the decisions made within it, and as efficient as the speed with which the information required for each decision is made available.

In order to improve the efficiency of the management of the project in meeting its diverse targets (↓project time, ↓project cost, ↓product cost and ↑product functionality), a model that integrates the targets would be essential in relating the activities of the project to their outcomes. Existing information models in related areas such as design, product development, project management, computer aided design and manufacturing consider some of these targets, but not all of them simultaneously. Especially product functionality is not considered along with the other targets of the PI project.

The product introduction information includes managerial and technical information and complex associations among these two categories. Its representation places a challenging and novel set of demands on database technology as it is evolving, distributed and heterogeneous. Existing information models do not address the link between the managerial and technical information, and their continual evolution. Based on a detailed analysis of its nature and content, the thesis presents a three dimensional model of the product introduction information from three related but different viewpoints:- ① entity-relationship or objects, ② intra-layer integration and ③ evolution, each capturing important aspects of the PI information, but all required for a complete description. The overall three dimensional information model includes the following layers:- from view 1 - product functionality, process or project, product introduction resources, product and information map; from view 2 - node, relationship, and organisation; from view 3 - meta-model, data model, and data. Each model describes one aspect of the product introduction information but contains references to the others. The building blocks of the information model are described using schema definitions.

The information model has been implemented in a prototype system. It captures the product introduction process information at a number of levels:- managerial level - project database, resources database; technical level - product functionality database, product database; relationship between managerial level and technical level - information map database. Individual data managers have been developed to manage these databases. The individual data models have been integrated into a single global model with the use of meta-modelling technique; a meta-model manager has been developed towards this end. To demonstrate the use of information model, an activity-dependency analysis tool and algorithms for communication acceleration that enable effective information flow have been developed.

LIST OF ABBREVIATIONS

ARIS	Architecture of Integrated Information Systems
BOM	Bill of Materials
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAPP	Computer Aided Process Planning
CE	Concurrent Engineering
CEDS	Concurrent Engineering Data Structure
CIM-OSA	Open System Architecture for Computer Integrated Manufacturing
CPM	Critical Path Method
DB	Database
DFA	Design For Assembly
DFD	Data Flow Diagram
ER	Entity Relationship
GIM	GRAI Integrated Methodology
IDEF	Integrated Computer-Aided Manufacturing (ICAM) DEFinition
IIM	Integrated Information Model
ISS	Information Sharing System
KOF	Knowledge-integrated Object-oriented Feature-based product definition model
LPDE	Leeds Product Data Editor
OO	Object-Oriented
OR	Object-Relational
PBS	Project Breakdown Structure
PDM	Precedence Diagramming Method
PERT	Program Evaluation and Review Technique

PI	Product Introduction
PSE	Product Structure Editor
QFD	Quality Function Deployment
SCOPES	Systematic Concurrent design of Products, Equipments and control Systems
STEP	STandard for the Exchange of Product model data
UPDE	Unified Product Data Model
WBS	Work Breakdown Structure

CHAPTER 1

1. INTRODUCTION

1.1 INTRODUCTION

This research is concerned with the study and analysis of the product introduction process and proposes an integrated information model for managing the product introduction process. Intense competition forces industries to introduce products at an increasingly rapid pace. This places substantial pressure on product introduction teams to introduce better products and at the same time to introduce products faster. Product Introduction (PI) is the process of bringing a new product to the market. This chapter introduces the product introduction process and a number of areas related to its management and control. These areas include - process control, project management, concurrent engineering, teamworking, communication and information flow, and information management.

1.2 PRODUCT INTRODUCTION PROCESS

Through the process of product introduction, ideas and needs are converted to the information from which technical systems and products can be made (Pahl and Beitz 1984 and 1996; Hales 1987). To be successful, a new product must offer customers better functionality, in comparison with other existing products. The product introduction is characterised as "the process of devising artefacts to attain goals i.e. product functionalities". While companies may have different names for the phases of their product introduction process, they usually map into five stages: idea validation, conceptual design, specification and design, prototype production and testing, and

manufacturing ramp-up (Rosenthal 1992). The PI process is typically controlled as a project using traditional project management tools (Program Evaluation and Review Technique - PERT, Critical Path Method - CPM), and the terms 'PI process' and 'PI project' are used synonymously in this thesis. In a large and uncertain project like the introduction of a complex product, e.g. aeroengine, the entire project task network from beginning to the end is not specified to the same level of detail all at once. The project is usually sketched at a high level only, and it is decomposed further as the occasion demands for a finite period in the future. The interdependencies among the decomposed units can be quite complex and difficult to control. As the product introduction process is a mapping between problem space and solution space, within each stage there is much iterative refinement, and compromises between competing alternatives must be made. Global iteration between stages can also occur as changes in the specifications, discovery of new technologies and changes in the availability of materials can cause a return to a previous stage to alter or replace the process (Anderson and Crawford 1989). In other words, the process can be considered a problem solving activity where a problem and its solutions co-evolve (Chakrabarti 1993) in a sequence of stages, with each one characterised by a problem solving cycle. To enable project time compression, it is essential to maintain effective control over the iterative process with efficient feedback loops. Speedy communication of information is fundamental to this task and is one of the prerequisites for successful control (Wu 1994).

1.3 PROCESS CONTROL

The way of controlling the product introduction process is by project management procedures (Parker 1997). Two critical factors for the success of the PI process are

the management of the product introduction activities, and the quality and functionality of its output (i.e. the product), which itself depends on the quality of the product introduction process. A general strategy for managing/solving a difficult problem is to reduce the overall complexity by splitting it down into smaller, more manageable sub-problems. Each sub-problem can then be tackled more or less independently, though the links between them must always be kept in mind. Finally the individual solutions must be combined to produce an overall solution to the problem (Wallace and Bligh 1996). In case of the PI process, the overall problem is to "define a product that meets the given functionalities" and the solution is "definition of a product with the specified functionalities". Once the idea is validated, the process involves the following activities: setting the objectives, planning, communicating the plans, monitoring and controlling the execution of plans, and reviewing the outcome. Management of the product introduction process for a large/complex product involves management at various levels of hierarchy and requires the co-operation of groups of people from different disciplines, often distributed in different geographical locations. To ensure that the project is achieving the objectives that are set, the process should be capable of receiving feedback at any stage. Thus, the managerial actions are based on the principles of feedback control. Feedback compares the output with a specified criterion in order to reduce the deviation between the actual state of a system and the desired state (Wu 1994). The output of the activities of the PI process would be the prototype of product units and information on the product units. The criteria for comparison would be the performance parameters of the product functionalities, as well as performance parameters such as cost and time of the activities of the project. Thus, for effective management of the PI process, the process should be capable of

receiving such information on product, product functionalities and project continuously.

Concurrent engineering - Concurrent Engineering (CE) has emerged as one of the most commonly referenced enablers for improved product introduction (Kusiak 1993; Sata 1993). The definition of concurrent engineering proposed in the work of Cleetus (1992) is arrayed using the print format to reflect the arrangement of thoughts:

> "CE is a *systematic approach*
> to integrated product development that emphasizes
> *response to customer expectations*
> and embodies
> *team values* of co-operating, trust and sharing
> in such a manner that
> *decision making* proceeds
> with large intervals of *parallel working* by all life-cycle perspectives in the process,
> *synchronised* by comparatively *brief exchanges*
> to product *consensus*."

A key term in a more precise definition of CE is to perform "interacting" activities at the same time. Thus parallel performance by itself is not a sufficient condition to characterise a process as concurrent. More managerial effort is needed in performing interacting activities rather than serial ones as they require considerable co-ordination effort to utilise resources effectively (CERC 1992). The effective deployment of concurrent engineering requires a smooth flow of information across the organisation boundaries involved in the PI project. It is therefore expedient to carry out concurrent engineering with the aid of computer technology. However, computer-assisted concurrent engineering can only be implemented when concepts have been developd to fulfil the following requirements - ① uniform data models for displaying the product characteristics at the various phases of development, ② a system of information management to organise the information arising in the various phases and amendments

to this information, and ③ communication capabilities to support the simultaneous exchange of information between various disciplines (Pfeifer et al. 1994). In order to manage and control the PI process, it would be necessary to integrate the above mentioned data models and communication capabilities with the process models that represent the process and activities that generates and make use of the information. Thus, a marriage of process and information in the form of process-based information representation would seem inevitable in light of the growing interests in these two areas. The infrastructure required for realising this identifies several tools and research issues. Many of these tools and issues deal with technology that is currently maturing, while others identify potential research initiatives.

Teamworking - The birth of multidisciplinary teams in PI is a result of implementing concurrent engineering practices and the need for employing the best resources for effective decision-making. The success of the implementation of concurrent engineering methods, tools and technologies in any industry will depend on the success of their teamwork (Kusiak 1993). Cross-functional teams work co-operatively to address downstream development issues as early as possible to create a high-quality, low-cost product more quickly. However, in a number of organisations, physically co-locating a team of relevant experts is very difficult or even impossible. For a variety of reasons, the relevant people, their tools and computer equipment are in geographically dispersed locations, and the cost involved in co-locating the people and the appropriate tools may be too high. One of the key factors for teamwork to be effective is 'good communication' i.e. the ability of team members to build on each others' ideas. The importance of information sharing and information flow in teamworking has been discussed by many researchers (Karinthi et al. 1992; Ranky 1994). Thus, the

foundation for the team-based co-operation is sharing of information, and the teams would require an information infrastructure that supports the sharing of information.

1.4 COMMUNICATION AND INFORMATION FLOW

Effective communication is one of the prerequisites for successful control of the PI process. The PI process is as effective as the decisions made within it, and as efficient as the speed with which the information required for each decision is made available. Thus, the whole control process depends on the flow of information among the activities. Information is received, processed and transmitted. The overall PI process may be considered as the conversion of information. After each new step, it may become necessary to upgrade or improve the results of the last, that is, to repeat it at a higher information level and to reiterate until the necessary improvement has been made. As a result, every conversion of information provides data not only for the next step, however small, but throws fresh light on the previous one (Pahl and Beitz 1984 and 1996).

The timing of access to information is particularly critical to a project's success because it can directly affect the cycle time to introduce a new product. Providing appropriate product design information to the project team as soon as it is available will allow them to get an early start in activities that are critical to a successful product release. The pursuit of reduced product development cycle time is likely to be sufficiently important to make communication acceleration an important information processing function (Rosenthal 1992). Existing process networks include a lot of information, but what is missing is the organisational influence, or how, where and

what information is generated and how, where, what and to whom information has to be sent across organisational boundaries. The information chain is a network of potential input and output through the PI process chain where the PI process chain is composed of a hierarchy of projects with activity networks. It would be necessary to analyse the relationships such as input and output between the activity and the information associated (used and/or generated by) with the activities. Thus, "information" is one of the crucial elements for managing the PI process, and the flow of information would need to be managed for the PI process to be effective.

1.5 INFORMATION

A great deal of knowledge is generated during the product introduction process. This evolving product knowledge would need to be captured and represented in a structured form for easy dissemination and sharing among the PI resources i.e. teams. In general, product models are used to store such information (Krause 1989; Carver and Bloom 1991). The information is about the various aspects of the product such as physical structure, functional structure, production methods and drawings; and, hence, they would be heterogeneous in nature. Information associated with the activities of the PI process would be viewed from multi-perspectives that would lead to various semantics of the information. The results of the comparison of the actual output information (achieved functionalities in terms of parameters of the product) with the desired output (target functionalities) would control the direction of the PI project. The direction of the flow of information would change depending on the status of the project and product information. The representation of such technical information of the product and the flow of information among the activities that are managed within a project management framework would necessitate integration of the management

information and the technical information. In other words, an integration of activity and output of the activities would be essential. In case of the PI process, the information is also of an evolving nature and possess different semantics at different stages of the process, thus adding complexity to its representation. Typically the modelling process involves determining not only what data needs to be represented from the various disciplines involved in the PI process, but also how this model can be represented in a way that would be dynamic, neutral, useful to all members of the PI team who need it and allow information flow.

1.6 OBJECTIVES

Process management includes project management, information flow management and information management. Crucial for the management of the product introduction process is the availability of a suitable and formally defined technique for the representation of the required information. The main objectives of this research have been to:

1. design and develop a representation of the information evolving out of the activities of the PI process of a complex product
2. design and develop a representation of the link between the managerial information and the technical information, which can support the management of the overall PI process, and
3. test the validity of the representation by developing a prototype to demonstrate the primary features of the PI process.

1.7 THESIS ORGANISATION

This thesis describes an integrated information model for the management of the product introduction process, how its ability to support the management of the PI process was tested by developing a prototype and the tools that have been developed to demonstrate the use of the integrated information model are described.

1.7.1 Review of existing knowledge areas

Chapters 2, 3 and 4 provide a basic structure for understanding PI project management, information management and the necessity for their integration.

Chapter 2 addresses and examines the approaches that support the management of the product introduction process and discusses the importance of the project and/or product information flow among the activities in a process based information model.

Chapter 3 reviews in detail the existing modelling methodologies which have been developed to facilitate the design and implementation of the information systems for complex environments, and relevant reference models.

The information about the product generated by the PI process would be represented in a product model. Chapter 4 reviews product modelling, product models from the literature and the relationships between process chain and product model. A comparison of product models is presented and it is argued that the existing product models do not address the issue of how to link the process that generates the product data with that data.

1.7.2 Analysis and design

Chapter 5 analyses the product introduction information in order to identify the basic elements, relationships and methods for integration. The results of the analysis are used to address the fundamental research questions on the dimensions that exist for modelling information and how to represent the evolving information. It also presents the demands on the database technology to develop an integrated information model and the requirements for information management to support concurrent engineering in product introduction.

Chapter 6 proposes an integrated information model and discusses the architecture of the model. The data structures that constitute the building blocks of the architecture are detailed, and the important characteristics of the architecture of the information model are highlighted.

1.7.3 Prototype and tools development

An overview of the design and development of a prototype that shows the primary features of the integrated information model is presented in Chapter 7.

Chapter 8 discusses how the global database representing the integrated information model can be developed, how the primary features of the integrated information model have been developed in the prototype, and the tools that have been developed to demonstrate the use of the integrated information model.

Finally, Chapter 9 summarises the research in the thesis, discusses the conclusions and the achievements, describes the limitations and applications of the proposed integrated information model, and presents recommendations for future research.

CHAPTER 2

2. STUDY OF PRODUCT INTRODUCTION PROCESS

2.1 INTRODUCTION

Product introduction (PI) is a vital process for manufacturing firms' growth and prosperity (Zirger and Maidique 1990). It can be seen as a decision-making process whose input is a perceived need or set of customer requirements and whose output is a detailed plan for realising a process which meets that need or set of requirements. Through the process of product introduction, ideas and needs are converted to the information from which technical systems and products can be made and sold. Two critical factors for its success are the management of the product introduction activities, and the quality and functionality of its output (i.e. the product), which itself depends on the quality of the PI process.

In this chapter, we examine the product introduction process, and approaches such as project management, concurrent engineering, teamworking and information management systems that support the management of the product introduction process. We note the importance of the project and/or product information flow among the teams carrying out the activities, and consider the relationships between activities and the information associated with the activities, the roles of the information with respect to the activities, dependencies among the information, and dependencies among the activities based on the associated information that would be needed for a process based information model.

2.2 PRODUCT INTRODUCTION PROCESS

The product introduction is a process. There are various definitions for process in the literature. A process is a sequence of interrelated activities (Manganelli and Klein 1995). A process is an occurrence of some duration which is started by an event and completed by an event (Scheer 1993). A process is any activity which takes an input and transforms it into an output (Munro-Faure 1994). A process is the transformation of a set of inputs, which can include actions, methods and operations, into outputs that satisfy customer needs and expectations, in the form of products, information, services or – generally results (Oakland 1995). In this research, a process is defined as a sequence of interrelated activities that has a set of goals, constraints, inputs and outputs. PI process aims to define a new product that satisfies specific product functionalities within a specific time and budget, and generates the definition of the new product. Process/Project management techniques are applied to the PI process, and the whole process is normally subdivided into manageable subsets. Checks are applied at the end of each sub-process for conformance. Each of these sub-processes is called a stage and the conformance audit points are called the gates in the stage-gate process (Figure 2.1).

Each stage is designed to gather information needed to progress the project to the next gate or decision point. Each stage is multifunctional, and consists of a set of parallel activities undertaken by people from different functional areas. These activities are designed to gather information and reduce uncertainties in the project. According to Cleetus and Reddy (1992), each stage of the product development process comprises steps shown schematically in Figure 2.2.

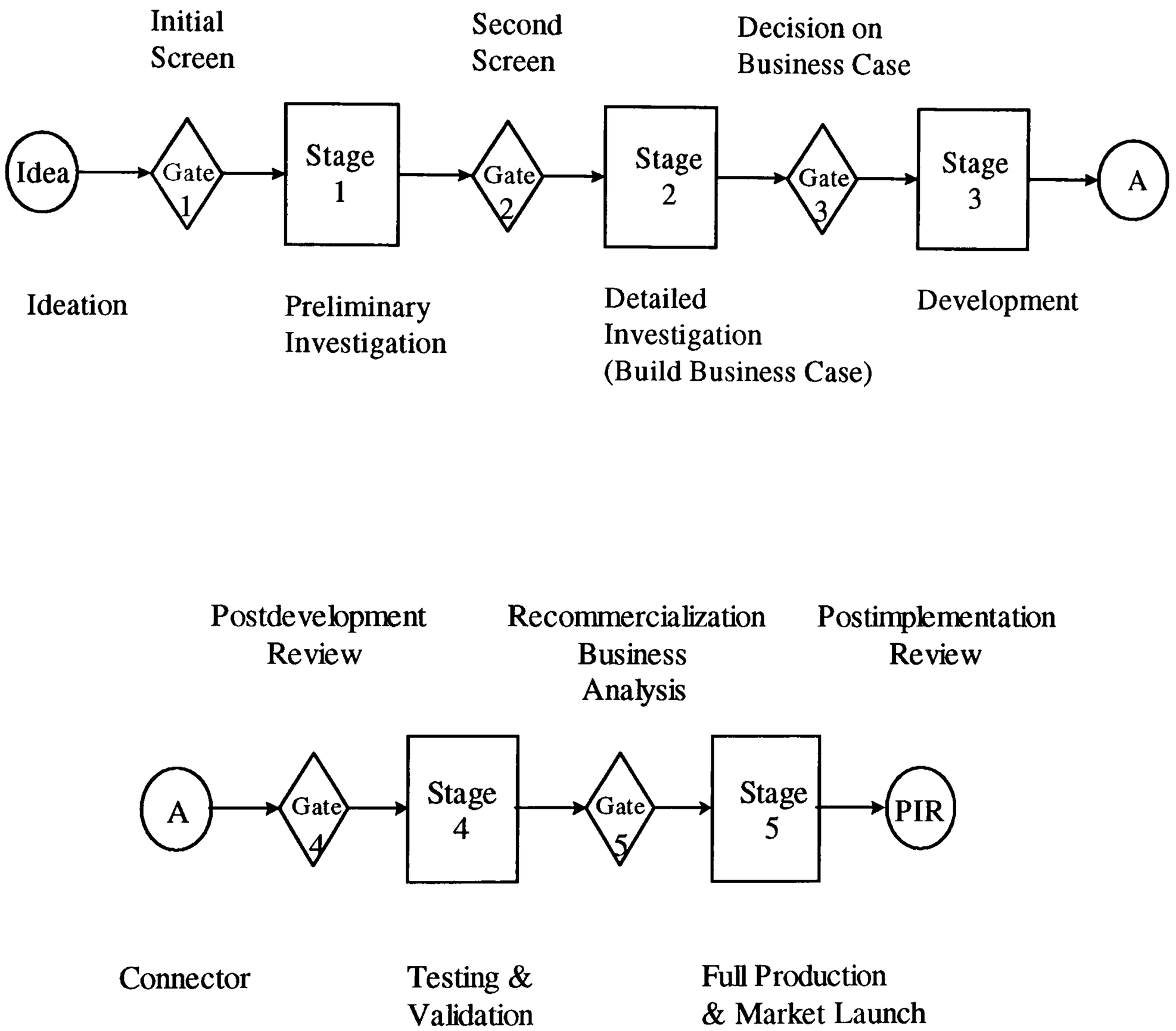


Figure 2.1 A generic stage-gate new product process (Cooper 1993)

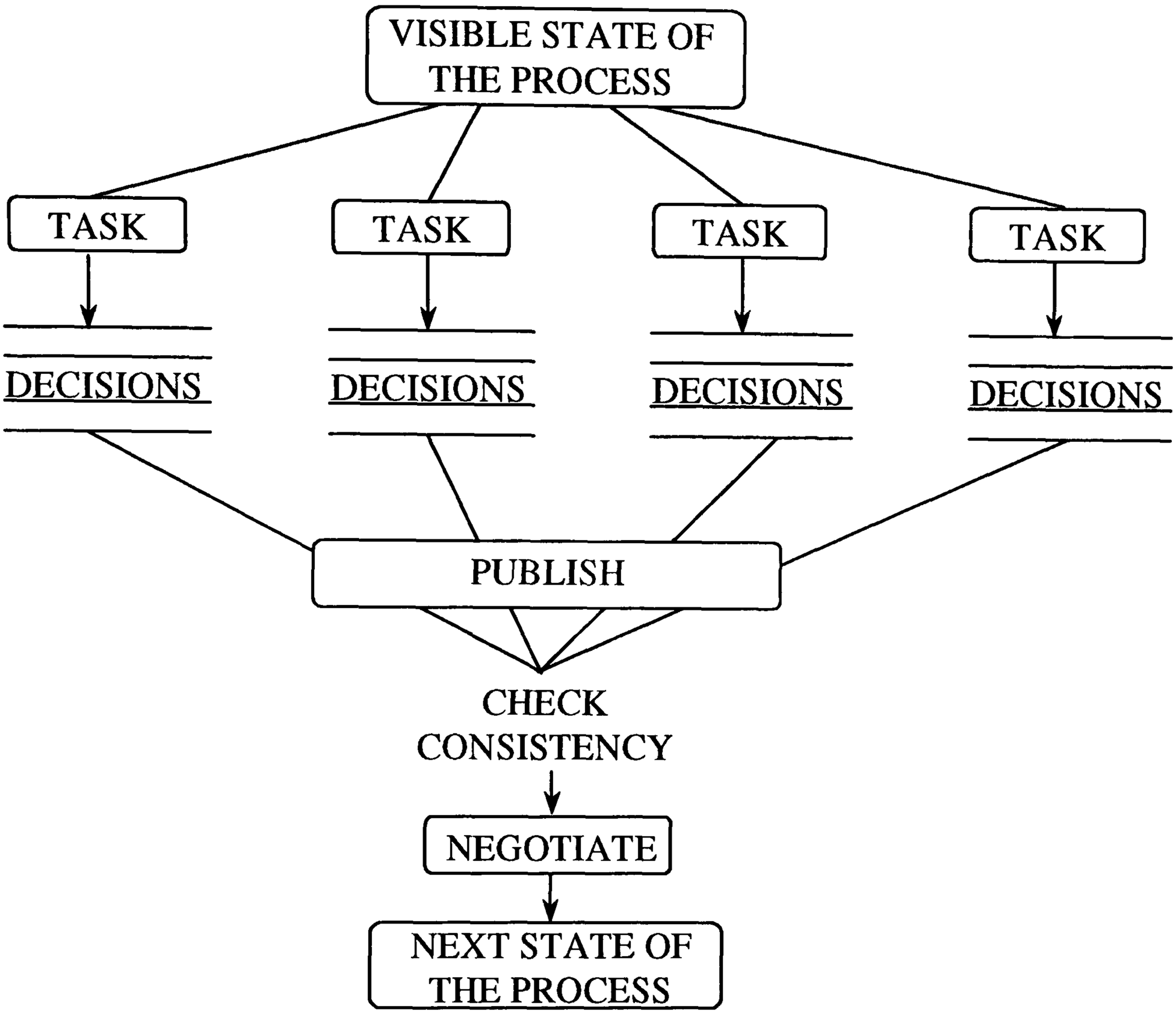


Figure 2.2 Steps in a stage (Cleetus and Reddy 1992)

Figure 2.3 depicts a general model of the product introduction process and it consists of the following stages: need assessment, analysis, decomposition, synthesis, integration and evaluation. The first stage is concerned with the assessment of the desired need and requirements which are usually fuzzy in nature. Analysis involves the specification, identification of the problem and producing an explicit statement of goals. Decomposition is concerned with breaking the problem into parts and defining the boundaries of a space in which a fruitful search for the solution can take place. Synthesis is concerned with discovering the consequences of putting a new arrangement into practice. Evaluation is judging the validity of the solutions relative

to the goals and selecting one among the alternatives. The inner cycle implies that the outcome out of the integration phase is revised and improved by re-examining the analysis. The outer cycle demonstrates that as the solution is evaluated it might revise the perceived needs.

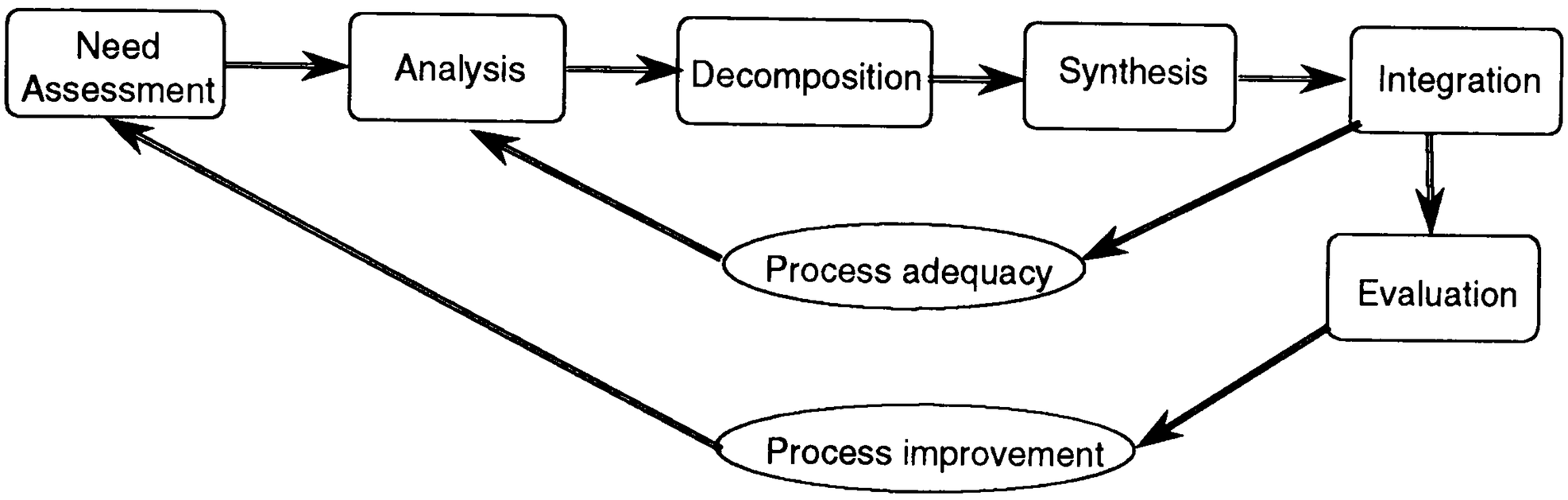


Figure 2.3 A general model of PI process

According to Cooper (1993), the product introduction process is both a conceptual and operational model for moving a new product from idea to launch. It is conceptually simple but the intricacies and design of stages and gates are quite complex. According to Shenhar et al. (1995), the product introduction process involves a technical/engineering process and a managerial process along the various phases of the project's life cycle. The first process consists of technical activities (selection of the product's concept and configuration, the engineering design, building and testing, redesign) that lead to the assembly of external and/or internal pieces of technological knowledge to create and shape the features of the project's outcome. The second process involves the managerial activities (planning, scheduling, budgeting, contracting, organising, staffing, controlling) that are performed to allocate, use, and monitor the project's resources; co-ordinate the parties involved;

manage the communication and information flow; and support the technical process via decision making and data management.

An effective process is defined as one that results in a product satisfying the actual need (Blessing 1993). For effective control of a process, a management system has to meet the following requirements (Drongelen et al. 1996): ① it has to specify the *goals* the process will aim at and use these goals as guidelines for management, ② it has to have a *model of the process*, describing how internal and external disturbances and management measures can influence the behaviour of the process, ③ it has to have *information* about the actual situation of the process and the relevant variables in the environment, and ④ it must have a *sufficient variety of corrective measures*.

2.2.1 Goals of the product introduction process

New products are more successful if they are designed to satisfy a perceived need (Zirger and Maidique 1990). Customer satisfaction is an essential goal and is the key to the success of a new product (Amalnik 1994). First and foremost the customer wants a product that performs the function that is expected, and serious customer dissatisfaction will result if this fails. Therefore, an organisation stands or falls on its ability to deliver products that meet functional requirements (Fox 1994). A product does not depend on one function alone but will have a series of key functions which combine together to define its value. At the beginning of the product introduction project, the proposed product is described in terms of a set of functions which it must fulfil (Wallace and Bligh 1996).

Depending on the complexity of the problem, the resulting overall function will in turn be more or less complex. A complex or overall function can be broken down into sub-functions of lower complexity. The combination of individual sub-functions results in a function structure representing the overall function (Figure 2.4). The function structure of the product is the primary determinant of its complexity; the solution structures which follow determine its modularity, and the decisions made on them set the length of the introduction cycle. It has been estimated that 10-20% of the total time spent in devising function and solution structures determines 80-90% of the product cost (Hundal 1993). Every element of the product has a function (Blessing 1994). A small percentage of the parameters called critical parameters will be the key to the product's functional performance. By managing the critical parameters in a product the designer can have full control of the functional characteristics of the design (Fox 1994).

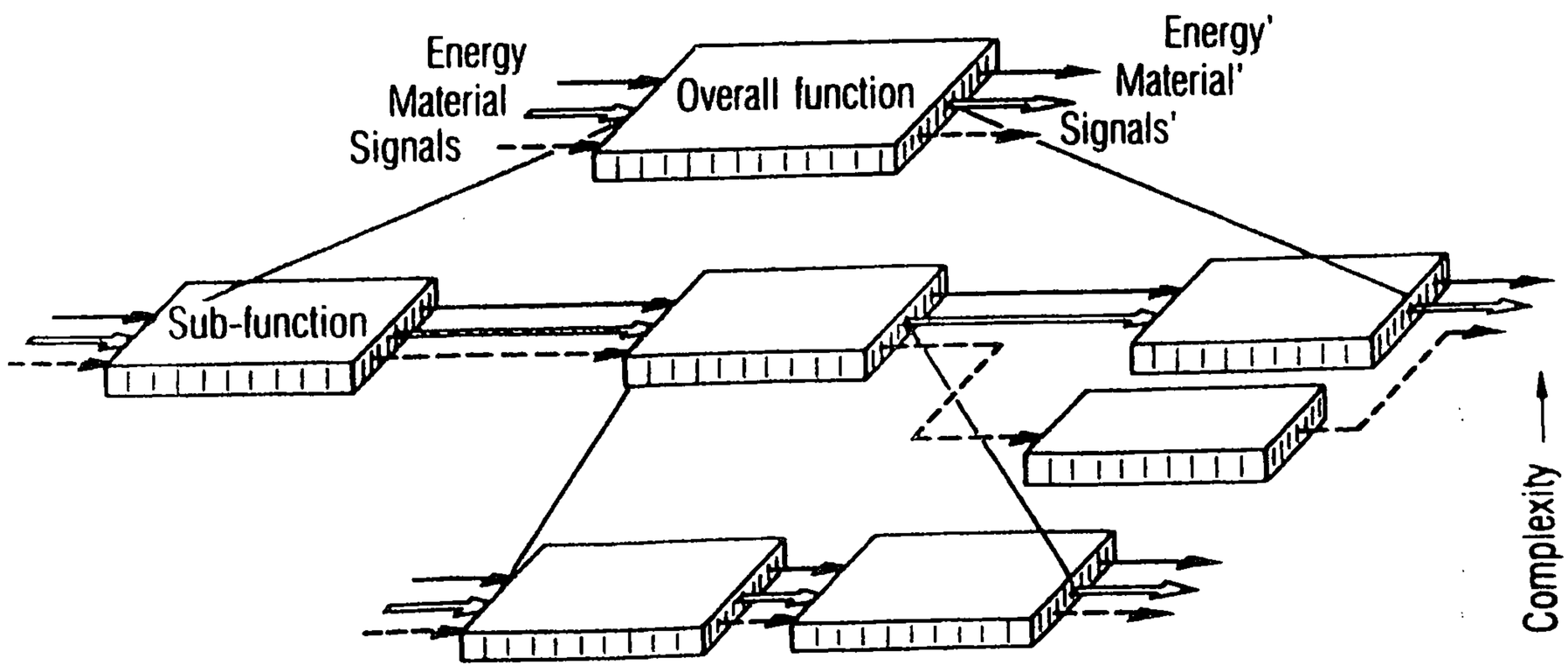


Figure 2.4 Function structure (Pahl and Beitz 1984 and 1996)

2.2.2 Targets of the product introduction process

Progress of the product introduction project from one stage to the next is controlled by a formal review to check that intermediate targets of performance and cost are being met (Lewis and Haiying 1998). In general, four strategic targets will provide the necessary common vision for a product introduction project: project time, project cost, product quality and unit cost. Project time is the elapsed time between making a decision to develop a product and having a new product, meeting the necessary requirements and available for purchase. Much of this time is taken up by the product definition process. Project cost is the cost of the activities of the product introduction process. The reduction of the product introduction project cost comes down to a reduction of the cost of its activities. Thus cost is considered as a criterion which allows the measure of the economic efficiency of an activity. According to the research findings of BS6046: Part 1 (1984), important controllable factors related to project costs are: timeliness of decisions, process effectiveness and organisational structure. An effective process is defined as one that results in a product satisfying the actual need (Blessing 1993); in other words, project effectiveness is about meeting product quality or functionality (Hauptman and Hirji 1996). An efficient process is defined as one that is both effective and in which the applied resources do not exceed the planned resources (Blessing 1993) i.e. meeting project budget and schedule (Hauptman and Hirji 1996). Product quality is usually defined as its fitness for purpose (validity of the product specification) in terms of its functionalities. Product cost includes cost of materials and manufacturing processes. A large proportion of the cost associated with a product is fixed comparatively early in the product introduction

process, as soon as the principle by which the specification is to be met has been selected.

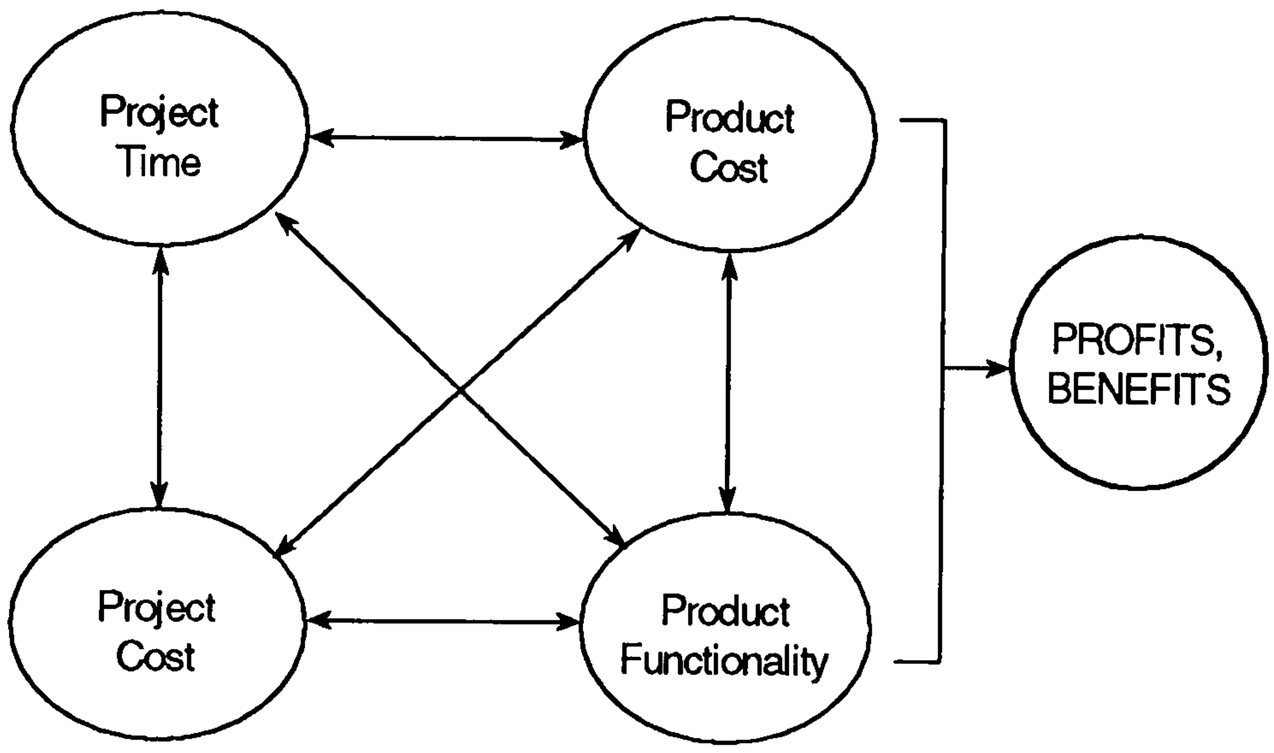


Figure 2.5 The trade-off parameters in the product introduction process

At the start of the project there is a compromise to be made between the above four targets (Brooks 1995). The first two of these targets relate to the product introduction process, while the last two relate to its outcome (product). All targets (i.e. targets related to the process and its outcome) should be considered simultaneously, and the resulting target set should be an integrated reflection (Figure 2.5) of the relative priorities among targets (Rosenthal 1992).

2.2.3 Outcomes of the product introduction process

Product is the result of activities or processes (BS7000:Part 10 1995). Figure 2.6 shows the many outcomes of the process. Apart from the physical product, there are other outcomes such as customer perception and cost. More importantly, there is also an information product. A great deal of information is generated to specify the product; this product information evolves as the process that generates it undergoes revision.

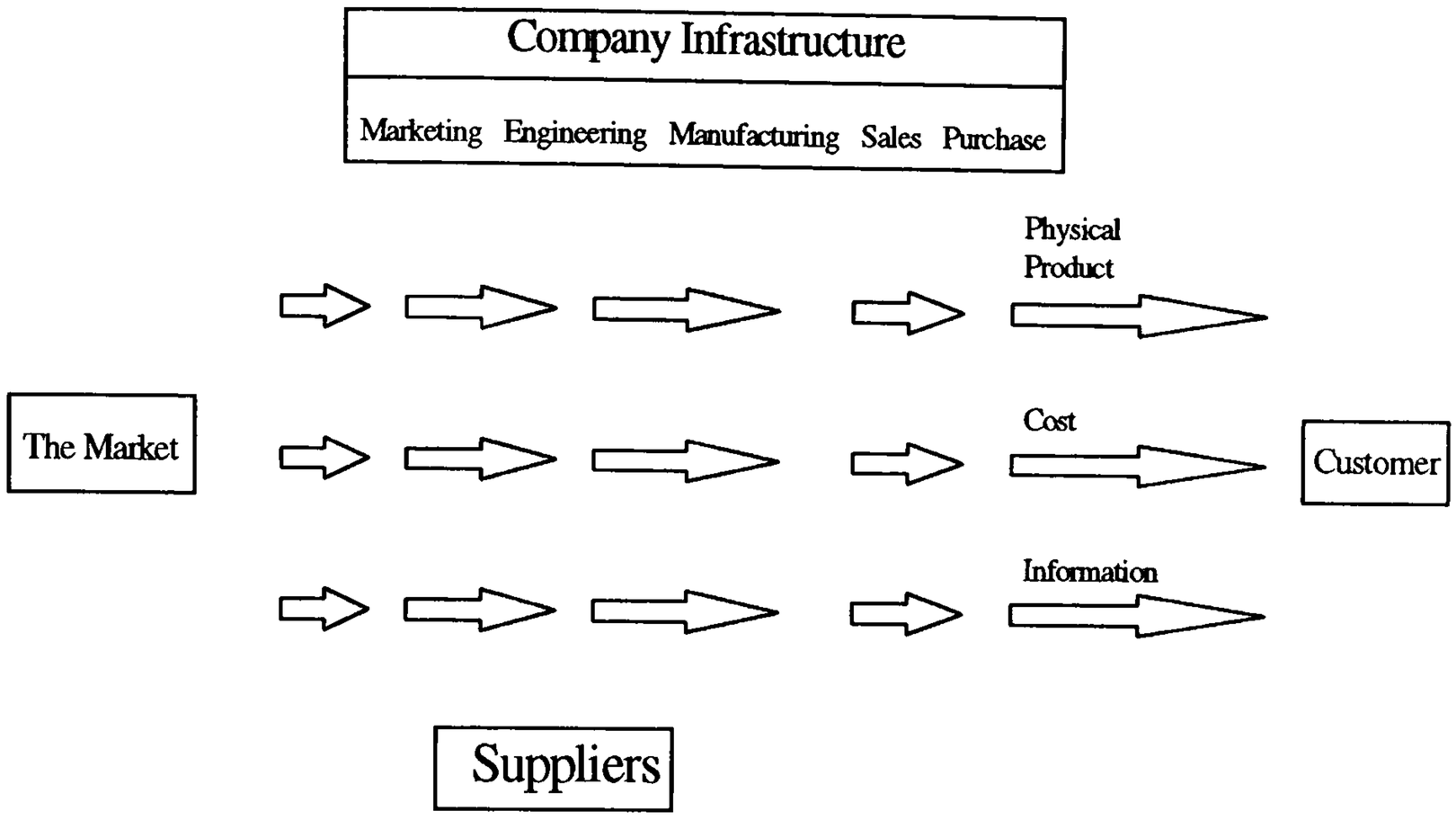


Figure 2.6 The detailed value chain model (Stark 1992)

Product model (Figure 2.7) contains the results of the PI process. Part of the product model is the relationship model (Blessing 1994). Relationships such as hierarchical, topological and functional exist between product elements. Not only product elements, but also their relationships are developed during product development. The relationships play an important role in the development of a product because they determine the consequences of introducing, modifying and detailing product elements, and they can trigger other product elements or relationships. The importance of relationships or interactions between product elements is emphasized by many authors (Andreasen 1990; Ullman 1993; Blessing 1994). A product should be defined as a collection of elements connected by relationships, and it is argued by Blessing (1994) that a product functions because of its relationships.

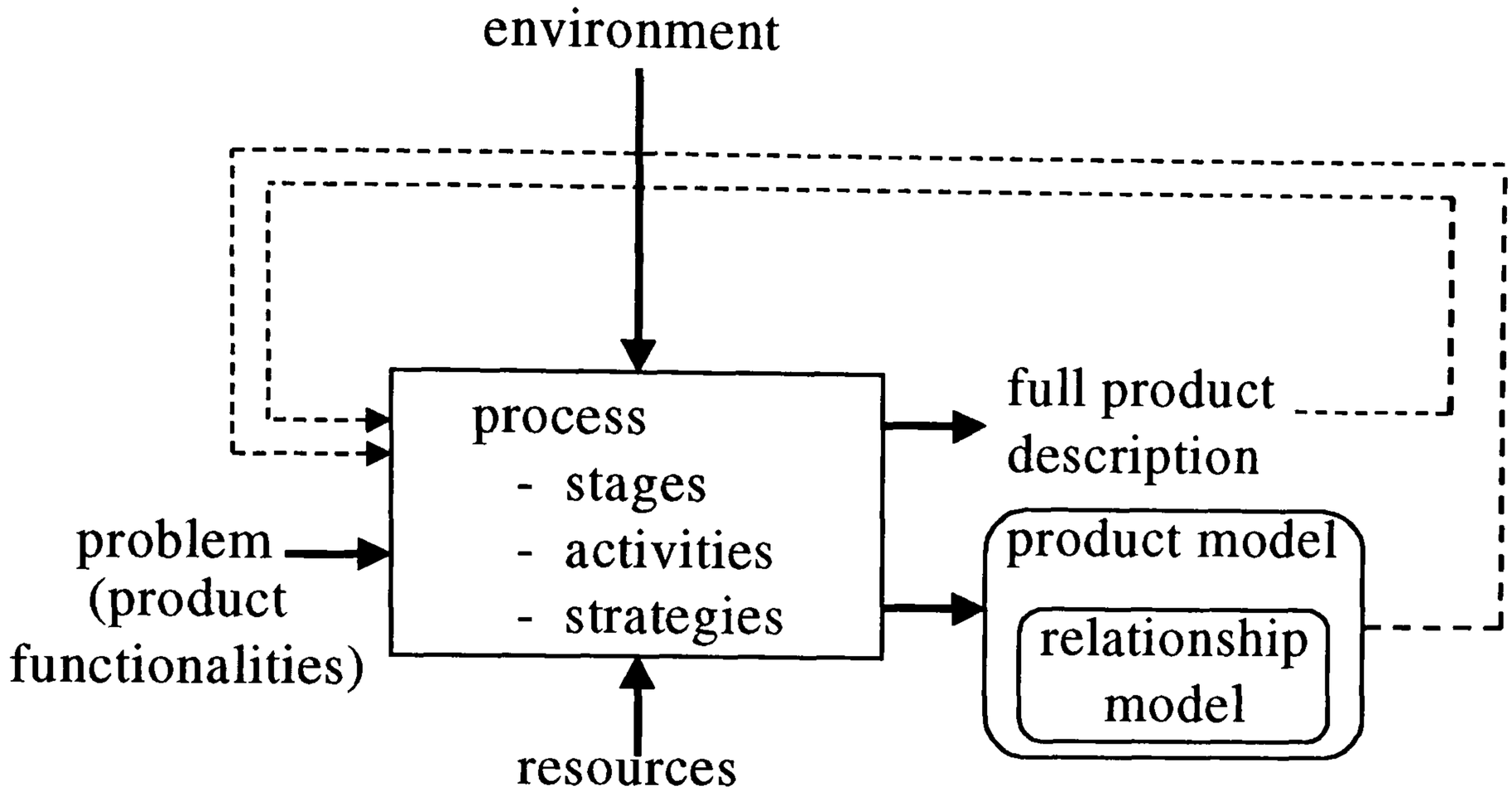


Figure 2.7 Product introduction process model

The activities that result in a product are managed and controlled using project management procedures. Thus, the project itself is the process by which the product is introduced. The quality of the product strongly depends on the quality of the process (Blessing 1993; Munro-Faure 1994). In other words, management of the process has an impact or affects the quality of the outcome of the process.

2.3 APPROACHES TO SUPPORT THE MANAGEMENT OF PI PROCESS

Success in product development is a critical management issue for technology driven industries (Zirger and Maidique 1990). There are many approaches to support the management of product introduction process. These include project management systems, concurrent engineering technique, teamwork and information systems. These approaches are discussed in the following sections.

2.3.1 Project management systems

Based on an investigation - "What makes a real winner in winning at new products", Cooper (1993) concluded that the factors that describe the way the project is organised and undertaken - actions, process, and players - dominate the list of reasons for success. The product introduction process can be seen as a series of projects, each of which is made up of a series of decisions that combine to transform an initial idea into a marketable physical activity (Cleetus and Reddy 1992; Rosenthal 1992). A major problem which often arises is that a person involved in the product introduction process may make changes in the product/process without considering their overall effect, and/or might delay the task without knowing the impact the delay has on the overall program schedule. Many authors (De Maio 1994; Hayes et al. 1988; Parker 1997) have pointed out that the PI process has features of complexity and inter-functionality that make a multi-project management approach very suitable for controlling it. But, it is not possible to represent the iterative procedures and feedback loops that are essential for the effective management of the PI process using existing project management tools.

The co-ordination of various tasks - planning, scheduling, budgeting, contracting, organising, staffing and controlling activities, selecting product concept and configuration, design, building and testing activities, as well as information gathering and sharing, decision making, negotiation and many other tasks may be approached through project activity management (Badiru 1988 and 1996). Any product introduction project should begin with a plan tied to specific objectives for that project. According to Rosenau and Moran (1993), the glue that holds the new product

development project together is the business plan. Project planning is usually described as a process of activities that start by decomposing the project's work into a work breakdown structure (WBS), which is a tree-like chart that separates the work into the product's subunits and additional support activities. Each activity is then assigned to the organisational unit or team responsible for carrying it out, it is budgeted, and its projected length is estimated. This process results in a project schedule, intermediate milestones, and a project budget that are set in advance as constraints for project management. Three parts of the project management system that can be developed from the project breakdown structure are shown in Figure 2.8. Specific inputs and outputs, a duration estimate, a cost estimate, resources estimates, and precedence relationships are usually associated with each task in the WBS (Pietras and Coury 1994). The chain that is connecting the hierarchical subsystems can be thought of as passing requirements down and directing accumulating definition upwards (Ruffles 1995).

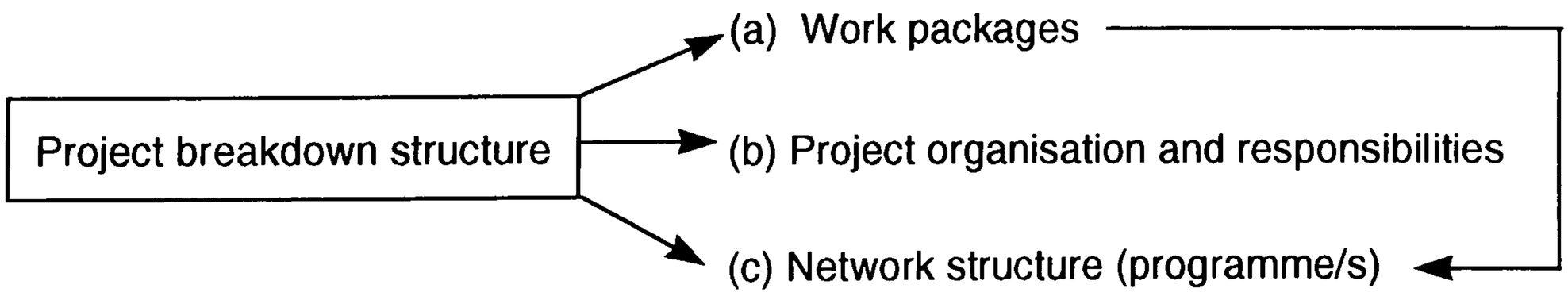


Figure 2.8 Project breakdown (BS 6046:Part 1 1984)

Scasso and Larens (1991) present the project breakdown structure (PBS), which includes work breakdown structure and organisation breakdown structure. Two major subsystems identified are the project units that are the elements that fulfil a specific function in the project, and the agents who act on or are responsible for the units

bringing about the transformation that will lead to the achievement of the project objectives. PBS model is represented using ten levels: project identification, areas, sectors, project units, subunits, quantities of work or components, speciality, subspeciality, stages of project and executor. Badiru (1996) proposes a triple C model as an effective tool for project planning and control, based on the integrated functions of communication, co-operation and co-ordination.

Design of a product is an important process in product introduction as approximately 70% of the total product cost are determined by the design department; even 70% to 80% of the product success will depend on this area (Lohse 1993). 80% of new product development project expenditure is in the design and development of a product (Hogarth and Tabeshfar 1993). While discussing the shortcomings of the design frameworks, Erens et al. (1993) bring out the difficulties in linking project management and design. These include: ① structural (meta-) data is typically more dynamic and is considerably intertwined with repetitive (instance-) data, ② amount of structural information is large compared to the size of the information content and ③ it is difficult to accommodate different viewpoints in a single structure. The design management response to the challenge of reducing product development lead time has typically been to encourage engineers to develop the product and its associated manufacturing process concurrently. This policy has two beneficial effects. First, it emphasises the need for design engineers to be aware of production issues, and this is the focus of the popular "design for manufacturing" approach. Second, the designers are sharing or transferring information to their counterparts in manufacturing engineering much sooner than they had done previously (Eppinger et al. 1989). These

trends bring up new issues in design project management. To enable project time compression, it is essential to maintain effective control over the iterative process with efficient feedback loops.

2.3.1.1 Iterative procedures and feedback controls

The product introduction process is a mapping between problem space and solution space. One cannot gather information meaningfully unless one has understood the problem, but one cannot understand the problem without information about it (Blessing 1994). Thus, iteration increases the level of information and the teams have to iterate until the information level is high enough to find an optimal solution. Within each stage of the PI process there is much iterative refinement (BS7000:Part 2 1997), and compromises between competing alternatives must be made. Global iteration between stages can also occur, as changes in the specifications, discovery of new technologies and changes in the availability of materials can cause a return to a previous stage to alter or replace the process (Anderson and Crawford 1989; Chakrabarti 1993). To ensure that the design is optimised, the process should be capable of receiving feedback at any stage (BS7000:Part 2 1997). Thus, the managerial actions are based on the principles of feedback control (Wu 1994). Feedback is the sub-system function that compares the output with a given criterion in order to reduce the deviation between the actual state of a system and the desired state, when the system is subjected to unpredictable disturbances from its environment. System feedback takes place whenever information about any of a system's outputs is used to correct its operation (Wu 1994). The essential components of a feedback control operation are shown in Figure 2.9.

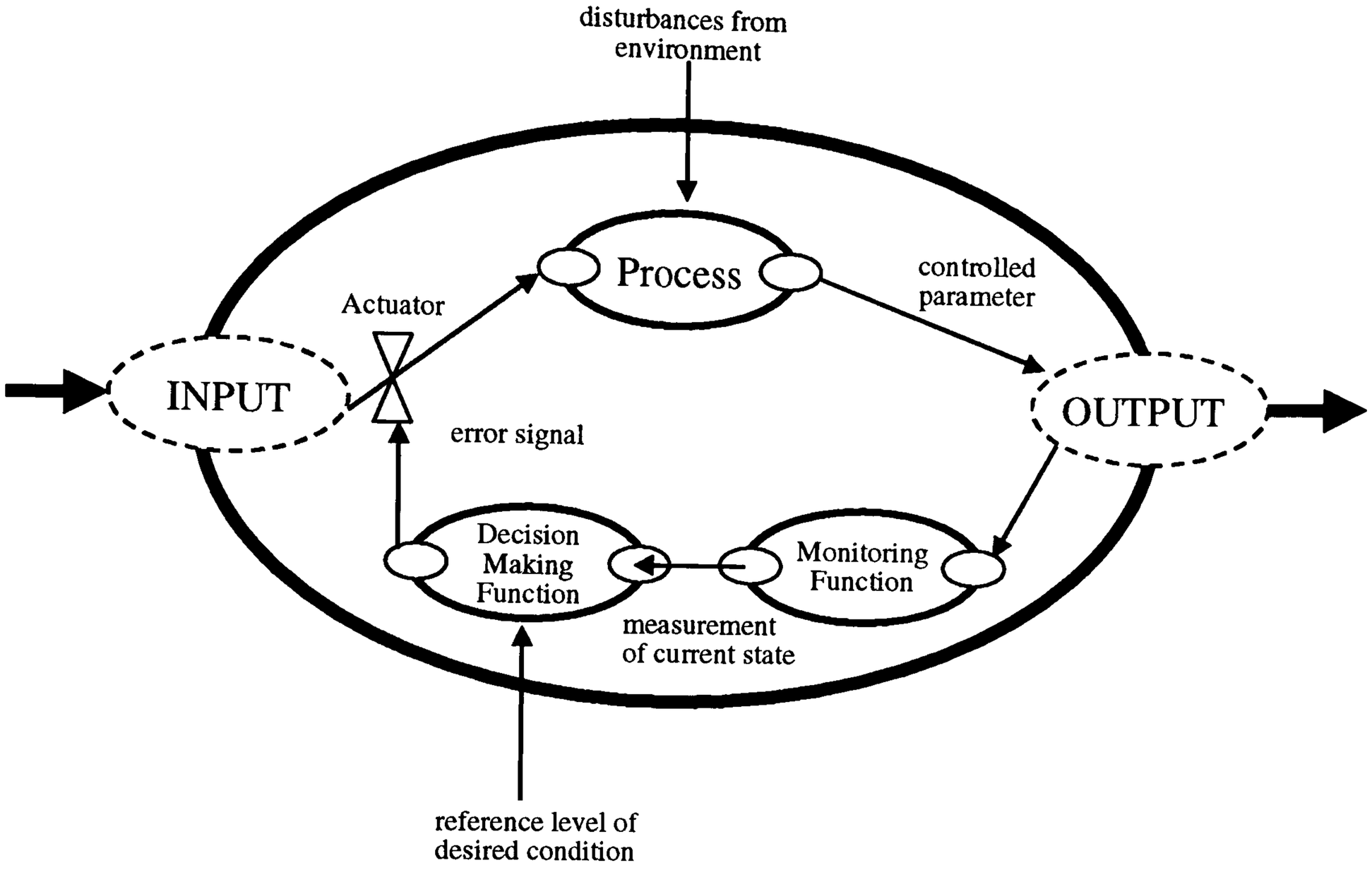


Figure 2.9 Essential components of feedback control (Wu 1994)

The information feedback system is very complex in its structure. Effective communication is one of the prerequisites for successful control. The whole control process depends on the flow of information among the activities. Various pieces of information are needed to enable decisions to be made. Output of the decision-making function will again be information in the form of instructions or constraints, and these must be channelled to the intended destination to initiate control action (Wu 1994). Thus, speedy communication of information is fundamental to the management of the task and is one of the prerequisites for successful control.

2.3.1.2 Limitations of Project Management Tools

Many authors (BS6046:Part 1 1984; BS 6046:Part 2 1992; BS 6046:Part 3 1992; Evans 1993; Kerzner 1995) have presented methods that are useful to planning,

scheduling and controlling complex projects using project management techniques. Even though project management is increasingly applied in a variety of industries, many managers find it difficult to capture the complexities involved in managing product introduction projects (Shenhar and Laufer 1995). This is because executing the process involves linking two different, though not disjoint, processes along the various stages of the project's life cycle - the managerial process and the technical process, and it is the technical information that controls the direction of the project.

The most popular project planning tools (PERT - Program Evaluation and Review Technique, CPM - Critical Path Method, PDM - Precedence Diagramming Method) use network diagrams to represent the precedence relationships among activities. Existing project management packages are all oriented towards repetitive and completely foreseen sequences of tasks from the beginning to end, which is not valid for product introduction, where allowances must be made for exploration, opportunistic contribution and joint planning of work. The project management tools have the following limitations when applied to the product introduction process. The first three limitations are from Eppinger et al. (1989).

1. The tools require that there be only one-way progression along paths, with no feedback or iteration, and no feed-forward of information part-way through a task. The emphasis is placed on the interactions between the tasks, not on the details within the tasks. The tendency is therefore to define tasks in the large, ignoring a multitude of engineering interactions required within each one. Furthermore these techniques are not aimed at providing communication.

2. Project management tools are generally applied to the product introduction process on the basis of "start task/complete task" representation, ignoring the technical information related to the tasks, the information the tasks need or produce, and the overall information flow network that underlies the whole effort. When a representation omits significant effects, then important system behaviour remains unexplained.

3. Project management software tools can typically analyse project sequence diagrams only if they contain no coupling (loops). The representation requires the coupled tasks to be bundled into larger design tasks. If the project planner chooses to consider the tasks separately, then the essential information coupling must be neglected.

4. Project management tools consider only resource management, not the goals (product functionalities) of the project.

5. While managing the PI process, it is necessary to monitor progress in meeting the defined specifications (product functionalities), and to make a compromise between project cost and the product functionalities. This necessitates the integration of workflow (process) and the product information. As the information such as product functionality can not be represented using the existing project management software tools, the tools cannot be used for managing the PI process effectively.

2.3.2 Concurrent engineering

In order to introduce high-quality, low-cost products quickly, now a days, concurrent engineering technique is practised by industry. Concurrent Engineering (CE) has been loosely used to mean a host of different things. Both 'concurrent' and 'simultaneous' have the meaning 'happen at the same time', but the word 'concurrent' has the additional meaning 'agreeing; co-operating' (Lindberg 1993). Concurrent Engineering, with its emphasis on product teams made up of individuals from different departments and even different companies, and parallel working on processes that were previously carried out in series aims to overcome the disadvantages of the traditional method. Its purpose is to deliver the product in the shortest time possible and at the highest quality level (Budill 1989), and it necessitates the focusing of the total effort towards achieving the common goal. Concurrency is present in three respects. At the highest level, different domains work on different perspectives of the product introduction process in parallel, if at all possible. Furthermore, several parts or components going into the final product may be worked on simultaneously; this is the second aspect of concurrency. And finally, within one domain, a group of designers guided by the group leader may be working on several different analyses of alternatives (Cleetus 1989). According to Carver and Bloom (1991), concurrent engineering involves the integration of people, systems and information into a responsive, efficient system. According to Brooks (1995), it involves the integration of process, people and systems (Figure 2.10).

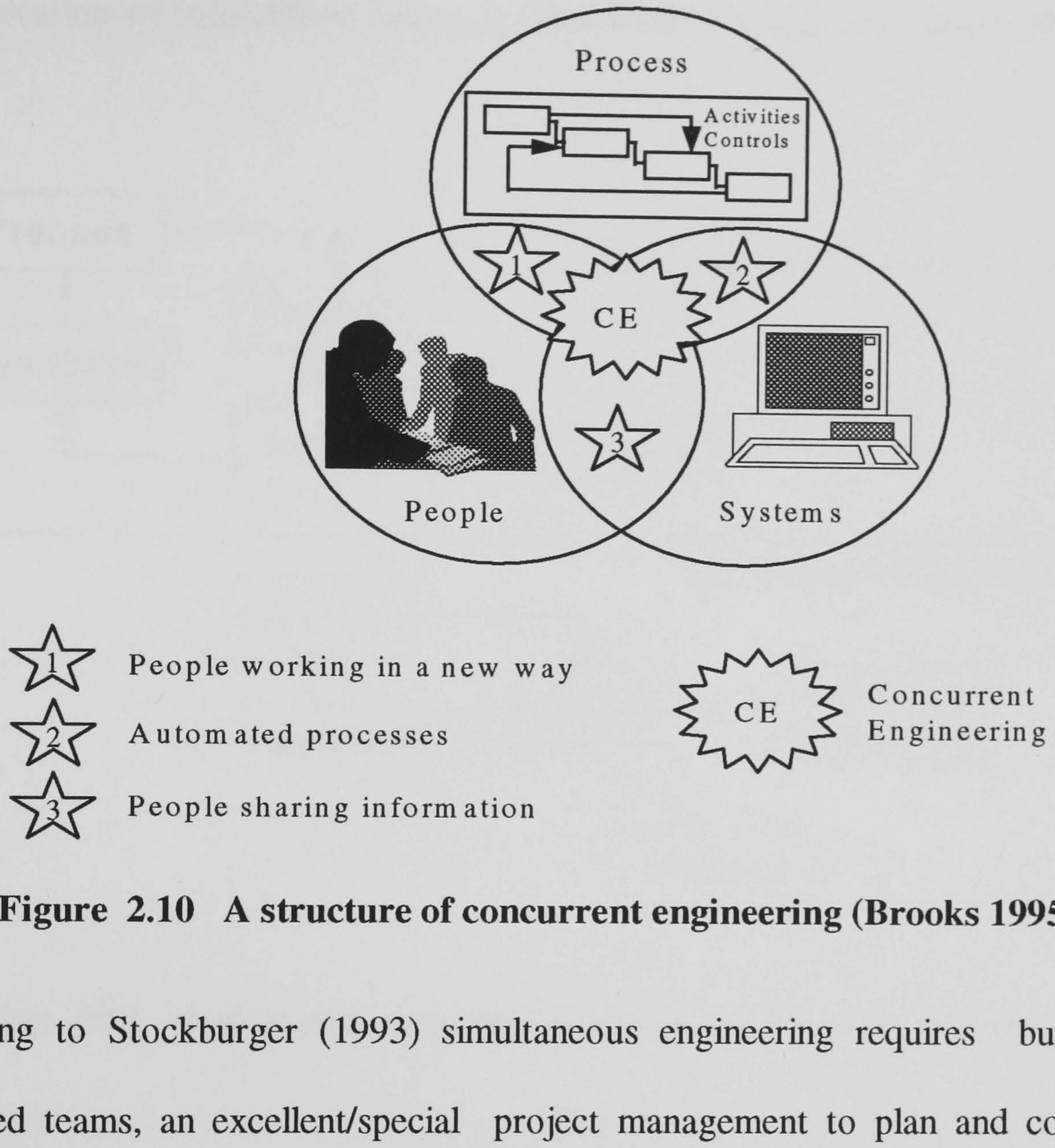


Figure 2.10 A structure of concurrent engineering (Brooks 1995)

According to Stockburger (1993) simultaneous engineering requires building up integrated teams, an excellent/special project management to plan and control the work, and simultaneous performing of work by using more analytical work instead of time consuming tests. Thus, in CE, the intention is to verify all important effects of the decisions during the initial phase (Figure 2.11). The resulting benefits are related to the possibility of early recognition of mistakes, their correction without loss of resources (Krause and Ochs 1992). For effective concurrent engineering, the information has to be shared among the team members. Marlow and Schulz (1994) identify engineering libraries that supply on-line, integrated, current and comprehensive information on parts, methodologies, and manufacturing processes are key enablers to a concurrent design process. It is only feasible in a well-controlled engineering environment, with a high availability of engineering information and rapid

communication of information between individuals (Carver and Bloom 1991; Stark 1992).

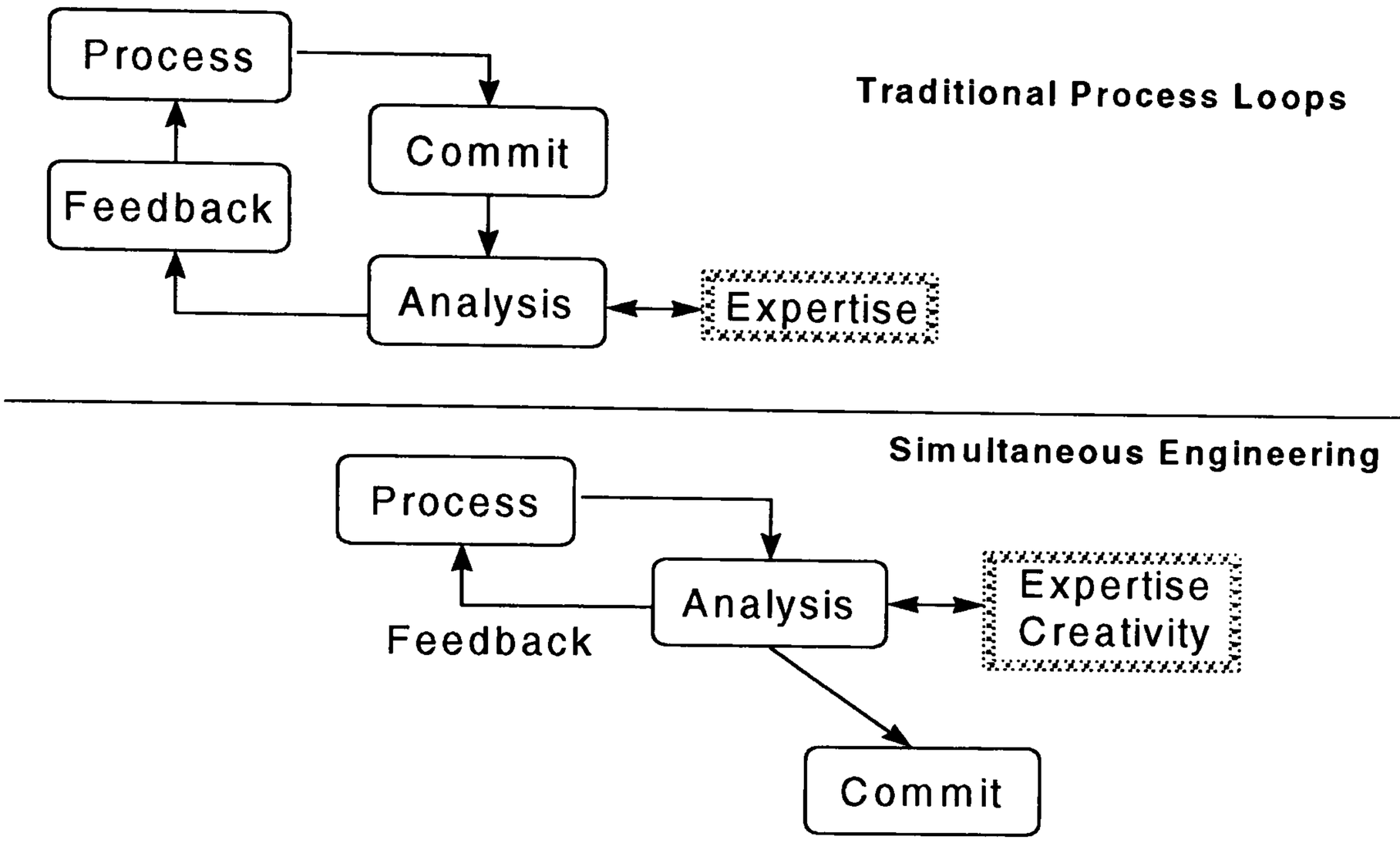


Figure 2.11 Traditional versus simultaneous engineering process loops

Thus, CE philosophy requires integration, cooperation and communication across the upstream product development and downstream functional groups. Such integration and cooperation present numerous difficulties, since organisations have evolved into specialised disciplines often geographically distributed (Amalnik 1994). Integration and proper co-ordination and communication for exchange of data between various functions in a diversity of formats without appropriate mechanisms and structured tools can lead to various difficulties and bottlenecks for concurrent engineering in product introduction. The necessity of the development of an integrated product data-base structure incorporating supporting features for accommodating the use of CE is emphasised by researchers of product introduction (Menon and Syan 1992). While discussing the goals and methods of CE, Rzevski (1993) identifies "improving access

to information" is the second sub-goal of "a reduction of lead times". Thus, CE is a process that integrates through the sharing of information. It could be implemented by assembling a team of (human) experts, each of whom is a specialist responsible for one or more stages of the product introduction. The human team approach works, but is limited by the amount and complexity of the information (Carver and Bloom 1991).

2.3.3 Team working

The birth of multidisciplinary teams is a result of implementing concurrent engineering practices and the need for employing the best resources for effective decision-making (McGrath et al. 1992; Ranky 1994). In order to reduce the project time, teamworking is essential. For large projects of long duration (introducing an aeroengine takes 4 to 5 years, software MS-Office 97 took 3 years, Ferguson tractor takes 4 years, automobiles such as cars take 3 to 5 years (Roy and Allchurch 1997)) require the formation of special teams (Lock 1984; Vonderembse and White 1991; Rzevski 1993). Co-ordination of the work of these experts then becomes an important ingredient of successful product introduction. If the company is large, there may be considerable geographic dispersion among the groups participating in the product introduction process. This is even more of a problem for a global company with engineering and manufacturing facilities in different countries. In this situation, some sharing or splitting of product introduction responsibilities across thousands of miles will frequently be necessary. In such cases, the concept of a virtual team, an electronically-networked team of product developers, serves as the foundation for developing technology in support of CE (Crabtree et al. 1993).

A product Introduction team normally consists of experts in market research, business strategy, mechanical, electronic and software engineering, process planning, manufacturing system design, product distribution and servicing, reliability and safety engineering, industrial design and ergonomics. It is general practice to keep a core of these experts throughout a PI project (Rzevski 1993). Thus the product introduction is carried out by cross functional teams. Vonderembse and White (1991) describe the roles of participants during the various phases of the product development cycle. A significant amount of overlap between the phases of new product development and a significant amount of information exchange between participants are not only necessary, but also desirable to achieve a high quality product design that meets customer expectations (Vonderembse and White 1991; Stockburger 1993).

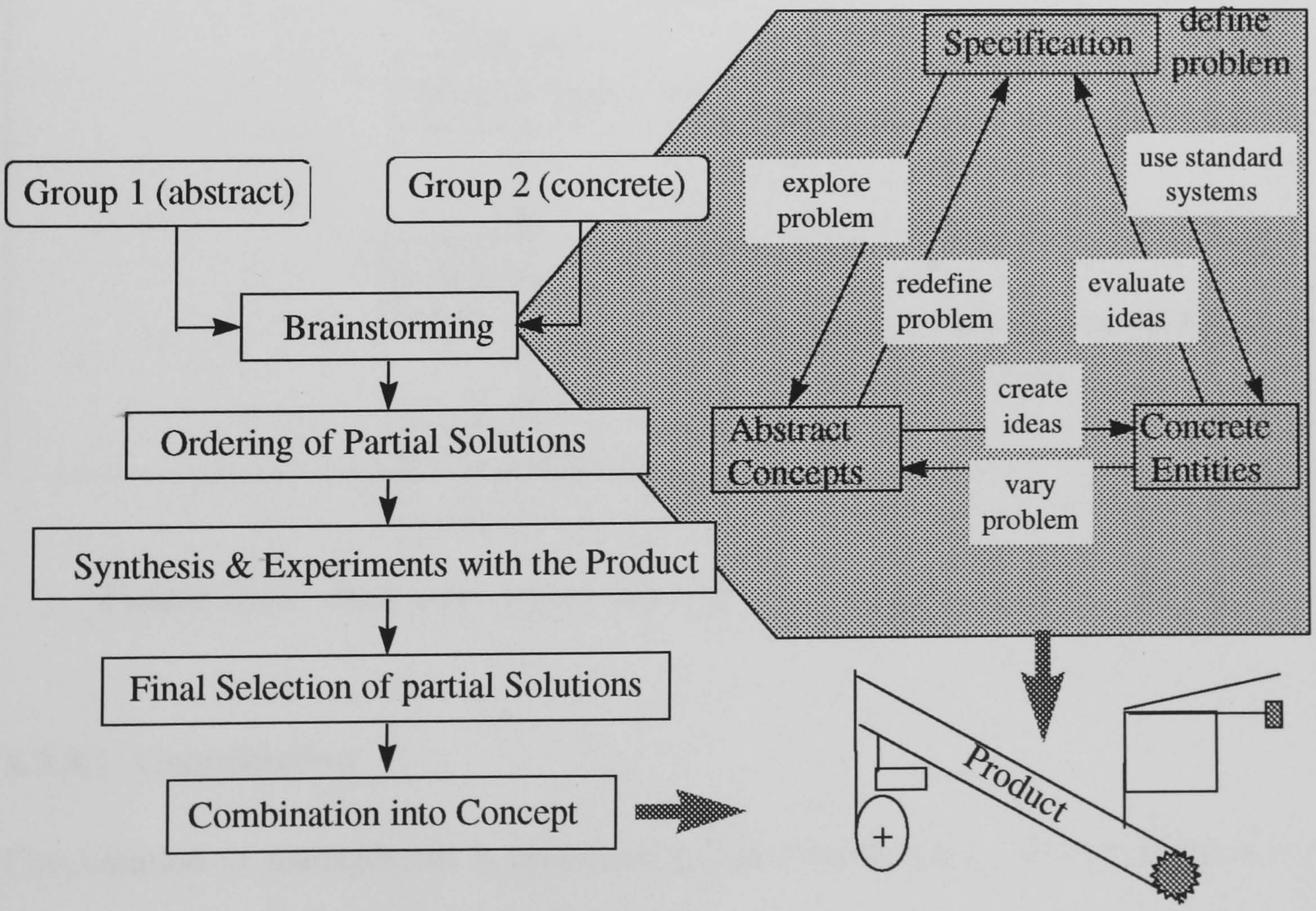


Figure 2.12 Teamwork moderation for systematic development (Bauert 1993a)

Bauert (1993a) gives a model for systematic moderation of creative teamwork as shown in Figure 2.12. While discussing the systematic approach to team work (Figure 2.13) for the development of a powered mobile arm support, Bauert et al. (1993b) states that role play within teams is important, and identifies eight roles (chief designer, designer, project manager, customer, inventor, analysis expert, standards expert and method developer) for a team to perform effectively.

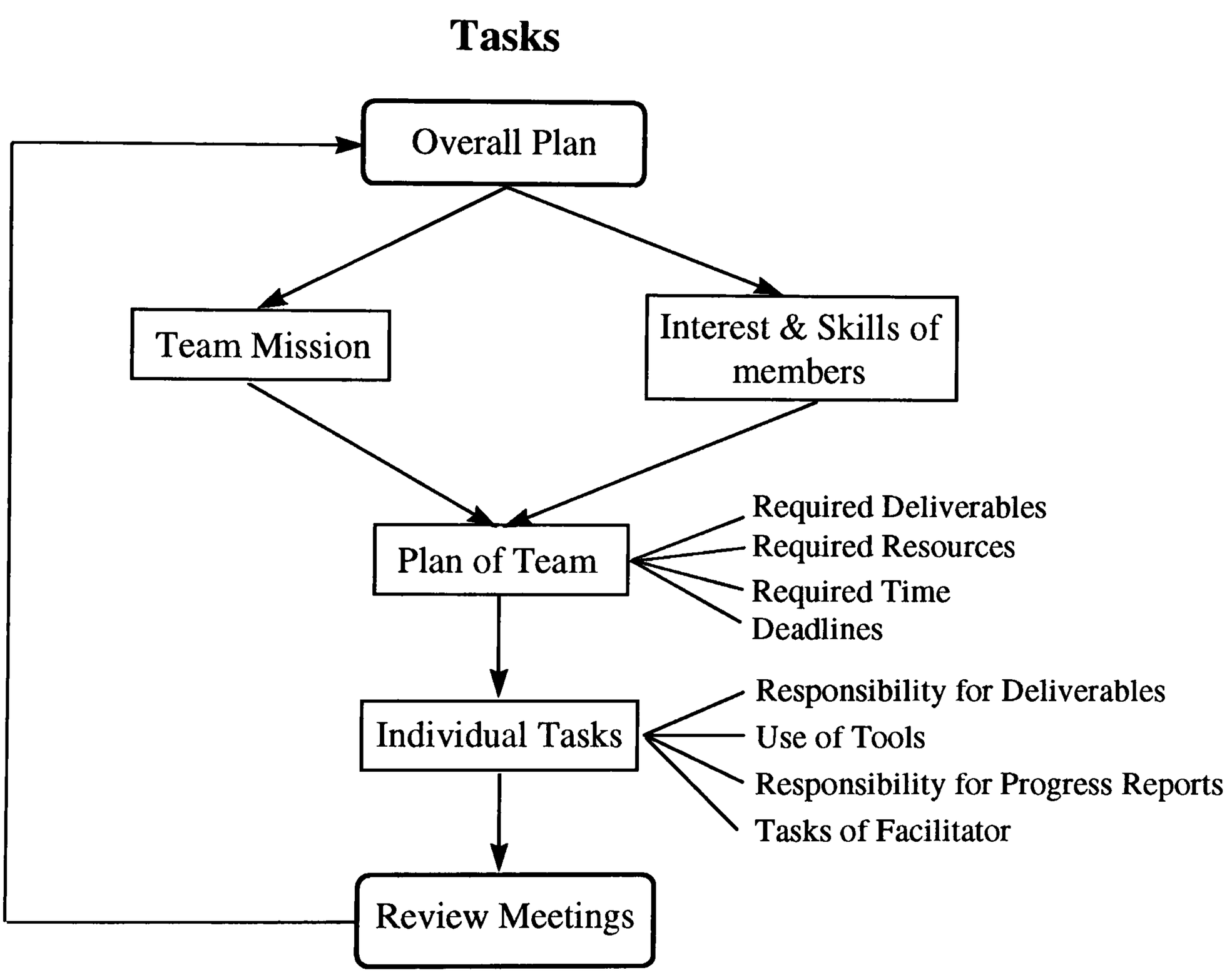


Figure 2.13 Systematic approach to teamwork (Bauert et al. 1993b)

2.3.3.1 Coordination

Coordination or management is central to product introduction. Coordination is the process by which individual efforts in an environment are exerted towards achieving a set of goals (Londono et al. 1992). It refers to how each participant's task can be

managed so that it integrates well with the results of others. Central to coordination is the ability to manage the processes and the individual members. Without effective coordination, work can be duplicated, decisions can be made with incorrect information leading to sub-optimal decisions, and conflicts can remain hidden, allowing cost to rise, quality to fall and time to be lost (Londono et al. 1992; Crabtree et al. 1993). In order to have an effective coordination, the product introduction work flow must be managed; information must be available to the right people at the right time for effective group decision making; conflicts must be detected as early as possible and resolved quickly; and to, ensure that product introduction converges to the required product with the necessary product functionalities, the progress of the entire process must be monitored.

According to Londono et al. (1992), coordination is achieved through a global database with appropriate control mechanisms to access information in the database. Ranky (1994) points out that "Managing the total information flow to gain competitiveness - that's integration". According to Crabtree et al. (1993), better team co-ordination includes "..co-operative work, activity and task-management services and decision tools fed by knowledge from the shared-information model". Coordination is enhanced when the responsible workers get to know about the completion of tasks on which they are dependent (Cleetus et al. 1996). On completion of tasks, information generated by the tasks need to be communicated to the resources responsible for the dependent tasks (or) the resources should be informed about the readiness of the generated information. Thus, coordination can be brought about by communication. A coordination framework is responsible for communicating work

and for ensuring focus and progress towards a goal (Smith 1991). Since the goal of the product introduction process is achieving a product definition of the product that meets the required product functionalities, a framework that integrates the process, resource, information generated and the product functionalities becomes essential.

2.3.3.2 Communication

One of the key factors for teamwork to be effective is good communication, i.e. the ability of team members to build on each others' ideas. As the team is involved in a continuous improvement cycle of plan, implement, monitor and improve collection of activities vital to the success of the PI project, there should be good communication between the teams executing the overall project. The management team should communicate plans, policies and targets down the project pyramid (Figure 2.14). The results of the process (product information) and proposals for improvements should be communicated up the pyramid. As the process must link problem solving cycles in time, it is necessary to know what to communicate, when to communicate, whom to communicate, and in what form to communicate.

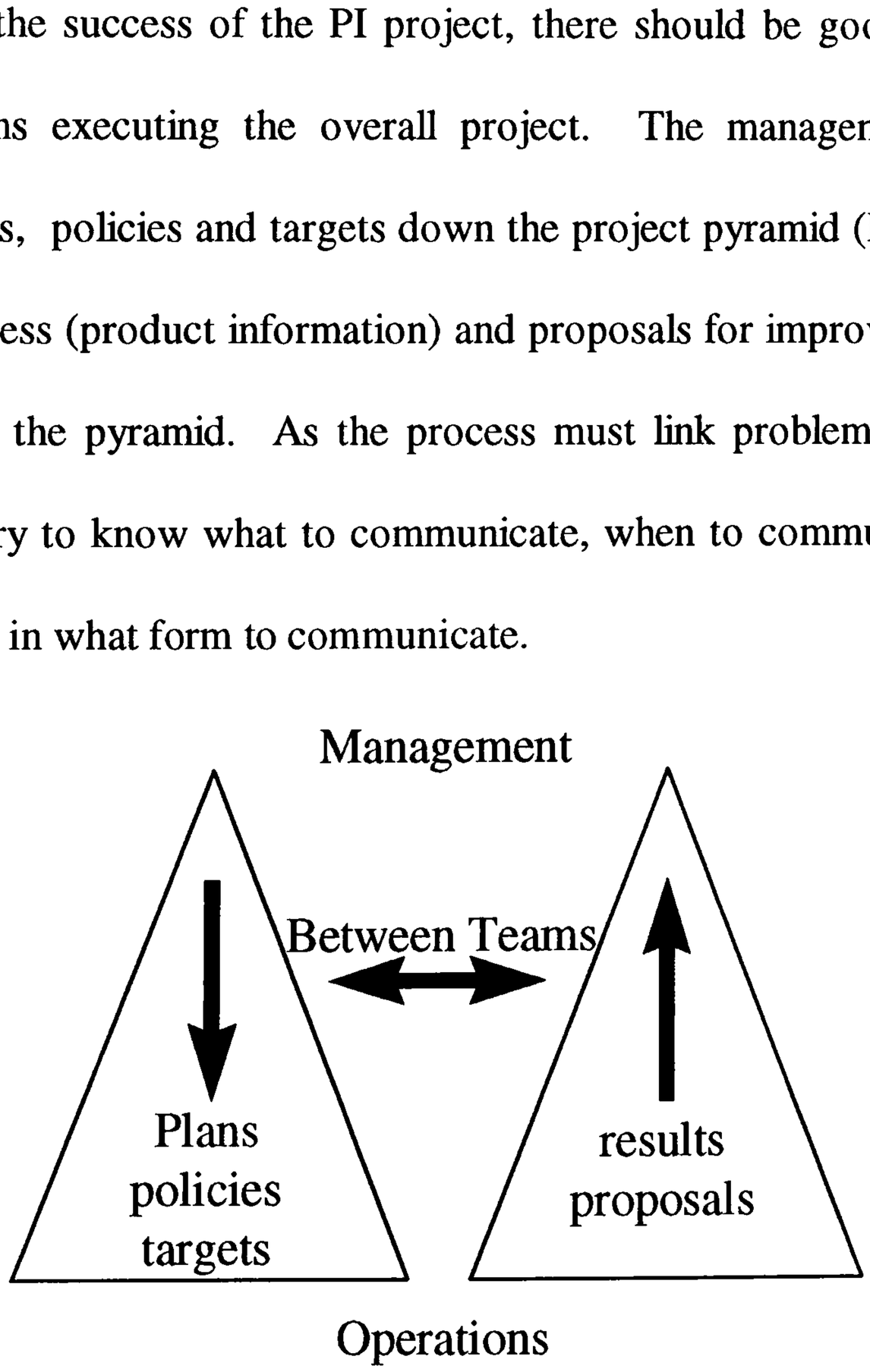


Figure 2.14 Communication up and down the project pyramid and across the teams

A framework for a multi-level model for the design process in an industrial context (Figure 2.15) shows the levels of communication. When concurrent engineering is used, communication among team members - communication between members within a team, between members of different teams of same domain, between members of different domains (i.e. horizontal communication) is very essential. Vertical or hierarchical commitment is achieved through the different roles of product champions and sponsors (Doyle 1994). Horizontal communication can be provided only when the dependencies among the activities in same/different domains are known as it is established based on the input-output dependency relationships among the activities.

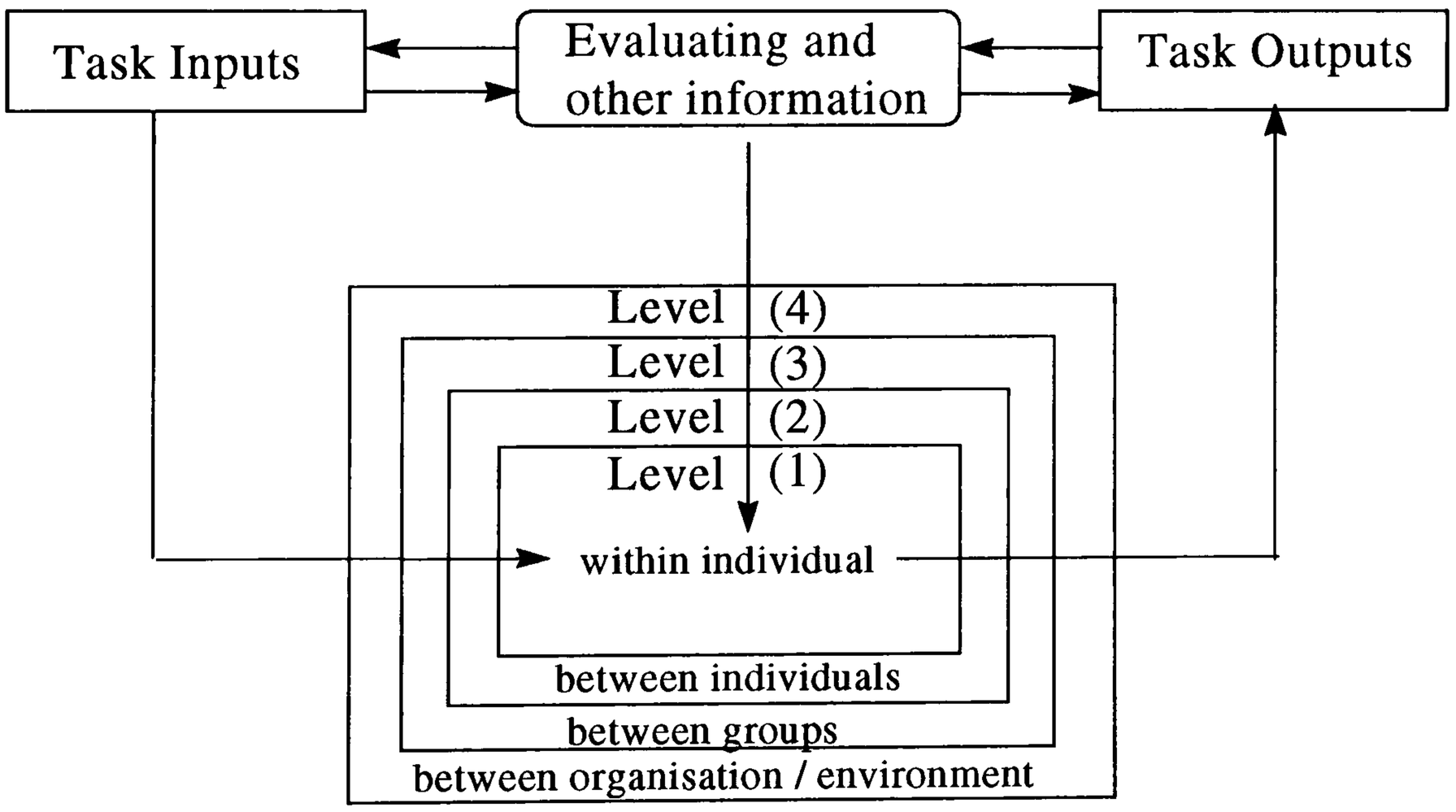


Figure 2.15 Framework for a multi-level model (Hales 1987)

The foundation for teamworking is sharing of information. Adoption of team work is difficult due to the rigid hierarchical structures of organisations and a lack of tools that enable co-ordination, and communication of information among team members working in a heterogeneous, geographically distributed environment.

2.3.4 Information systems

Product introduction process is information intensive (Anderson and Crawford 1989) and can primarily be seen as an information transformation process, transforming information about ideas, demands and technological advancements into the detailed information from which a product or system can be made (Wallace and Bligh 1996; Drongelen et al. 1996). The problem solving aspect of the PI process needs information conversion (Figure 2.16) (Pahl and Beitz 1984). The process is as effective as the decisions made within it, and as efficient as the speed with which the information required for each decision is made available (Angus and Murdoch 1993). Information is received, processed and transmitted, and it is also necessary to store the information. The foundation for team-based co-operation is sharing of information, and the teams require an information infrastructure that supports this. Apart from this, cross-functional integration has been identified as an important ingredient of effective product introduction (Rosenthal 1992). The cross-functional integration can be promoted by the following set of five information processing functions:

1. capture of the evolving information
2. providing the information assemblies necessary for an activity
3. enabling the information flow among the activities
4. communication acceleration
5. enhancing information translation.

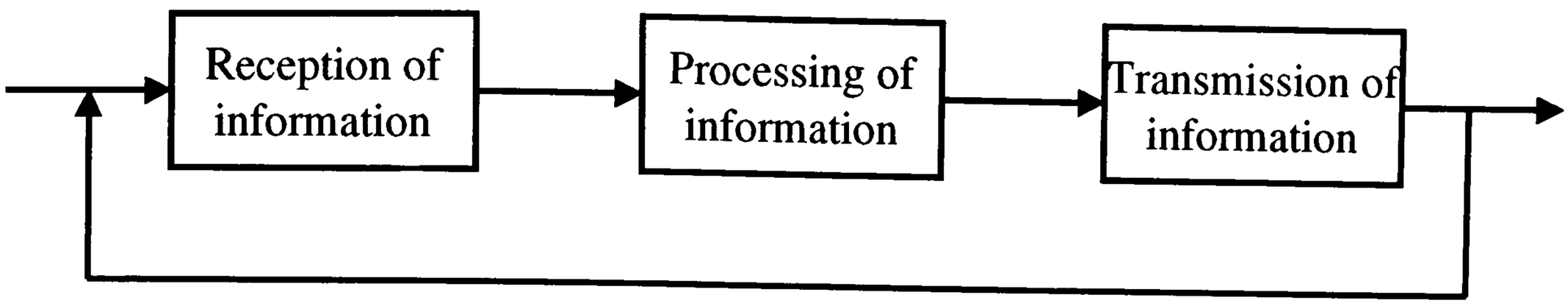


Figure 2.16 The conversion of information (Pahl and Beitz 1984)

Capture of the evolving information

As an idea is being analysed, a new product is being designed and developed, information is constantly accessed, interpreted, augmented, transformed and deployed. The basic elements of product introduction process are activities that require input information, take time to execute, and produce decisions or output information for input to other tasks (Eppinger et al. 1989 and 1994). After each new step, it may become necessary to upgrade or improve the results of the last, that is, to repeat it at a higher information level, and to reiterate until the necessary improvement has been made. In this cycle, the process undergoes revision and the product information evolves out of the process. The accuracy, consistency and availability of such information essentially determines the extent to which a new product achieves required functional capabilities and ease of manufacture (Rosenthal 1992). Hence it would be necessary to capture the evolving product knowledge and represent it in a structural form for easy dissemination. It is also necessary to know when and to whom, in what form to disseminate the information. The storage methods employed for information used by the product introduction teams has a great impact upon the success of its subsequent use and reuse. An important aspect of information of product introduction is that it cannot be dissociated from the process/activity that uses and/or creates this information. As the activities of product introduction process

generates information, activity model would become an essential part of the information model.

Information assemblies necessary for an activity

Multidisciplinary teamworking in product introduction necessitates collection and integration of detailed product and process information from varied and innovative perspectives (functionality, physical structure, materials, tooling etc.) on many aspects. An activity requires information of different type, content and range (Paul and Beitz 1984). Thus, multiple sources of information have to be referred to in order to collect the information necessary for an activity of the PI process (Figure 2.17). This information may be traditionally available in various databases. Apart from the technical information, it is necessary to consider the managerial information such as PI resources (teams), time constraints and cost constraints. It would be necessary to

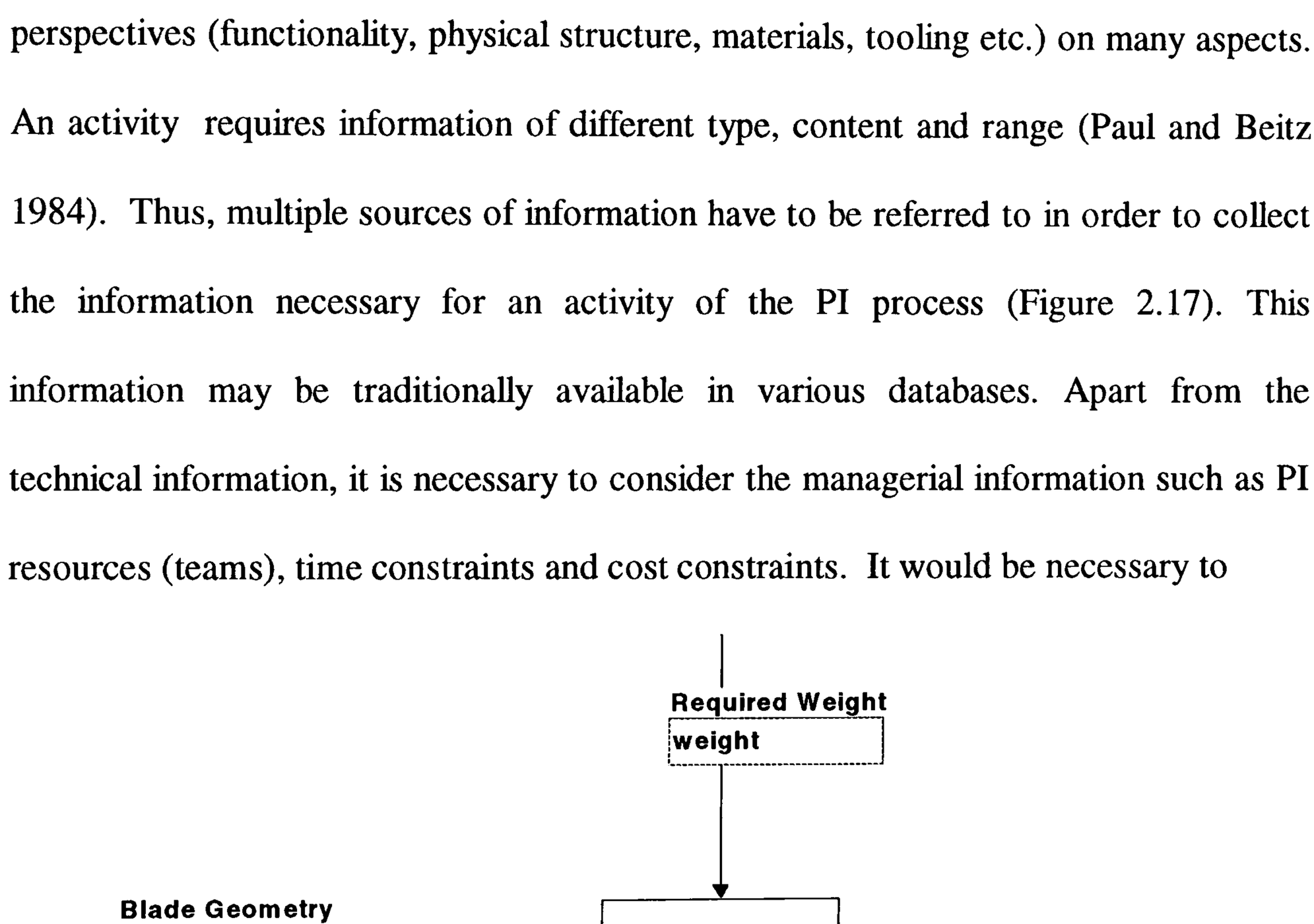


Figure 2.17 Set of technical information assemblies associated with an activity

routinely provide the information to the teams that carry out the activities. Thus, it becomes necessary to assemble sets of related information that are traditionally available at the same time but not in a consolidated form facilitating easy access and use. This information processing function is termed as "focused information assembly" (Rosenthal 1992).

Better access to information is achieved with the "organisation...unified with mechanisms for storage, control, and retrieval of all information and data relevant to the product" (Crabtree et al. 1993). In order to provide the focused information assembly, first it would be necessary to store the information generated by the activities of the PI process in an unified information model and second, it would be necessary to provide information translation mechanisms in the following two ways (Figure 2.18): ① translation from the output information assemblies of the activities to the unified model and ② translation from the unified model to the input information assemblies of the activities.

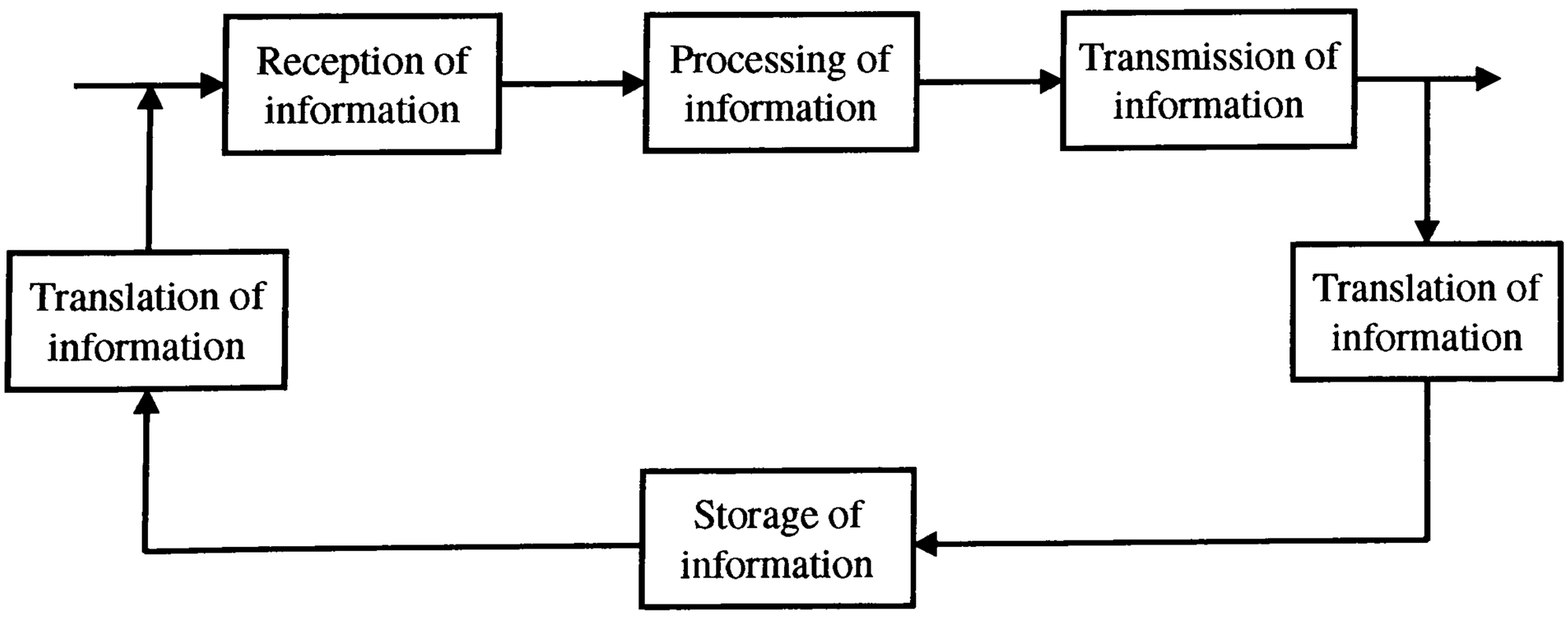


Figure 2.18 Conversion of Information

Information flow among the activities

Problem solving can be considered as a form of information conversion, and it demands a constant flow of information (Pahl and Beitz 1984). The nature of the task determines the information needs of the project team (Hauptman 1986). Information delivery to a decision maker is clearly a critical factor in the effectiveness of the decision making process (Angus and Murdoch 1993). It would be essential to synchronise and manage the flow of the information and the resources (capacity in time) available. The interdependent or coupled tasks model given in Figure 2.19(d) is a more realistic diagram of simultaneous engineering, where information transfer is essential and iteration is typical. The basic effects of simultaneous work can be represented by a bi-directional information flow between tasks worked on in parallel (Figure 2.20).

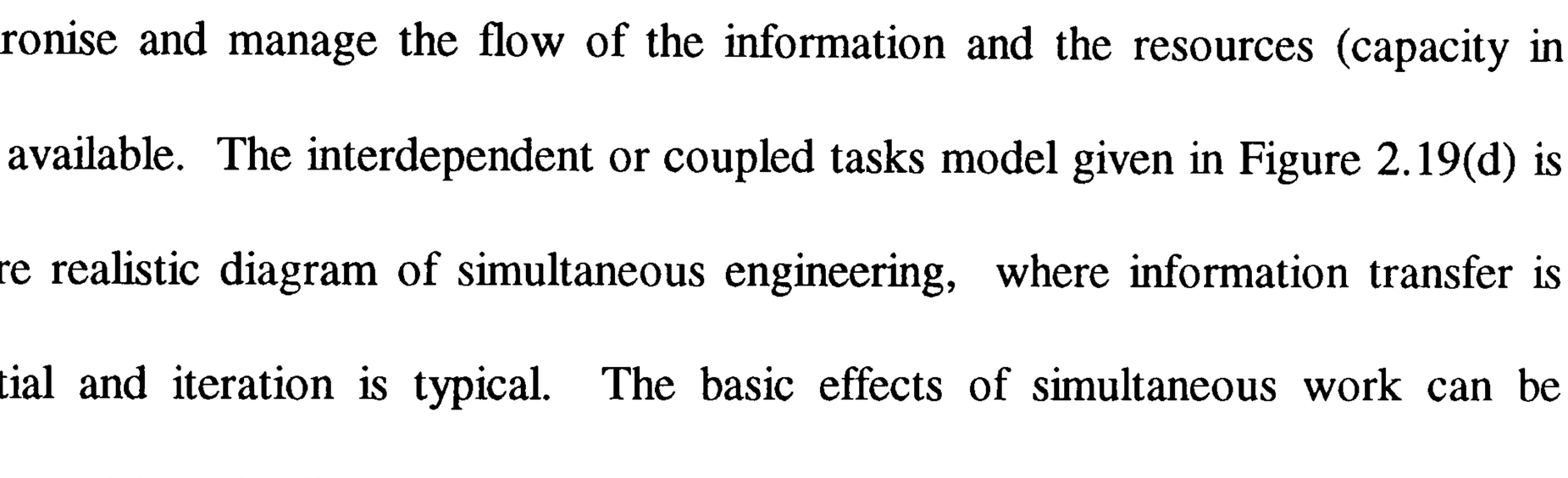


Figure 2.19 Various types of tasks

The first direction in Figure 2.20 aims at early providing of results of the leading task 1. As this task is not yet completed, these results are just partial, containing some initial decisions already made, and are vague. Nevertheless uncertain or vague information can represent valuable orientation for following or overlapping tasks. The

second direction is backwards from task 2 to task 1. In task 2 input information is analysed under task specific conditions. For example, the suitability of the design methods in achieving the critical parameter values and effects can be reported back early. This analysis provides basic information for a productive feedback and allows iteration loops (Krause and Ochs 1992).

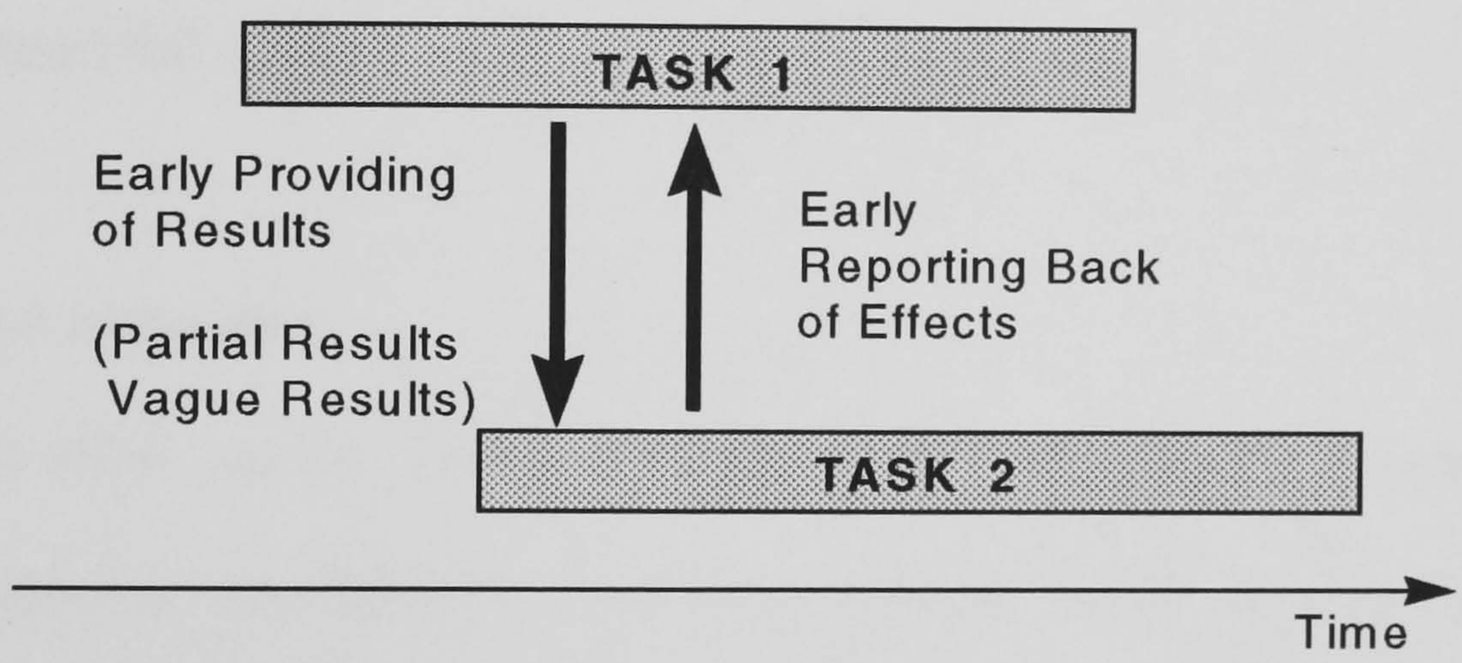


Figure 2.20 Bi-directional information flow within simultaneous tasks (Krause and Ochs 1992)

Communication acceleration

Having the right information at the right time at the right place in the right format throughout the product introduction process facilitates the decision making and thereby its success. It is widely agreed that the problem does not lie in finding the relevant information, rather in the time and effort involved in getting this information.

The timing of access to information is particularly critical to a project's success because it can directly affect the cycle time to introduce a new product. Design technologies and practices that allow downstream stages of the process to start earlier (such as concurrent engineering) may also reduce chances of oversight and error (and thereby reduce variability in the process), and allow evaluation of greater number of

alternative approaches to the particular task. Such technologies and practices achieve the function of communication acceleration (Rosenthal 1992). Providing appropriate process/product information to the project groups as soon as it is available will allow them to get an early start in activities that are critical to a successful product release. The pursuit of reduced product development cycle time is likely to be sufficiently important to make communication acceleration an important information processing function (Rosenthal 1992).

Information translation

The project effort can be hindered by the inherent breakdowns in communication brought about by the distributed and heterogeneous islands of information bases. What is needed is a facility that transforms sets of information from one point of view to another. This information processing function, called translation (Rosenthal 1992), should be explicit and as routinised as possible. For example, usage of the quality function deployment (QFD) techniques would facilitate translation of customer requirements ('nice writing flow' of a pen) into design specifications ('xx viscosity of pen ink and yy roller ball pressure in pen'); design for assembly (DFA) algorithm translates product design specifications to manufacturing engineering requirements (Table 2.1). QFD adds value to the product and the PI process by better co-ordinating information and people to meet the objectives and increase organisational capabilities. Translation enhancements can lead to improvements in effectiveness of the product introduction project (Rosenthal 1992).

Table 2.1 Information translation techniques

Technique	Translation
Quality Function Deployment	marketing requirements → product design
Design For Assembly	product design specifications → manufacturing engineering requirements, target unit cost → yield objectives for manufacturing ramp-up
Computer Aided Process Planning	product designs → {manufacturing routings, detailed process plan}
Bill of Materials plan	product part hierarchy → {purchasing requirements, assembly facility layout}
Value Engineering	{product functionality, service requirements} → {product cost, material guidelines}

Information technology tools such as Computer Aided Design (CAD), Computer Aided Engineering (CAE) and Computer Aided Manufacturing (CAM) systems, analysis and simulation programs and product data management systems are used in the research and development process. Fourth generation systems are found useful but they are hardly used in research and development either as a medium to store knowledge or as a means to get insight into and access to knowledge or as a means to actually convey knowledge (Drongelen et al. 1996). Based on the definition of knowledge as internalised information that has the value for the organisation, Drongelen et al. (1996) have come out with a knowledge management model (Figure 2.21) that visualises R&D projects and activities as a funnel with different stages, referring to the gradual uncertainty reduction by means of new acquired information from outside the company, and knowledge from the organisational knowledge base. The knowledge management is aimed at ① improving *knowledge push*, the knowledge handling activities (sifting knowledge from many pieces of internalised information, storing this knowledge, opening it up and conveying it to a user), ② improving *knowledge pull*, the activities that search for, acquisition of new information and

knowledge, creative work of transforming the inputs into new meaningful outputs including the evaluation and application of information and knowledge. The knowledge management model is at descriptive level, nothing is mentioned about the implementation concepts. The implementation of knowledge push and knowledge pull calls for various research issues such as 'how (in what form) to sift the knowledge', 'how to open it up', 'how to convey it'; this calls for the study and analysis of the relationships between the process that generates the knowledge and the knowledge, and their representations.

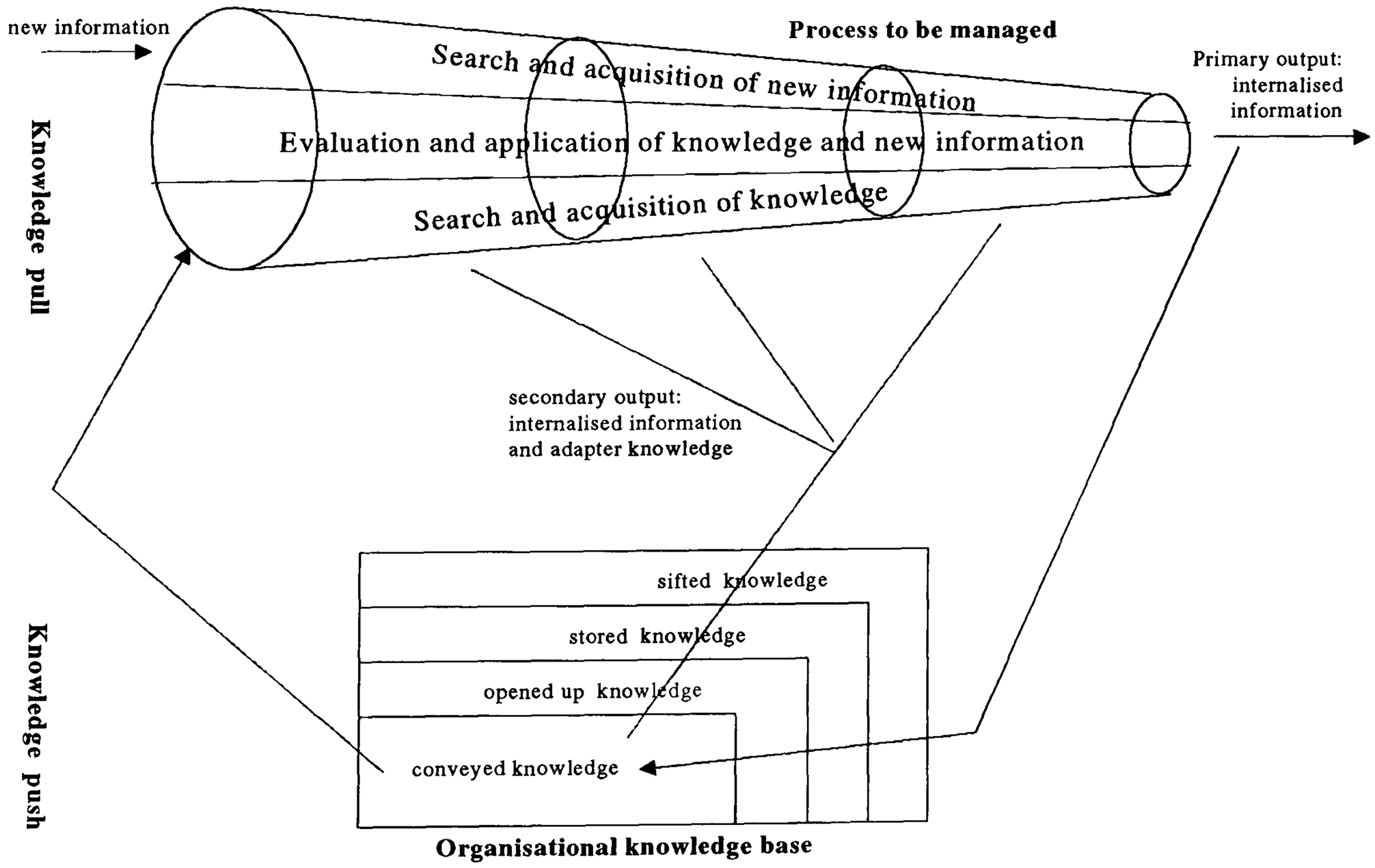


Figure 2.21 Knowledge accumulation and dissemination activities in the research and development process (Drongelen et al. 1996)

2.4 DISCUSSION

The PI process continually evolves and generates a great deal of knowledge, especially product information (Figure 2.22). Initially customer or market needs/requirements are translated into product functionalities, and a plan for the process of achieving the product functionalities is created, critical parameters are identified and their expected values are set up. Activities of the PI process are performed using various methods (mathematical analysis, simulation tools etc.) with the support of resources, and product information (information on product structure, design, production process) is generated as the result of the activities. The generated information (results) is evaluated against the expected values in order to make a decision, and this cycle continues until the required functionalities are obtained taking into consideration the time and cost constraints on the PI process. Product information evolves as the process that generates product information undergoes revision, and this evolving product knowledge would need to be captured and represented in a structured form for easy dissemination (Thirupathi and Roy 1997a and 1997b).

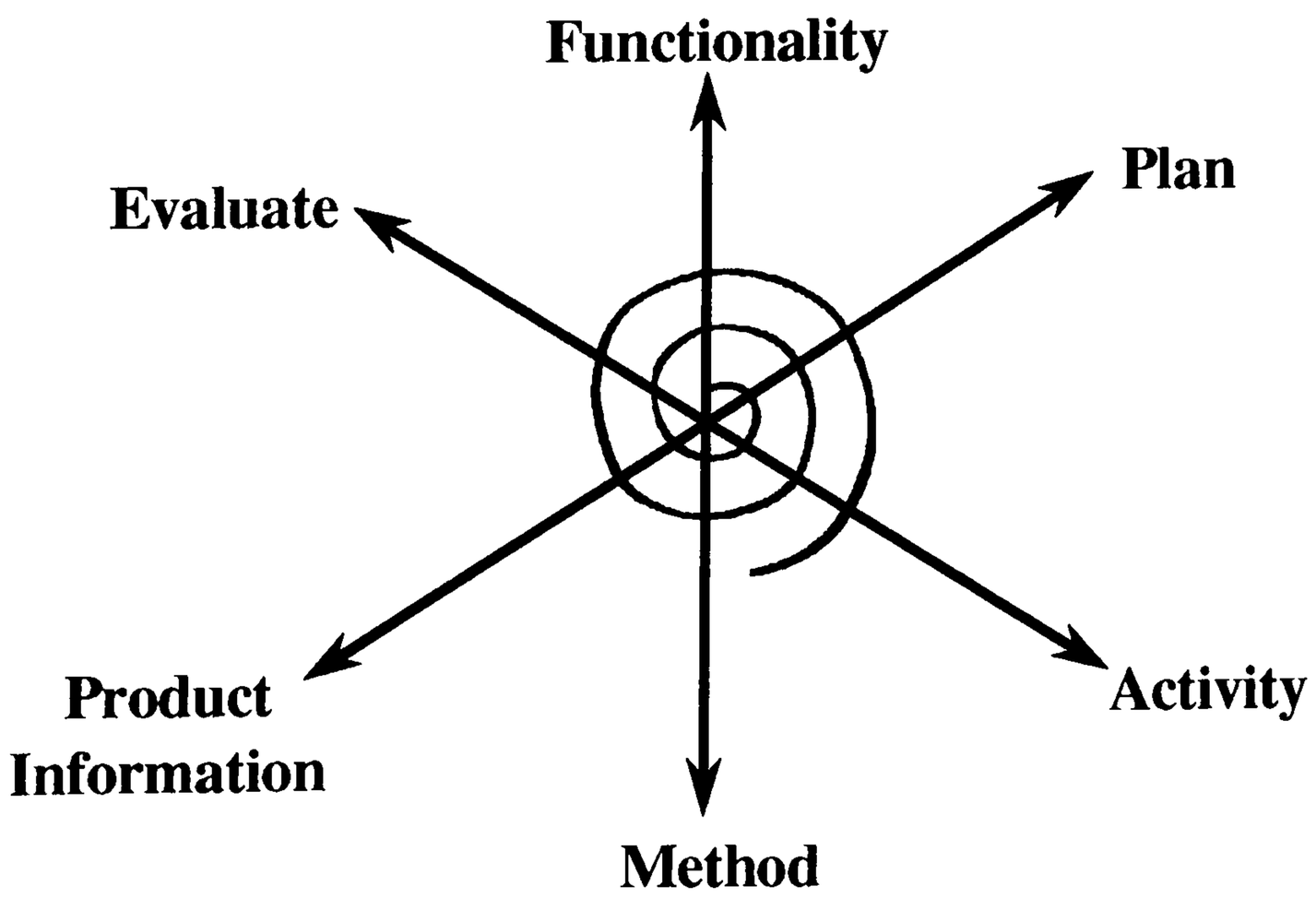


Figure 2.22 Continual evolution (Thirupathi and Roy 1997b)

Success in PI is a critical management issue (Zirger and Maidique 1990) and success in innovation has been strongly influenced by the effective management of the overall project. PI process management has to be seen within a complicated network of relations between the purpose of the project (product functionalities), the activities which output the product with the required functionalities and the responsible persons (or teams) carrying out the process. The factors that play a role in it are:

Project Management - strategy, targets for formulating and managing the project.

Technical Information - including the knowledge about technology, product functionalities, product and production.

Resources - the team which creatively and methodically carries out the project and is driven by its targets, and technical aids needed to carry out the various tasks.

Completion of project scope, work that must be done in order to deliver a product with the specified functionalities is measured against the plan, while the completion of product scope (the functionalities of the product) is measured against the required functionalities of the product. Both project and product scope management must be well integrated to ensure that the work of the project will result in delivery of the specified product. This necessitates integrating project management information as well as product functionalities and product information (Figure 2.23).

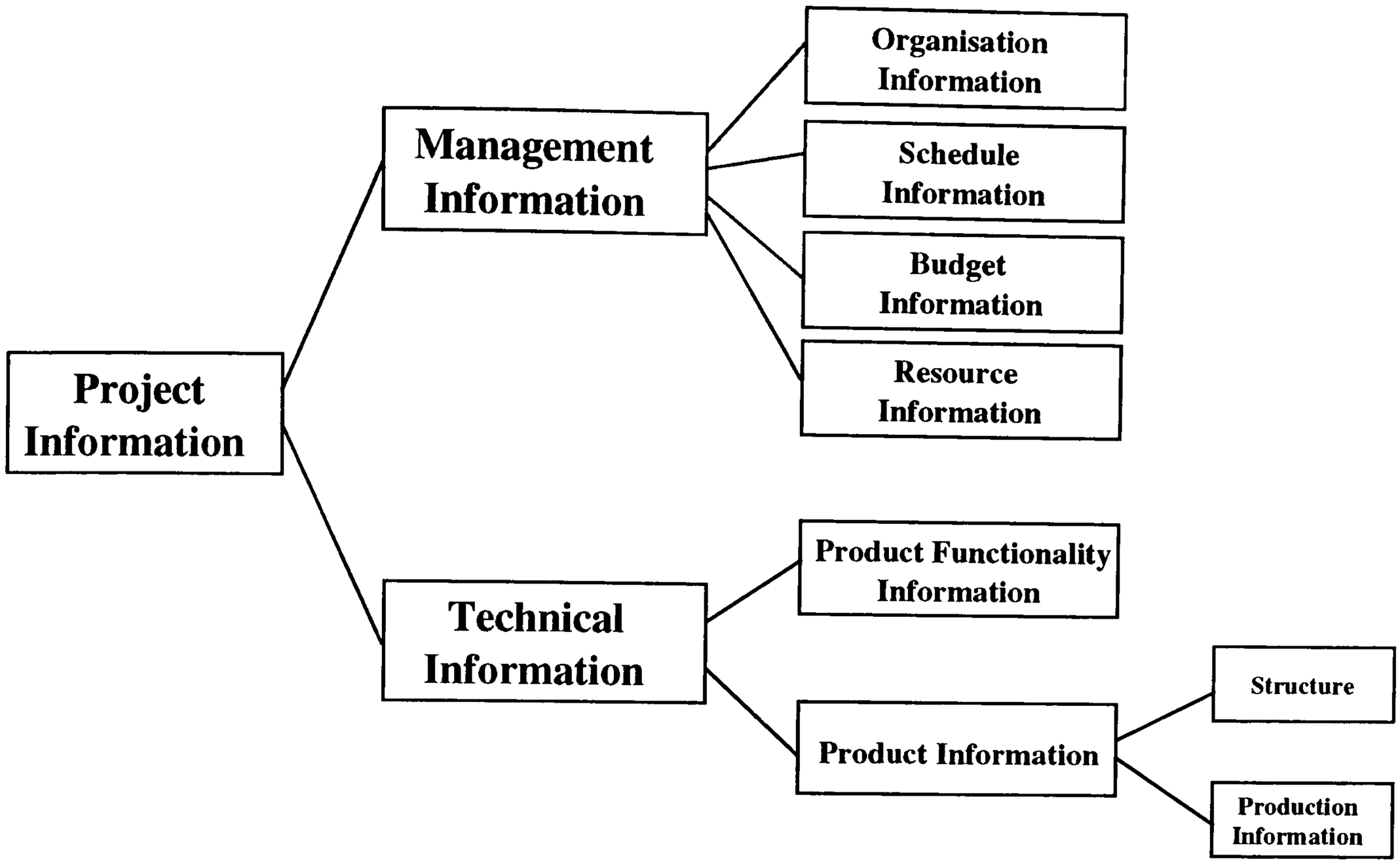


Figure 2.23 Product introduction information (Thirupathi and Roy 1997b)

The resources need to be planned and managed based on what is to be achieved (product functionality - technical information). This necessitates a link between resource management and technical information management. In existing information systems, resource management is separated from technical information management. An analysis on what is the link between these two, and how to represent the link would be essential.

It is also necessary to identify what parts of the PI process must be done in series and what can be done in parallel. This identification must be done at an early stage to fully benefit from the advantages of simultaneous engineering and reduce time to market (Hollins et al. 1993). There is a three stage approach for developing integration and concurrency or parallelism potential in the PI process - ① procedure-oriented

parallelism, ② object-oriented parallelism and ③ information-oriented parallelism (Eversheim et al. 1995). The procedure-oriented parallelism is based on activities and information flows where the activities are coupled to form a single procedure via their input and output information. Procedure-oriented parallelism is extended to include the object in the case of object-oriented parallelism where the degrees of potential for parallelism can be derived by dividing the activities up into the individual objects of the product. The flow of information is optimised in information-oriented parallelism, regardless of the activities defined input and output information. This allows for early transmission of information and feedback of information from downstream activities to the upstream activities. Using these methods, critical information on the product and the processes can be generated at an early stage, and passed on to the appropriate activities. Thus, representation of activity, information flow and information play a major role in the management of the PI process.

Speedy communication of information is fundamental to the management of the task and is one of the prerequisites for successful control. Hence, it would be necessary to capture the overall information flow network that underlies the whole project. The PI information include: product functionalities (aim of the process), existing product definition (input), activities, constraints (schedules, cost) attached with the activity, teams (resources) carrying out the activities, generated product definition (output), changes in product specification (input) and the relationships among them. As the information is generated and used by the process, and the information controls the direction of the process, it would be necessary to understand the link between the process and information, and use this link in the representation of the overall product

introduction information. Thus, effective management of the product introduction process requires the establishment of some model of the process, the information associated with the process, and the association between the two.

CHAPTER 3

3. INFORMATION MODELS

3.1 INTRODUCTION

The product introduction (PI) project is a process of gradually building up the right information and linking up the activities with required skills so that the project can meet its targets. Each step, or activity, in the workflow of the PI process has its own information needs, information input, and information output. Within an activity, people use information, and the end of an activity is characterised by information being prepared, signed off, and released. Between activities, information is transferred whenever an iteration or change occurs in the workflow. Information flow has to be synchronised with workflow so that the information is available when and where it is needed (Stark 1992). For effective management of the PI process, it would be necessary to link the process and the product information that controls the process, the product functionalities that the PI process aims at and the information on PI resources (teams); in other words, a process-based integrated information model would be necessary.

Information models provide a structured description of the information entities which exists within an enterprise, and the necessary relationships between them. The information model that will support the management of the product introduction process should ① capture and represent the product information and information on the process that generates the product information, ② provide immediate access to information about product and process, ③ offer immediate access to information

about required product functionality and achieved product functionality in order to control the quality of the outcome of the process, ④ allow access to the current state of the product and the process that generates the product, and ⑤ keep data to be shared by the PI team members in commonly accessible databases. The complexity of co-ordinating and structuring the design and development of such support systems would necessitate the use of formal modelling methodologies for their representation. These methodologies include techniques to model information and processes. Section 3.2 is dedicated to review in detail the modelling methodologies which have been developed to facilitate the design and implementation of the information systems for complex environments. The idea of combining modelling methodologies in integrated frameworks for the development of system concepts has resulted in the creation of reference models for enterprise integration, and a review of the relevant reference models is given in section 3.3.

3.2 MODELLING METHODOLOGIES

A method is an organised, single purpose discipline or practice and a modelling methodology refers to a class of similar methods (Molina 1995). Various methods have been developed to help in modelling the different aspects of information systems for manufacturing environments. The review does not include all the modelling methodologies; however the more popular methods used in design and development of information systems which support engineering activities are considered. The modelling methodologies have been classified in the following groups:

Data/information modelling methods	:	Data Flow Diagrams, Entity Relationships Diagrams, IDEF1, IDEF1X, NIAM and EXPRESS
Process modelling methods	:	IDEF0 and IDEF3
Hybrid modelling methods	:	Object-Oriented methods, Object-Relational methods and Meta-modelling.

3.2.1 Data / information modelling

For data to be useful in providing information, they need to be organised so that they can be processed effectively. The objective of data modelling is to organise data so that ① they represent as closely as possible the real world and ② they are amenable to representation by computers. A data model consists of three components: ① a collection of data structure types which forms the basic building blocks of any database that conforms to the model; ② a collection of operators or inferencing rules, which can be applied to any validly defined instance of the data types in order to retrieve, or derive data from any part of those structures in any combination desired; ③ a collection of integrity rules, which implicitly or explicitly define the set of consistent database states - these rules are sometimes expressed as insert-update-delete rules (Codd 1980; Date 1991). Thus, data modelling allows the description of the information structure relevant to a system in an implementation-independent format named a data model. Data modelling methods have been developed as aids to database design, and support the modelling of entities and the relationships between entities.

Information modelling is an outgrowth of data modelling and is related to the identification, representation and composition of the data, information and knowledge that describes real objects. The task of information modelling is to provide a sound basis for mapping between the world of interest and a representation of it that can be used as a specification for defining a database and/or applications (Eastman and Fereshetian 1994). The ultimate goal in information modelling is to formulate descriptions of real world information so that it may be processed and communicated efficiently without any knowledge of its source and without making any assumptions. Data modelling is concerned with specifying the appearance and structure within a computer system of the data that represents particular types of information. Information modelling has the goal of describing information so that the representative data could be computer processed. Thus, one distinction between data and information modelling is that one is explicitly targeted for computer processing of the data while the other has the potential (which may, of course, be realised) for such processing. The other major distinction is in the treatment of the interpretation rules. In an information model these must be made explicit and formally documented. In a data model, the rules are typically implicit; even if they are made explicit, they are informally documented (Schenck and Wilson 1994). Therefore data modelling techniques can be used to develop information models if the appropriate formal documentation is generated during the modelling exercise.

Data Flow Diagrams (DFD) - Data Flow diagram is a graphical representation of flow of data in a system, various data stores and processes/functions to/from which they flow (Cutts 1987; Pressman 1992). It is generally used to determine the information contained within a system, and may be partitioned into levels that

represent increasing data/information flow and functional detail. DFDs emphasise flow of data but not flow of control (DeMarco 1979). They do not supply the explicit indication of the sequence of processing.

Entity Relationships (ER) model - The ER model views the world as consisting of entities and relationships between them, where an entity is anything that can be distinctly identified, and a relationship is an association between entities. The modelling method is simple and is used routinely for system analysis and design. The Entity-Relationship (ER) model is a way to unify the network and relational database views (Chen 1976). The network models provides a natural view of data by separating entities and relationships, and they represent only unidirectional relationships. The relational model is based on relational theory and can achieve a high degree of data independence, but may lose some important semantic information about the real world. This is due to the fact that the same domain name may have different semantics in different relations. The ER model incorporates some of the important semantic information about the real world and the semantics of data are much more apparent; it also represents both directions of the relationship by specifying the roles of both the entities involved in the relationship. Hence, it can be used as a framework from which the other data models can be derived. The mapping of ER model into a relational database design is simple and straightforward. But, ER model does not support inheritance hierarchies.

IDEF1 - IDEF (Integrated Computer-Aided Manufacturing (ICAM) DEFinition) methods are used to perform modelling activities in support of enterprise integration. The original IDEF methods were developed for the purpose of enhancing

communication among people who needed to decide how their existing systems were to be integrated. IDEF1 was designed as a method for both analysis and communication in the establishment of requirements, and requires the active participation of the information users. This serves to accurately model the organisation by forcing the users to think about how and where the information is being used and managed. IDEF1X (Data Modelling Method) was developed to assist in the design of semantic data models, which are extensions of the Entity-Relationship models. The extensions that they possess allow the capture of more meaning in the model by providing the designer and user of a database with a formalism whereby a substantial portion of the semantic structure of the application environment can be clearly and precisely expressed (Hammer and McLeod 1981; Peckham and Maryanski 1988). Semantic data models are more difficult than ER as they involve more complications in expressing the semantics of the information that is being represented. IDEF1X supports supertype/subtype hierarchies and it has a strong relational feel as the modelling process is based on the third normal form. In relational models, a relation is said to be in third normal form if it satisfies the following conditions - no repeating attributes in the relation, every nonkey attribute is fully dependent on the primary key, and no functional dependencies between nonkey attributes.

NIAM - Nijssen's Information Analysis Method (NIAM) is based on the binary-relationship approach. It relies on functional transformation where functions transform information from one form to another. NIAM is one of the main information models used in the Product Data Exchange using STEP (STandard for the Exchange of Product model data) (PDES). Like ER, NIAM has two major primitives - object and role, and it can represent constraints such as uniqueness, equality, subset, exclusion

and disjoint on objects and roles. It has only a limited support for object-oriented databases as methods cannot be specified and relations within a composition (for example, the relation between the geometric properties of a product) cannot be expressed (Eastman and Fereshetian 1994). NIAM has no features that can support the evolution of the information model once it has been implemented.

EXPRESS - EXPRESS is a conceptual schema language that has been developed as part of the PDES/STEP project. It was commissioned as the physical store for holding product models, and mainly deals with the information that is used or generated during product manufacturing. It represents entities using attributes to represent the data, and rules to represent the behaviour. It also allows classification of entities into subtype/supertype structures. An information model in EXPRESS consists of schemata that includes the definitions of entities, rules, relationships and rules on relationships (Schenck and Wilson 1994), but the rules defined cannot be executed. EXPRESS provides a powerful group of type constructors for building complex data structures, and also provides strong capabilities for defining the structures often developed in object-oriented databases. EXPRESS-G is the graphical representation of EXPRESS. EXPRESS is only for specification and is not executable. It does not reflect any dynamic and evolutionary nature of the information.

3.2.2 Process modelling methods

A process model is a representation of a process and its related components presented in a time-dependent fashion. A process model also captures the decision logic that may exist within the process.

IDEFØ - IDEFØ (Function Modelling Method) was designed to allow a description of a system's functions through the process of function decomposition and categorisation of the relations between functions (i.e. in terms of the Input, Control, Output and Mechanism classification (ICOMs)). An IDEFØ model helps to organise the analysis of a system and to promote good communication between the analyst and the customer, and is useful in establishing the scope of an analysis, especially for a functional analysis. But, the model does not explicitly represent the precedence relationships among the functions/processes.

IDEF3 - IDEF3 captures all temporal information, including precedence and causality relationships associated with enterprise processes by providing a structured method for expressing knowledge about how a system, process, or organisation works. The resulting IDEF3 descriptions provide a structured knowledge base for constructing analytical and design models. These descriptions capture information about what a system actually does or will do and also provide for the organisation and expression of different user views of the system.

3.2.3 Hybrid modelling methods

The hybrid modelling methods allow a system to be described by modelling data, process and behaviour in combination. The integration of different modelling methods enable the construction of models that reflect the reality.

Object-Oriented method - The object-oriented (OO) modelling is a way of building models that are organised around real-world concepts. The fundamental construct is

the object that combines both data structure and behaviour in a single entity (Booch 1986; Rumbaugh et al. 1991). An object model captures the structure of a system by showing the objects in the system, relationships between the objects, and the attributes and operations that characterise each class of objects. The object-oriented modelling method allows for data abstraction, inheritance, information hiding, dynamic binding and polymorphism. Inheritance of objects allows common structure to be shared among several similar subclasses without redundancy. Thus, they promote sharing at different levels, and have become very popular in the implementation of product data management systems in the manufacturing industries.

Object-Relational method - Object-Relational (OR) method provides the foundation for integrating relational and object technologies, which conspicuously lacking in current approaches to such integration (Date and Darwen 1998). The relational and object-oriented data models are combined into a single coherent unified model called object-relational model (Kim 1995; UniSQL Inc. 1995). It tries to blend the best of both approaches (high performance due to the procedural query language of the implemented relational models, data abstraction, inheritance and support for multi-media data, natural hierarchy of the object-oriented model) in order to overcome the drawbacks when they are applied individually. In this fusion process of object-oriented and relational models, relational model is considered as a subset or special case of the object data model (Bowen Inc. 1995). Thus object-relational data model offers a unique combination of relational and object capabilities. It restores the pre-relational notion of pointers and navigation; retains the flexibility, data independence, and standard query language of relational products; and adds the object-oriented notions of inheritance and encapsulation.

Meta-modelling - Meta-modelling method allows the design of integrated models. Each of the modelling method discussed above is good in describing the information from a particular perspective. Modelling the information in manufacturing, such as information related to product introduction process, requires the use of more than one modelling method. In such cases, it is necessary to integrate multiple models in a way that they more or less appear like one model to the user. In order to do this, the systems integrator needs to pay careful attention to which model supports which concepts, and how the concept of one model relates to concepts of another model. This analysis provides a list of concepts and their relationships, i.e. a meta-model, which can then be used to decide which data is kept where, and how tools that operate on models are supposed to talk to each other. Thus, the meta-modelling method builds the meta-models that are the foundation for data integration. In the field of software engineering, the method is applied to process modelling, and its goal is to support process improvement and management (Kokol 1993).

Although all of the modelling methods discussed can be used in the design of information models, they do not satisfy all the requirements for modelling the product introduction information. They lack a comprehensive framework to support the complete life cycle of an integrated information system from concept through implementation. Hence reference models and architectures for integrating manufacturing activities and enterprises have been designed for the development of integrated information systems.

3.3 REFERENCE MODELS FOR INTEGRATION

Various reference models, frameworks and architectures have been developed for use in information and manufacturing system development. They provide general representation of different aspects of a system, and can be referenced at various stages of the system life cycle (e.g. requirement definition, system modelling, design and implementation) (Molina 1995). A framework refers to an organised representation of characterised situation types that occur during an information system life cycle for enterprise integration; each situation specifies tasks, methods and tools which can be used to support a particular development situation (Zachman 1987; Molina 1995). An architecture denotes the information system architecture in terms of databases, networks, operating systems and integration utilities required for the enterprise integration. The underlying methodology in the context of a reference model for enterprise integration is a detailed process model, with guidelines of how to perform the development activity. The models that have an integration infrastructure for enterprise management, process management or information management are considered for study. The integration models considered include : Open System Architecture for Computer Integrated Manufacturing (CIM-OSA) that aims at enterprise integration, Architecture of Integrated Information Systems (ARIS) that provides a framework for the development of integrated application systems, Systematic Concurrent design of Products, Equipments and control Systems (SCOPES) framework that integrates the engineering and production knowledge, GRAI Integrated Methodology (GIM) that uses information system as the link between decision system and physical system, and Information Sharing System (ISS) that aims at enterprise integration through a model-based information directory.

3.3.1 Open System Architecture for Computer Integrated Manufacturing

The Open System Architecture for Computer Integrated Manufacturing (CIM-OSA) was designed under an initiative of the European Strategic Program for Research and Development in Information Technology (ESPRIT) programme. The goal was to develop guidelines that will support all levels of management in their strategic, tactical, operational planning as well as in the direct operation in the shop floor. The CIM-OSA architecture has two major themes:

- how to model the enterprise
- how to provide an integration infrastructure.

The architecture includes the integration infrastructure and constructs to model the enterprise. It consists of three levels of modelling abstractions: generic, partial and particular; and three different levels of modelling for each of the abstraction levels: user requirements specification, design specification and implementation description. For each abstraction level and for each modelling level, another dimension for categorising models termed as views is identified; CIM-OSA proposes four different views: function, information, resource and organisation (Figure 3.1).

Each enterprise function contains three parts: ① a functional part consisting of objectives, constraints, action to be performed, description of the inputs (material, data, controls and resources) and the expected outputs, ② a behaviour part in which an enterprise function is decomposed into a further lower level of enterprise function, consisting of associated lower level functions, and ③ a structural part consisting of relation between enterprise functions and their associated parents and/or children (Panse 1990). The domain process is decomposed into the following levels: business

process → enterprise activity → functional operation. Functional operations are executed by functional entities, which are independent closed units able to receive, send and optionally store data and/or material. The communication between functional entities is transaction-oriented, and the dependencies among the functional entities and/or the transactions define the order of communication but these dependencies are not addressed in CIM-OSA.

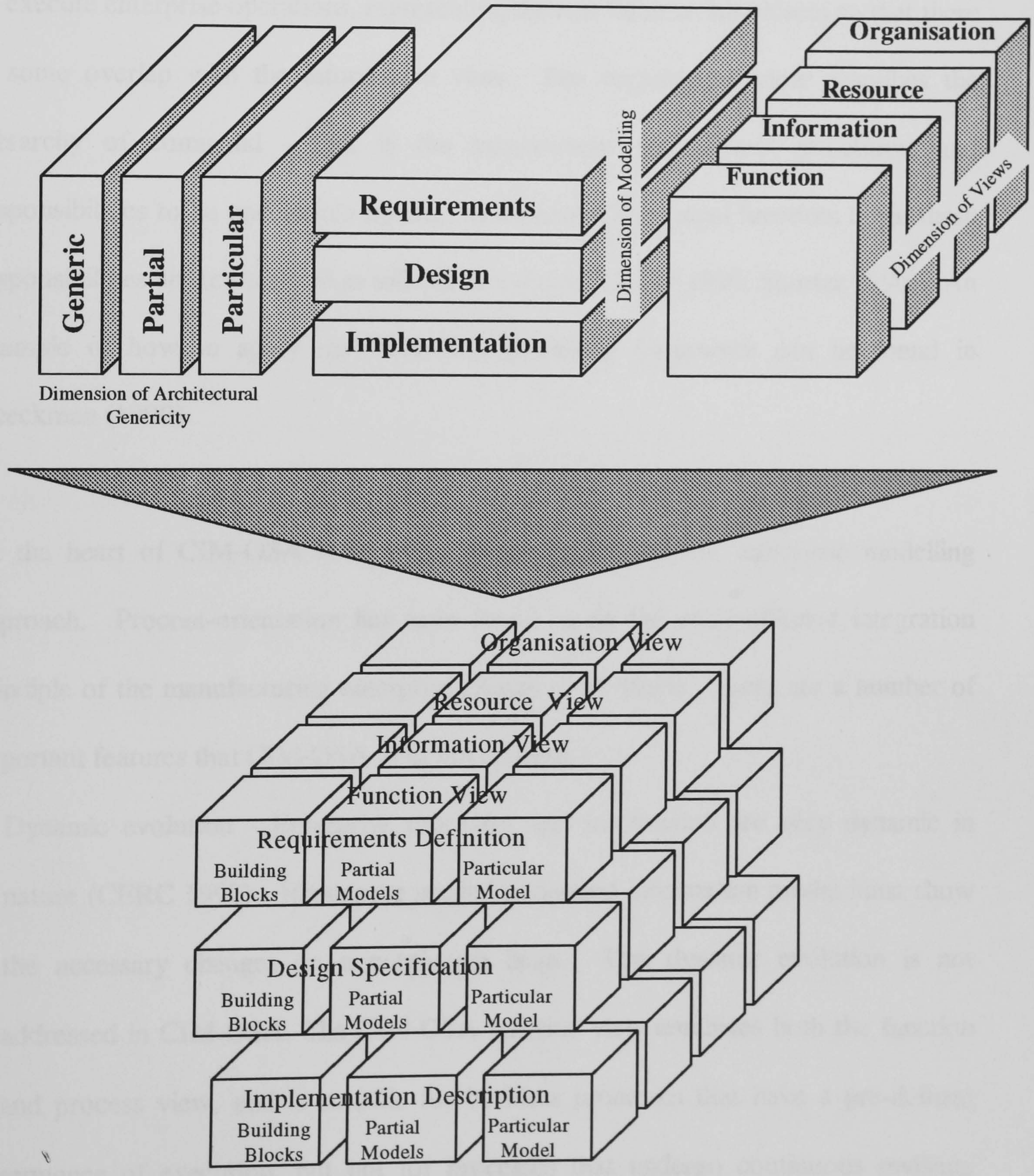


Figure 3.1 CIM-OSA architecture (Panse 1990)

In the information view, all business processes, enterprise activities, data, controls, resources and physical objects are described as information objects and they can be observed from different object views. Three schema approach for enterprise object views are defined : external schema (creation of information view), conceptual schema (entity-relationship approach) and internal schema (how the information is physically stored in the computer system). The resource view emphasises the resources needed to execute enterprise operations, representing them as information objects so that there is some overlap with the information view. The organisation view describes the hierarchy of command - that is the management and control structures, and responsibilities to be undertaken by each management or control function; again these responsibilities are represented as information objects (Panse 1990; Shorter 1990). An example of how to apply the CIM-OSA modelling framework can be found in Beeckman (1990).

At the heart of CIM-OSA is an event driven, process-based, enterprise modelling approach. Process-orientation has been found to be the most efficient integration principle of the manufacturing enterprise (Katzy et al. 1993). There are a number of important features that CIM-OSA does not address:-

1. Dynamic evolution - Enterprise processes and information are very dynamic in nature (CERC 1992). Hence, the needed integrated information model must show the necessary changes on a continuing basis. This dynamic evolution is not addressed in CIM-OSA. The CIM-OSA function view combines both the function and process view, and is suitable for business processes that have a pre-defined sequence of execution, but not for processes that undergo continuous revision. This is due to the fact that the emphasis of CIM-OSA is on automated

manufacturing rather than processes of design and product introduction.

2. Representation of goals and their relationships - What tasks are required to achieve the business goals and how the tasks are carried out (sub-processes, tasks, activity) are addressed, but representation of the relationships between the business goals at various levels are not addressed. Vernadat (1993) points out that one of the goals of CIM-OSA is to use the enterprise model to control and monitor the execution of business processes and enterprise operations. This necessitates an explicit representation of the objectives of the enterprise, relationships between the objectives that are broken down at various levels of the enterprise, relationships between the objectives and the process that aims to achieve them, and feedback information loops. CIM-OSA lacks representations of such essential relationships.
3. Representation of human organisation - Successful process representation requires identification of some key elements, such as interactions among people or decisions made during the design, as a primitive. In CIM-OSA there is no discussion on the human relations and human organisation details, which are essential when the enterprises are using teamworking.
4. Relationships among views - The interrelationships between the views are essential when CE technique is to be used, but, this is not addressed in CIM-OSA.

3.3.2 Architecture of Integrated Information Systems

The Architecture of Integrated Information Systems (ARIS) is a simplified derivative from CIM-OSA for CIM information systems. The architecture for integrated information systems constitutes a framework in which integrated application systems can be developed. Process chains are considered to be an important support for information systems. Elements of the process chains are the individual processes, and

manufacturing rather than processes of design and product introduction.

2. Representation of goals and their relationships - What tasks are required to achieve the business goals and how the tasks are carried out (sub-processes, tasks, activity) are addressed, but representation of the relationships between the business goals at various levels are not addressed. Vernadat (1993) points out that one of the goals of CIM-OSA is to use the enterprise model to control and monitor the execution of business processes and enterprise operations. This necessitates an explicit representation of the objectives of the enterprise, relationships between the objectives that are broken down at various levels of the enterprise, relationships between the objectives and the process that aims to achieve them, and feedback information loops. CIM-OSA lacks representations of such essential relationships.
3. Representation of human organisation - Successful process representation requires identification of some key elements, such as interactions among people or decisions made during the design, as a primitive. In CIM-OSA there is no discussion on the human relations and human organisation details, which are essential when the enterprises are using teamworking.
4. Relationships among views - The interrelationships between the views are essential when CE technique is to be used, but, this is not addressed in CIM-OSA.

3.3.2 Architecture of Integrated Information Systems

The Architecture of Integrated Information Systems (ARIS) is a simplified derivative from CIM-OSA for CIM information systems. The architecture for integrated information systems constitutes a framework in which integrated application systems can be developed. Process chains are considered to be an important support for information systems. Elements of the process chains are the individual processes, and

manufacturing rather than processes of design and product introduction.

2. Representation of goals and their relationships - What tasks are required to achieve the business goals and how the tasks are carried out (sub-processes, tasks, activity) are addressed, but representation of the relationships between the business goals at various levels are not addressed. Vernadat (1993) points out that one of the goals of CIM-OSA is to use the enterprise model to control and monitor the execution of business processes and enterprise operations. This necessitates an explicit representation of the objectives of the enterprise, relationships between the objectives that are broken down at various levels of the enterprise, relationships between the objectives and the process that aims to achieve them, and feedback information loops. CIM-OSA lacks representations of such essential relationships.
3. Representation of human organisation - Successful process representation requires identification of some key elements, such as interactions among people or decisions made during the design, as a primitive. In CIM-OSA there is no discussion on the human relations and human organisation details, which are essential when the enterprises are using teamworking.
4. Relationships among views - The interrelationships between the views are essential when CE technique is to be used, but, this is not addressed in CIM-OSA.

3.3.2 Architecture of Integrated Information Systems

The Architecture of Integrated Information Systems (ARIS) is a simplified derivative from CIM-OSA for CIM information systems. The architecture for integrated information systems constitutes a framework in which integrated application systems can be developed. Process chains are considered to be an important support for information systems. Elements of the process chains are the individual processes, and

a process is an occurrence of some duration which is started by an event and completed by an event. In contrast to a process, an event requires neither time nor resources. The components of an information system in ARIS are processes, events and conditions; the factors of production material, human labour (employees) and equipment (production equipment and information technology equipment); and organisational units which provide their structure, and relationships among these.

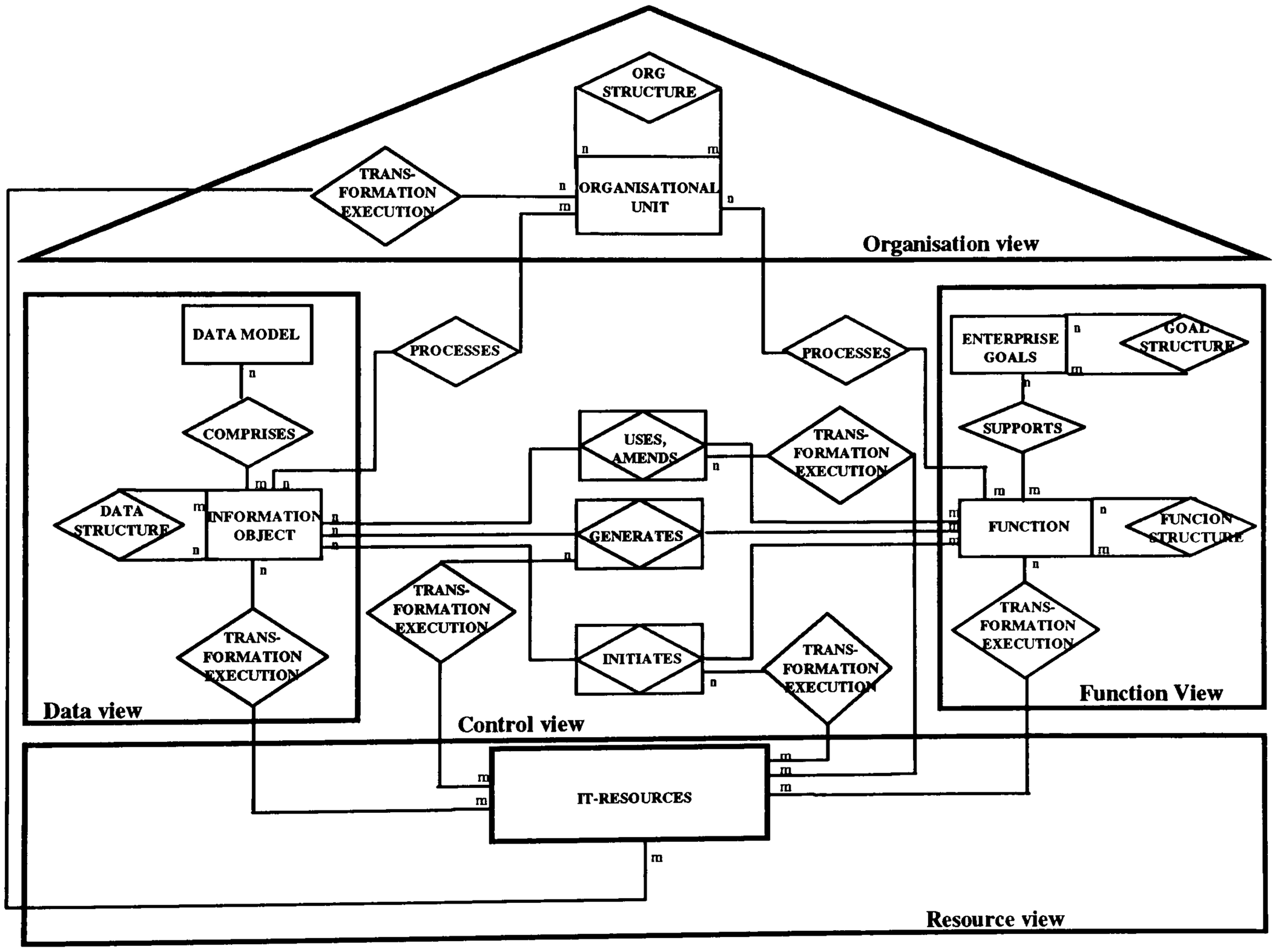


Figure 3.2 Information model of the ARIS architecture (Sheer 1993)

A process-specific view gives rise to redundancies in data representation and, hence, the process chain model is broken down into data view, resource view, function view and organisation view to reduce redundancies (Scheer 1993 and 1994). In order to record the relationships between the view points, an additional view called 'control' is introduced (Figure 3.2). The ARIS architecture clearly shows that there is a need for

a process is an occurrence of some duration which is started by an event and completed by an event. In contrast to a process, an event requires neither time nor resources. The components of an information system in ARIS are processes, events and conditions; the factors of production material, human labour (employees) and equipment (production equipment and information technology equipment); and organisational units which provide their structure, and relationships among these.

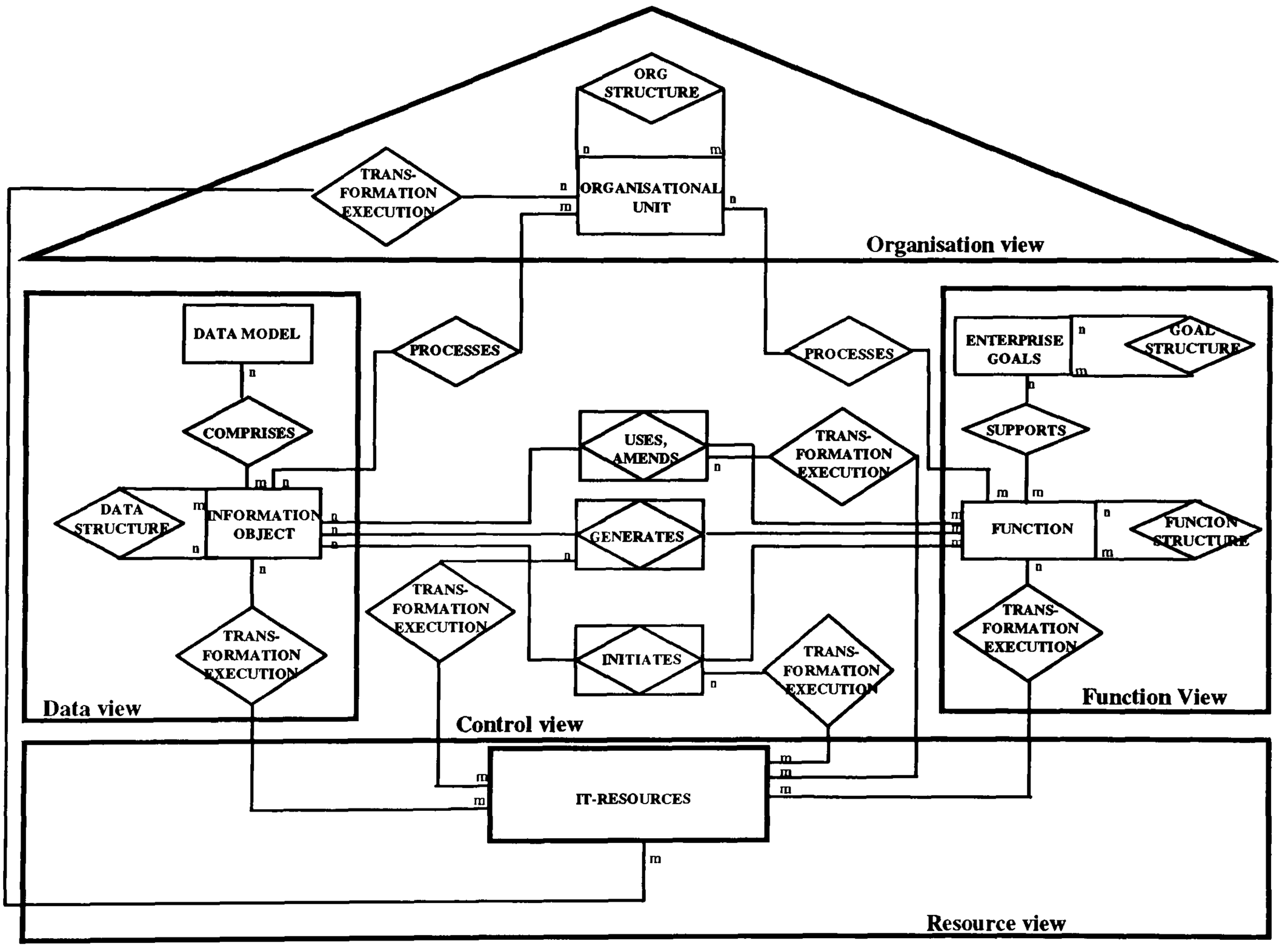


Figure 3.2 Information model of the ARIS architecture (Sheer 1993)

A process-specific view gives rise to redundancies in data representation and, hence, the process chain model is broken down into data view, resource view, function view and organisation view to reduce redundancies (Scheer 1993 and 1994). In order to record the relationships between the view points, an additional view called 'control' is introduced (Figure 3.2). The ARIS architecture clearly shows that there is a need for

integrating data models with the function and organisation models. But it does not address how to represent the link between the data model and the function and organisation models. It is important to note that the data model, data structure and the information object are all included in a single view called data view. When the data involved in the system is distributed and heterogeneous, the design of the conceptual models of the data and their integration is complicated, and it is necessary to segregate the data modelling view from the data view. Thus, the representation of the data view in such large models is not addressed in detail in ARIS to provide the necessary integration infrastructure.

3.3.3 Systematic Concurrent design of Products, Equipments and control Systems

The crucial integration of engineering and production knowledge is the objective behind the Systematic Concurrent design of Products, Equipments and control Systems (SCOPES) project. A system framework that stores and provides a company's knowledge of products, processes, equipment and shopfloor control strategies to the product development team was developed. SCOPES solution consists of the following modules (Figure 3.3):

- off-line modules that operate in a CAD environment comprising of : Product design, Assembly Planning, Resource Planning and Simulation
- Online supervision modules comprising of: control and monitoring modules
- scheduling and flow control modules, which are common to both off-line and online activities.

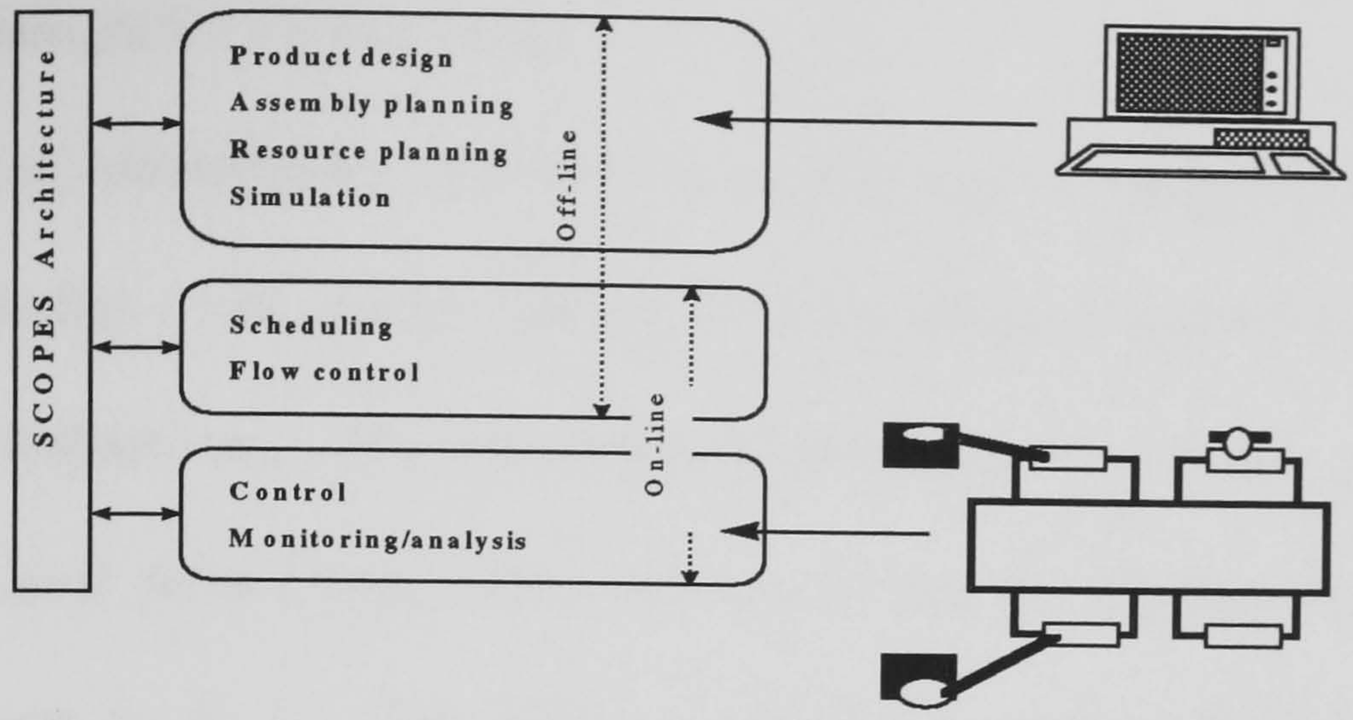


Figure 3.3 The SCOPES modules (Wallace 1995)

A system architecture that would enable the bi-directional communication between modules has been developed. It consists of 5 layers: knowledge, design, simulation, supervision and execution, each containing a circuit loop model that expresses the successive activities within each level that will enable concurrent engineering to occur. The communication paths within a loop and vertical communication between the modules on different levels are included in the model. The databases involved include database for assembly planning information, and a knowledge base dedicated to the resources. SCOPES solution aims to improve the communication between design and manufacturing members of product development teams, and facilitate the introduction of CE methodology. It provides design support on the downstream functions associated with the assembly of a product throughout the design process (Wallace 1995). In order to manage the overall process, it has to be linked with the upstream functions of the product development process. The model is more geared towards scheduling and flow control of the design, and simulation of an assembly workshop than information sharing. The human resources, the information used in the design process, how and in what format the information is communicated are not represented. Concurrent engineering must be based on relating activities (Hein 1994), and SCOPES does not discuss about the relationships among the activities in detail.

3.3.4 GRAI Integrated Methodology

GRAI Integrated Methodology (GIM) is a decision tool resulting from production management studies. The model consists of three abstraction levels - conceptual, structural and realisational. The conceptual level defines what to do, structural or organisational level defines who, when and where, and the physical or realisational level defines how to do it. The GRAI model for concurrent engineering system consists of a physical system and a control system (Figure 3.4). The physical system transforms the physical flow of raw material, components in manufacturing. The physical flow is controlled by the control system to reach the objectives given by the management of the company. The control system is split further into two subsystems : the decision system and the information system. The information system is the link between the decision system and the physical system on one hand, and on the other hand between the environment and the manufacturing system (Doumeingts et al. 1993 and 1994).

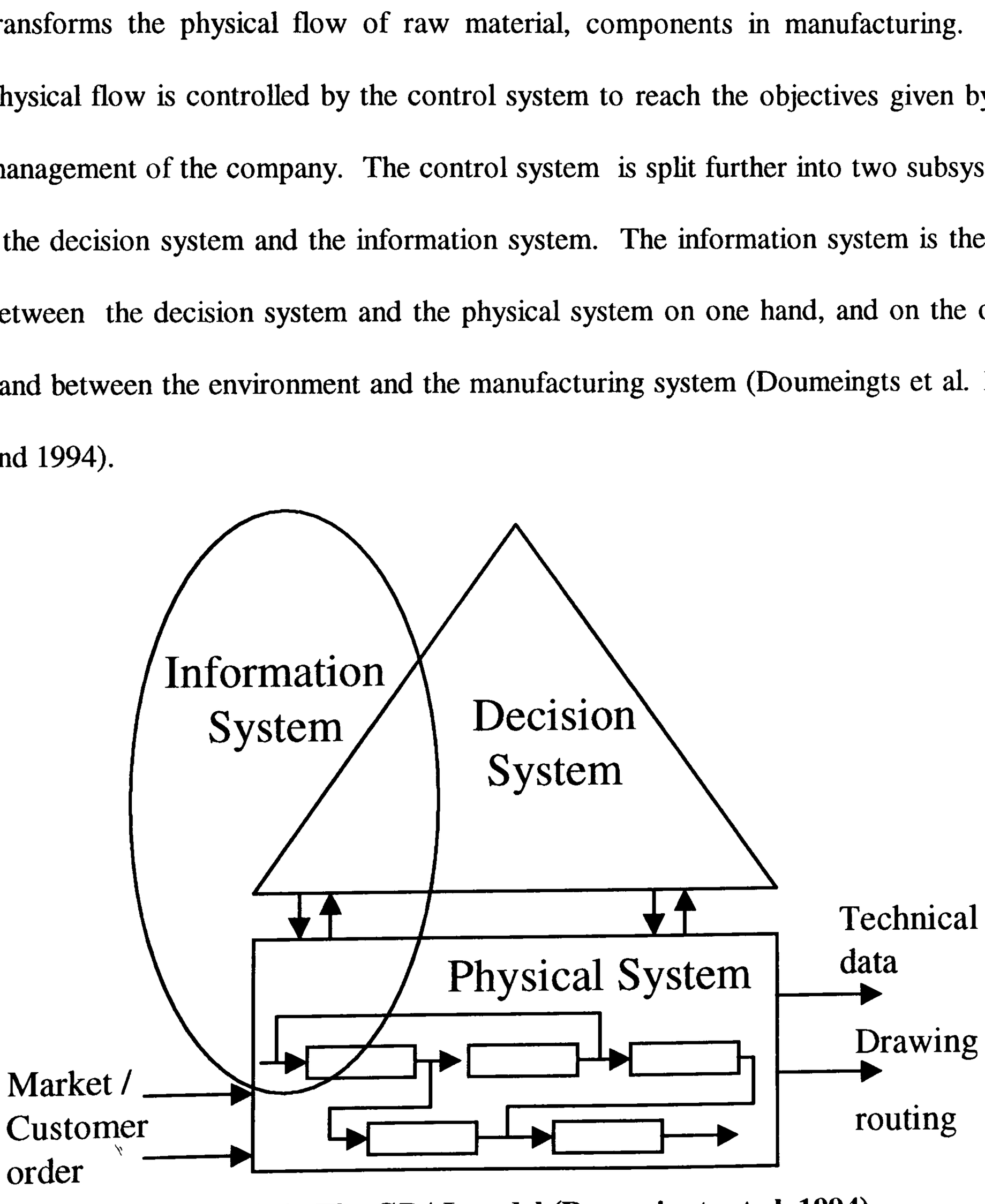


Figure 3.4 The GRAI model (Doumeingts et al. 1994)

When applying the GRAI method, a graphical tool called GRAI grid is used. It provides a hierarchical representation of the whole structure of the decision centres. Figure 3.5 shows the structure of a GRAI grid of an engineering activity which transforms information. The basic elements of the function "to manage information" are information (I) and time (T). For concurrent engineering domain, it is necessary to manage the physical flow i.e. information available in time, and the resources i.e. capacity in time (Doumeingts et al. 1994).

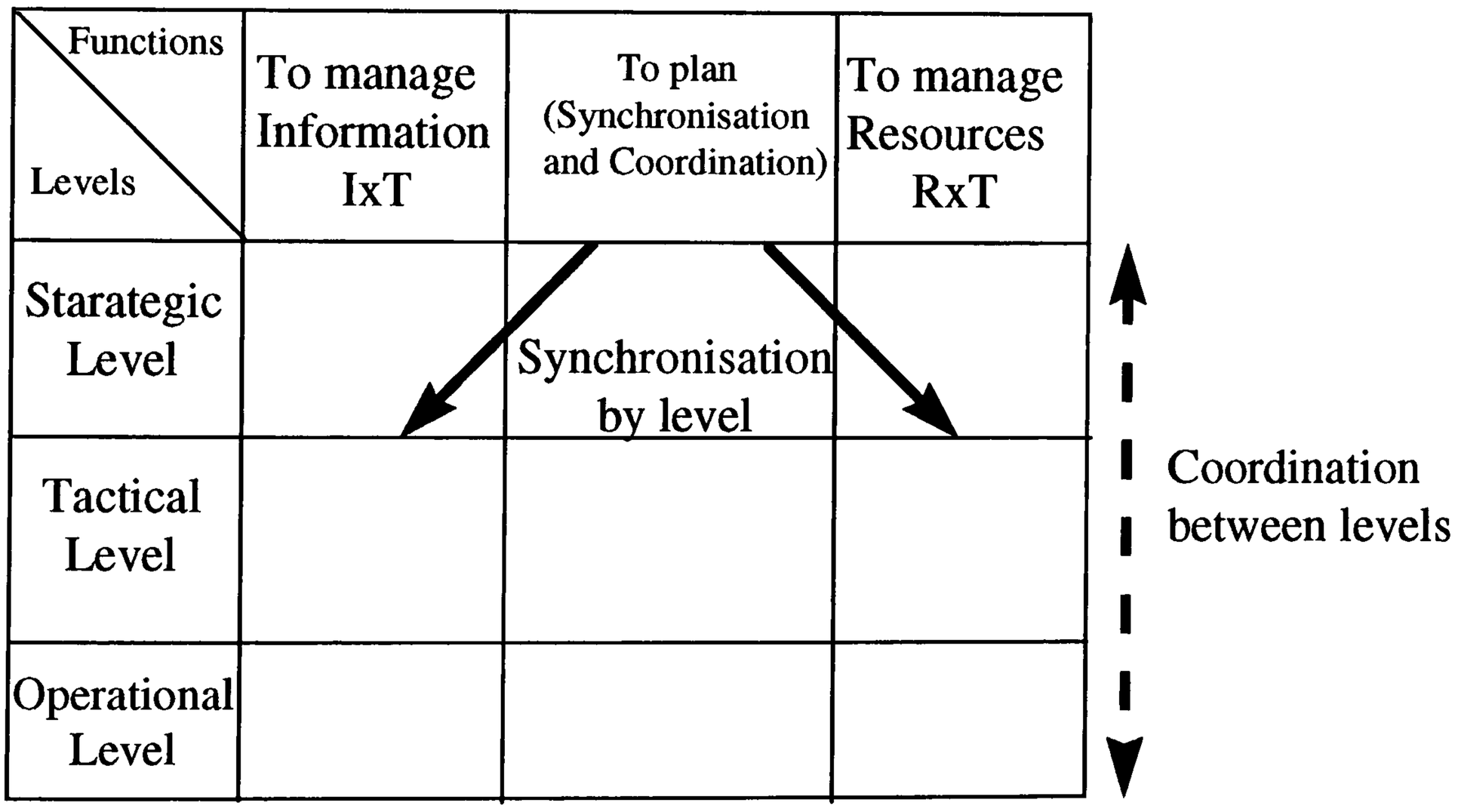


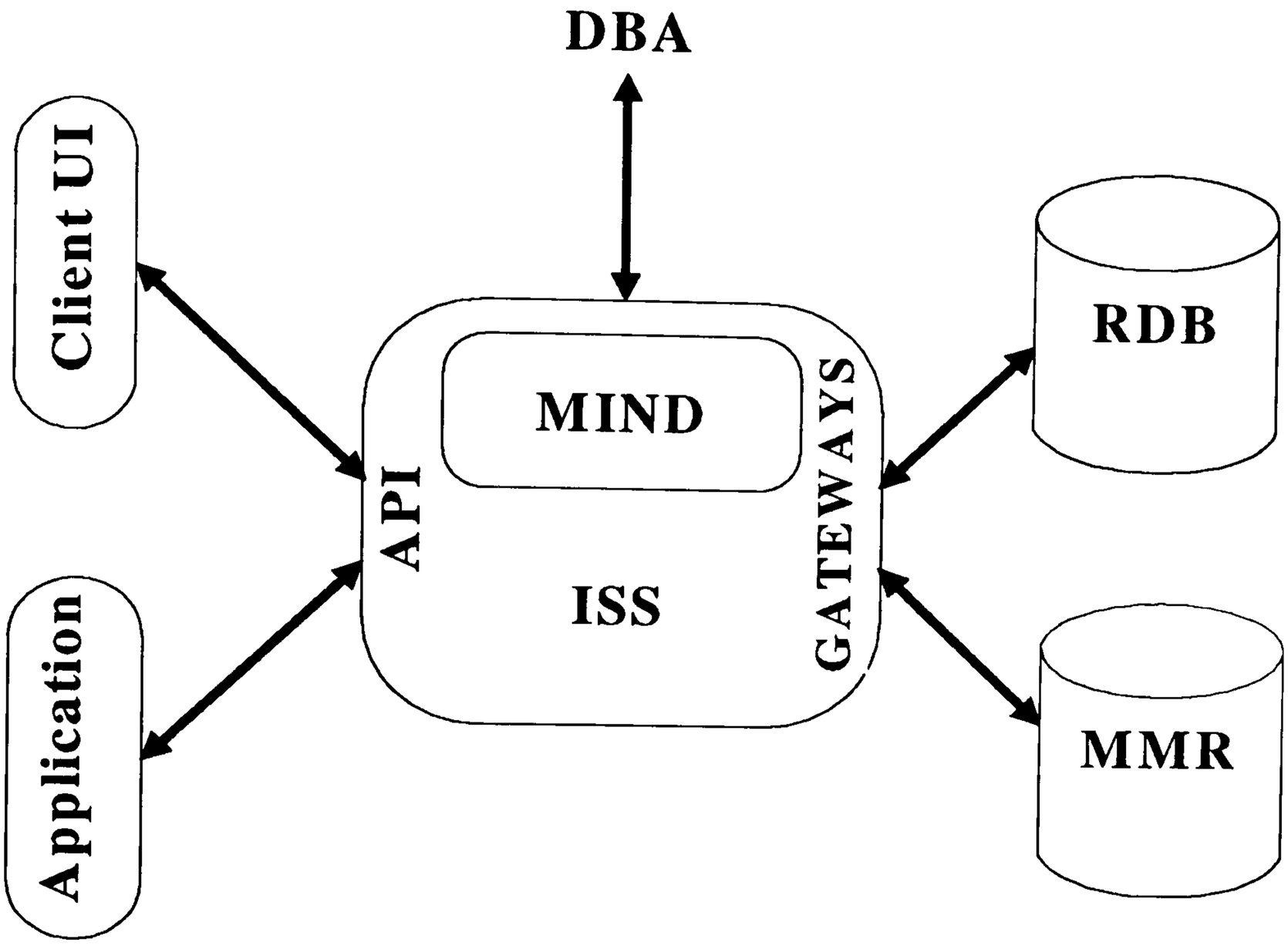
Figure 3.5 Basic GRAI Grid (Doumeingts et al. 1994)

Information must be aggregated in accordance with the decision making level. The higher levels (preliminary levels) are in charge of controlling the overall process. It corresponds to the long term management, the level of detail is low (aggregated information), and the time scale is large; the decisions are linked to the strategic decisions of the companies and is concerned with the nature of the product, the

research of the new domain etc. The middle level concerns the decision about human or technical investment. The lower levels deal with controlling a specific part of the process; the level of detail is more important, and the time scale is reduced and corresponds to short term management (Doumeingts et al. 1994). As GRAI method is a tool for analysing the decision making processes, there is no integration structure defined. Computer-based implementation concepts are not developed to implement and use this tool.

3.3.5 Information Sharing System

Information Sharing System (ISS) is part of a set of projects in concurrent engineering developed as part of the DARPA (Defence Advanced Research Projects Agency) Initiative in Concurrent Engineering (DICE) program. DICE started in July 1988, and views computer support for CE along five broad areas: sharing information, co-locating people and programs, integration of tools and services with frameworks, co-ordinating the team, and capturing corporate history. The functional requirements of these five areas are discussed by Jagannathan et al. (1991). The primary goal of CE technology is to enable a virtual tiger team to share information and information models (Almasi et al. 1992). The view of achieving enterprise integration through a facility of information sharing among the team members provides a model-directed view of an enterprise (Karinthi et al. 1992; Jagannathan et al. 1993).



ISS : Information Sharing System
API : Application Programmers Interface
MIND : Model-based INformation Directory
DBA : Database Administrator
RDB : Relational Database
MMR : Multi-Media Repository

Figure 3.6 Overview of the information sharing system (Karinthi et al. 1992)

The primary goal of the information sharing system (ISS) is to provide the means for a team member to transparently access information over the entire enterprise. Central to ISS (Figure 3.6) is a module called the Model-based Information Directory (MIND), which contains a model of the information which is distributed throughout the various repositories such as databases, documents, drawings and data files in the enterprise. ISS provides access to the model to enable a user to view how the information is organised. MIND maintains a mapping from the model to the actual information in each of the repositories. Using the mapping, ISS can assist a user in accessing the information which has been modelled. An example that defines the information needed

for a coffee cup designer and manufacturing engineer to communicate information about the geometry, material and colour of a particular type of coffee cup can be found in Karinthi et al. (1992) and Almasi et al. (1992). The schemata of the databases involved in the example are expressed using EXPRESS. A prototype system has been built integrating ORACLE relational database and multimedia repository.

During the design and development of ISS, it is mentioned that one of the characteristics of the engineering data modelled is that the schema does not change frequently (Karinthi et al., 1992), and it is assumed that the schema will remain static during the time that users will be accessing information from the ISS. Integration with different kinds of database management systems (other than ORACLE) is not addressed. The issue of how to retrieve information when multiple mappings are there is also not addressed in the model. Even though it is mentioned that process model forms a part of the overall enterprise model, nothing is mentioned about how the information is attached with the process, what information is used and in what format the associated information is structured and accessed. In other words, the link between the process model and the information sharing model is not attempted, nor any link with the management of the process.

3.4 DISCUSSION

It would be impractical to require a product introduction information model to be fully defined before implementation. Hence dynamic modification or extension is an essential feature of the information model for the PI process. Most data models assume static class definitions, and allow only instances to be defined at runtime.

Dynamic capabilities cannot be simply appended onto a model initially defined statically. The implications of dynamic change become apparent in the formal definition of a data model, in relation to its axiomatic properties. These must explicitly deal with extension and modification. This has been emphasised by Eastman C.M. and Fereshetian (1994) during their study on information models for the design process. In order to implement an information model for PI process, it would be necessary ① to provide an ability to dynamically evolve the schema at runtime and ② to have a database management system that allows on-line schema changes while applications are running.

It is clear from the literature that there is a lack of information models for the product introduction process. The model-directed view of ISS can be applied, but, the schema in it is assumed to be static. Meta-modelling method used in the software engineering field is good in integrating data models, and a model with a meta knowledge has been identified for designing integrated information models that will aid in building information systems that can be dynamic, co-operative and distributed. But how to use the meta-modelling technique for proving a dynamic information model that represents the information relevant to the product introduction process is not addressed in the literature, and it has to be explored.

It can be observed from CIM-OSA, GIM and ISS projects that integration can be achieved through a facility of information sharing among team members. Thus, information could be the glue for integrating the processes that are carried out in different domains of an enterprise and/or across enterprises. Mastering the information flow allows a better integration of the product life cycle (Beeckman 1989), but data

representation techniques for the information flow among the activities are missing. This would necessitate an analysis of the representation of the organisational influence, or how, where (in which process) and what information is generated and how, where, what and to whom information has to be sent across organisational boundaries. The association between activities that reuse the information and data, and the role the data plays with respect to the activity are very essential when integration is to be achieved, business processes are to be reengineered, or concurrent engineering is to be practised. However, the existing models do not address these concepts, and also the flow of information within the enterprise.

An optimal use of the resources such as people, computers and machines requires they are part of a system able to master the action flow and the information flow. Mastering action and information flow improves the quality of the products, due to avoidance of errors and increased production flexibility (Beeckman 1989). The quality of a product depends on the functionalities that it provides. Even though achieving product functionality is an essential goal and is the key to the success of a new product, there is no model in the related areas that considers product functionality and its relationship with the process that introduces the product; the semantic link (goals, control) between the product functionality and process have been vastly ignored. Modelling methodologies should tackle this problem of modelling product functionality in order to control the PI process effectively.

Process-based approaches are gaining importance in integrated information modelling, process modelling (Blessing 1994) and enterprise modelling (Vernadat 1993; Dutton 1993). As it can be seen from CIM-OSA and ARIS, the concept of *modelling views* to

deal with system complexity, providing 'windows' to focus on some enterprise aspects while disregarding others is becoming popular. For effective management of the PI process, it is necessary to have an integrated information architecture that represents the process, product functionality, PI resources and product. It is pointed out in CERC (1992) report that process and output integration is a very complex and highly detailed endeavour. The reference architecture which describes this endeavour is also complex and highly detailed.

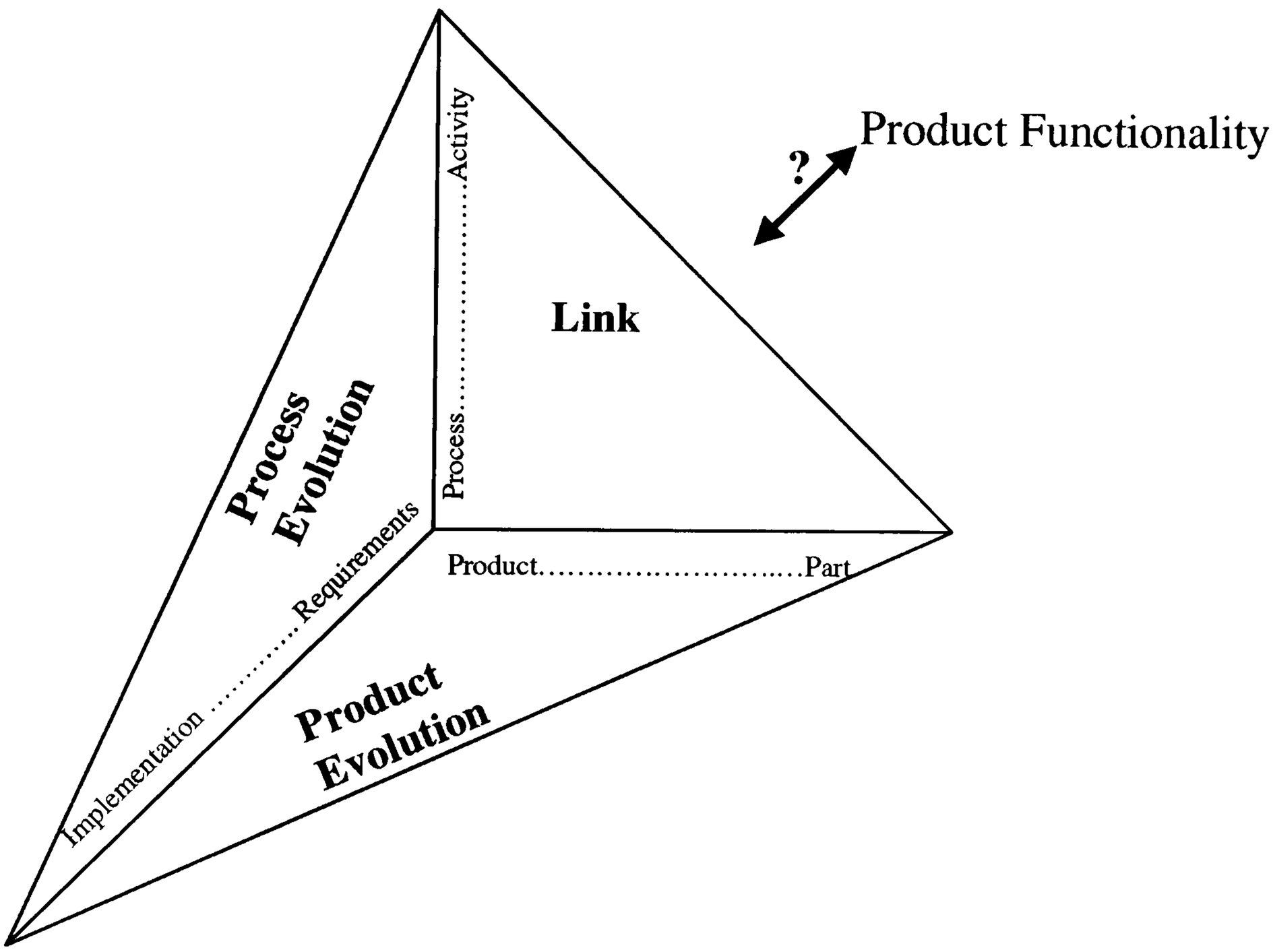


Figure 3.7 Dimensions of the product introduction process

None of the candidate architectures and associated methodologies are, as yet, completely developed, described and documented to address ① the goal (product functionalities) view of the process, ② the link between the process and its goal, ③ evolution of information and ④ the link between the process and its output (Figure 3.7). This thesis tries to study, analyse these necessary links in order to develop an integrated information model for the effective management of the product introduction

process. The information basis to store all relevant results of the PI process is the product model. Hence, it would be necessary to explore the product models before investigating the relationships between the PI process and the product model.

CHAPTER 4

4. PRODUCT MODELS

4.1 INTRODUCTION

The product introduction process generates and uses information about the output of the product introduction project i.e. the product. This information includes product functionality, product structure, product data (i.e. instances of the structure), and production methods such as assembly and manufacturing methods. The distinguishing characteristics of the product introduction process and the product information that it instantiates call for special issues in modelling. The product model provides the basis for storing all relevant results of the product introduction process, and is the focus of this chapter.

First, the role of product modelling in the context of product development is discussed, and different approaches to product modelling and types of product models are presented from the literature. The existing models are compared based on the complex characteristics of product data. The relationship between process chain and product model is discussed, and it is argued that the existing product models do not address the issue of how to link the process that generates the product data with that data.

4.2 PRODUCT MODELLING

Traditionally, results from technical design were captured in the forms of drawings on papers. Computer-aided design was introduced around 1960, and since then, a large variety of geometry-based approaches have been developed. During the eighties, new

techniques like knowledge processing, computer simulation were used to enhance the capabilities of geometry-based approaches (Figure 4.1). The demand for integrated, instead of isolated, product development approach during this period led to the concepts of product model and process chain (Krause et al. 1993). Besides

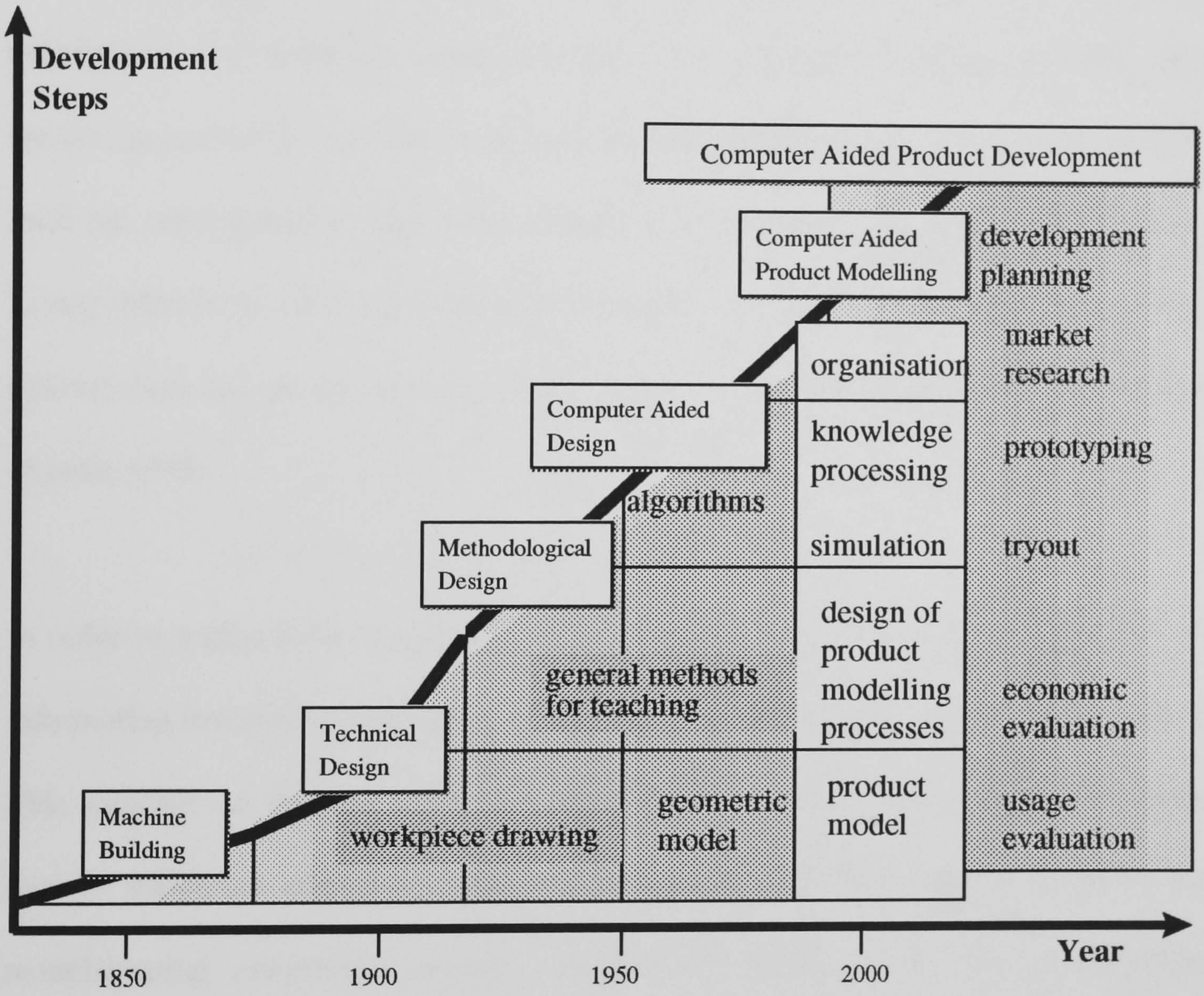


Figure 4.1 Product modelling in the evolution of product development

(Krause et al. 1993)

linking different product development tasks by directly connecting softwares, standard data formats and common databases have come into use. The term 'product model' first gained acceptance in the context of international efforts to develop a neutral format (for the transfer of product data between computer-aided systems) which is capable of representing product data for all sectors of engineering as well as all aspects

of the life cycle (Bloor et al. 1991). However, a unique and consistent representation as well as data management without too much redundancy is not supported to date.

Products in a broad sense serve a function in order to fulfil a need. This function is delivered by a geometrical solution, which is developed with the help of physical calculations and economic considerations. The geometry is then produced using specific processes in manufacturing systems. Many geometrical solutions may exist to fulfil the same function, each being made through different production processes and having different economical outcomes. Products that fulfil the function, have shortest delivery time and lowest cost etc. are the ones that should be selected for manufacture (Bjorke 1989).

In order to realise total integration, an ideal approach is to integrate all aspects of the information involved in producing a product into a single shared information model. This concept of sharing a product information model among different functional groups within an enterprise prompted new directions of research in design and manufacturing integration, namely: Product Modelling. The product modelling concept aims to achieve the integration and sharing of data/information among different engineering activities in an enterprise by using a model which can represent, transmit, manipulate and store all the technological information necessary for the design and manufacturing activities related to the product.

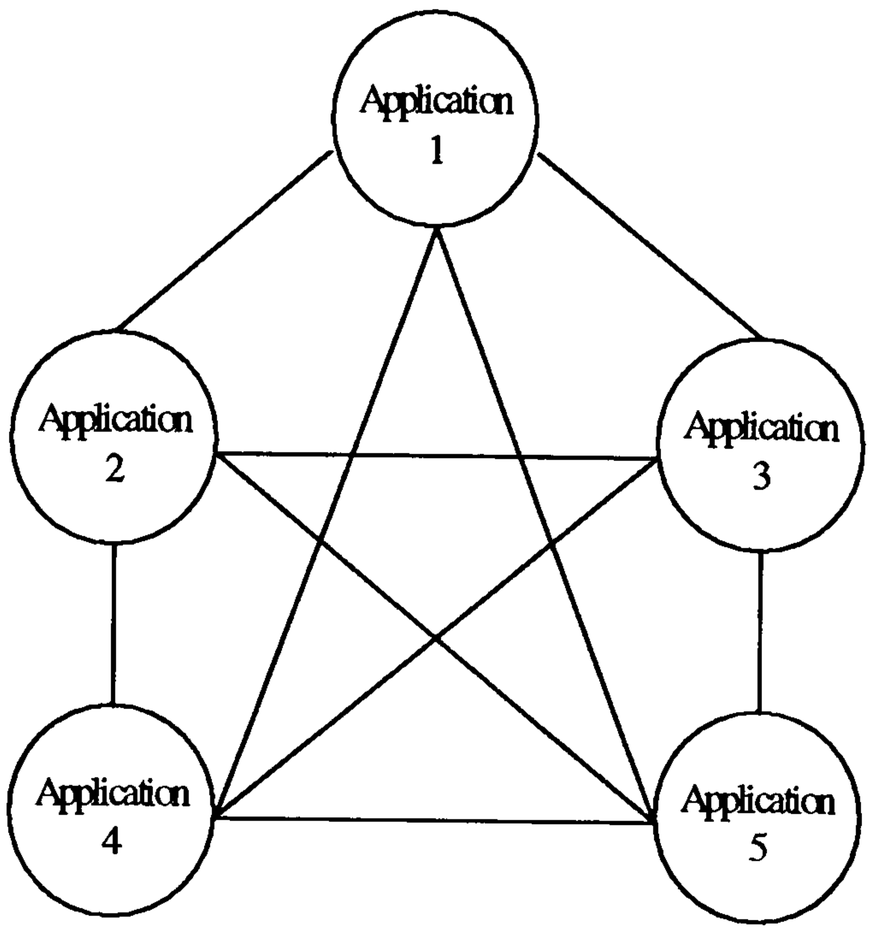


Figure 4.2 Data exchange

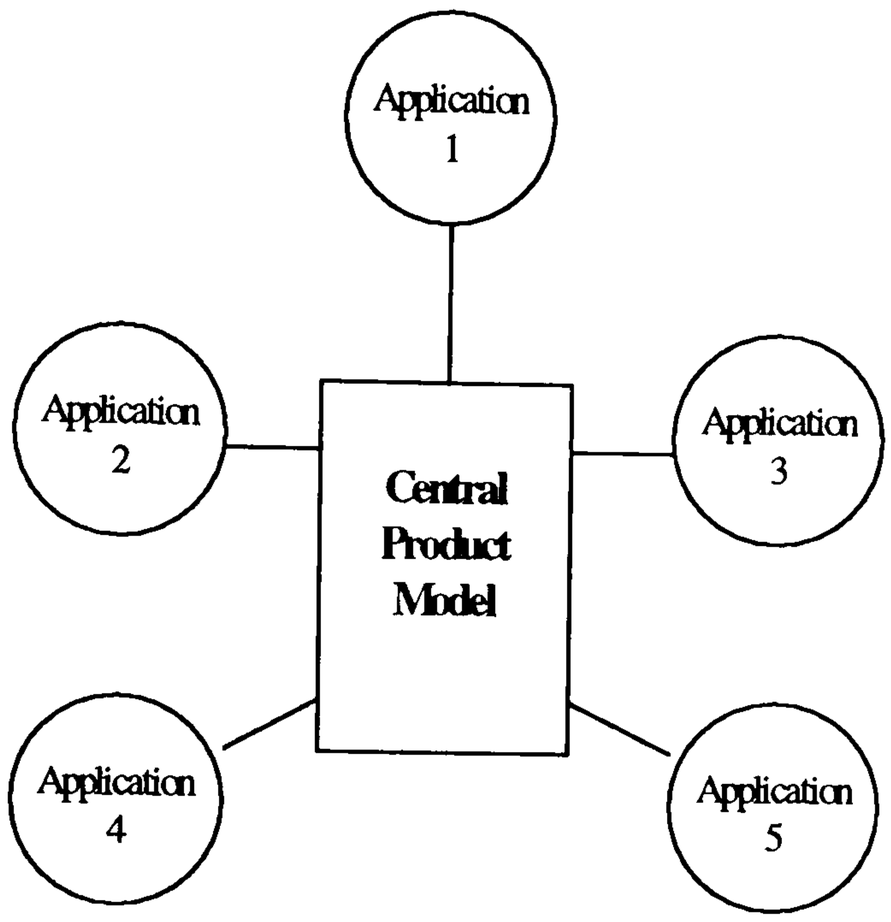


Figure 4.3 Product model

Traditional approach for integration was through data exchange methods (Figure 4.2) whereas the product model approach is an integrated approach to document all relevant data about a product within a single framework (Figure 4.3). It involves, in principle, everything from geometry and topology descriptions to production and maintenance information, combined into a single, consistent logical schema. This is to ensure the conceptual integration of the product data and to enable a shared "understanding" in heterogeneous environments (Hordvik and Oehlmann 1992). Thus, product modelling generates an information reservoir of complete product data to support various activities at different product development phases (Krause et al. 1993; Erens et al. 1994; Chapman 1996). A product model is also known as product data model (Shaw et al. 1989) and product defining model (Andreasen et al. 1996).

4.2.1 Product models based on method of formulation

Product models are categorised according to the method of formulation and their information content. There are two stand-points in the formulation of product data models: general and specific. A general (or universal) product model schema is valid for representation of the semantics of different types of products. The essence of the *general* is "hiding" information on the actual difference between two objects from the computer, and the model schema may not clearly reflect all information. This requires more participation of the user during modelling, since the undefined information has to be perceived by the user. The general product model schema must start with a view that reflects the general similarity among any product structures. It is based on the philosophy that everything can be viewed by "relationships" between sets of "primitive objects (entities)" (Figure 4.4). For example, 'component_of' and 'relative geometrical position' are relationships among parts and/or assemblies. There is no definite one-to-one correspondence between the data modelling method and the result (product data model). Using one data modelling method for the same product semantics may produce different results.

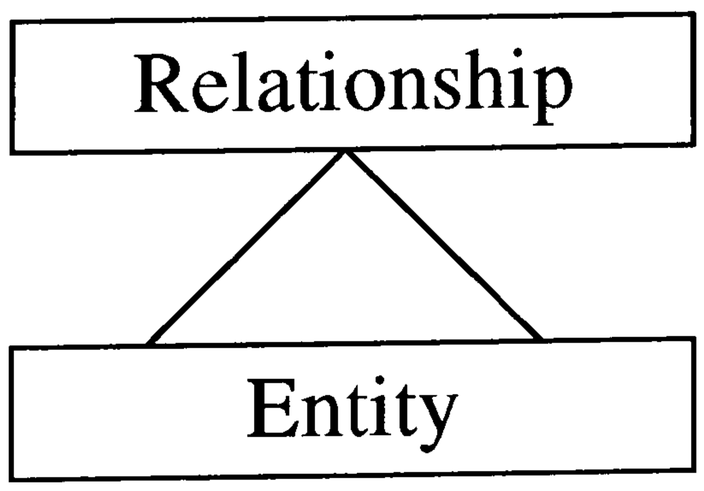


Figure 4.4 Model with two levels (Zhang et al. 1993)

Specific product models express the semantics of a product in a more transparent way. Product semantics is the discipline dealing with the relationship between features and

the product to which these features refer. For instance, an aeroengine blade has features: height of the blade, leading edge radius, stress, etc. To express the semantics of a product in a transparent way, product features should correspond to columns of a table representing the product in certain aspects.

Extendibility of a product data model is a property that describes the ease with which the product model may be adapted to changes of product semantics. It is a way to achieve flexibility of the model. To make a product data model highly flexible, the product semantics are represented by table contents, which can be achieved by "turning" the table 90° (Figure 4.5) (Zhang et al. 1993).

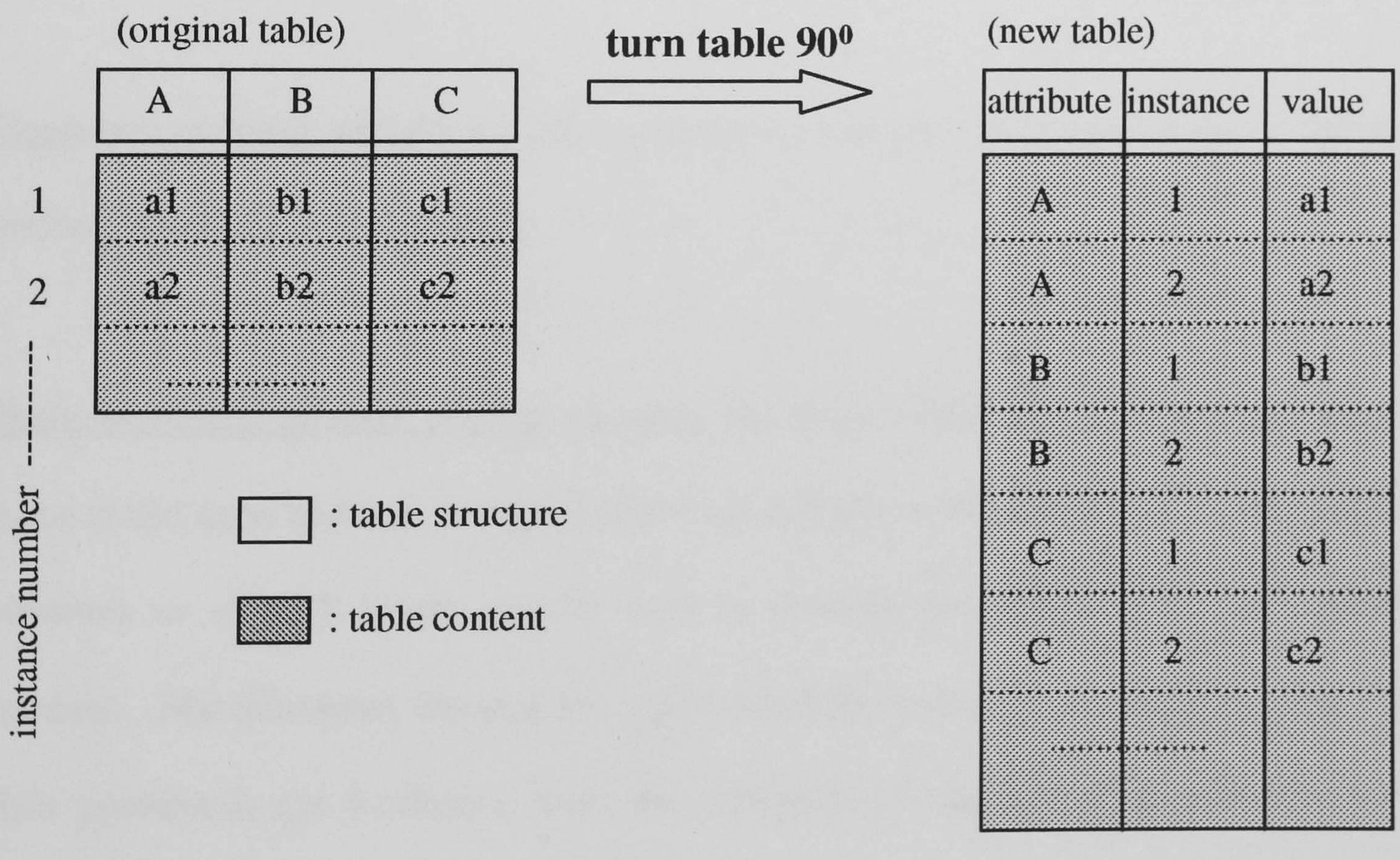


Figure 4.5 Rule for achieving a highly flexible product data model (turn table 90^0) (Zhang et al. 1993)

4.2.2 Product models based on their information content

To cope with large-scale and complicated products, such as automobiles, ships and aeroengines, it is necessary to integrate different kinds of product related information and to have a structured representation for them. Based on the product related information represented in the product model, the product models are categorised into: structure-oriented, geometry-oriented, feature-oriented, knowledge-based and integrated product models.

Structure-oriented product models are based on the product structure, as a description of the product's breakdown. For example, bill-of-materials can be used to represent the structure of a product.

Geometry-oriented product models are designed to represent geometry (wire frame, surface, solid and hybrid models).

Feature-oriented product models represent the shape patterns as coherent geometric items called form features. Design features are features in terms of primitive geometric elements or symbols which can be used to describe the functions of the desired product. Manufacturing features are constructed by combining shapes depending on their position in the workpiece from the viewpoint of manufacturing, assembly and inspection. Function oriented modellers is one approach of feature-based modelling that allows the designer to create mechanical part and product geometry by composing high-level functional features which contain dimension, location parameters and non-geometric attributes related to product's function, manufacture, engineering analysis, etc. Functional design features embody the designer's intent and knowledge and

express the designer's reasoning regarding the relationship with other features and functional characteristics. In such models, the features library include the functional features such as holes, chamfers and fillets; and primitive features such as cylinders, blocks, cones and spheres (ElMaraghy et al. 1993).

Knowledge-based product models are characterised by the use of artificial intelligence techniques like object-oriented programming and rule-based reasoning (Krause et al. 1993). These models have the ability to build abstract taxonomies of products or processes as objects and to store knowledge about previous designs, possible alternative parts in an assembly or the abilities and validity of production processes used for a specific class of products.

Integrated product models cover the abilities of geometry-, feature-, structure- as well as knowledge-oriented models. Many different types of product information can be stored in an integrated product model along with generic product knowledge. Generic product knowledge includes the product history, development principles, models of customer and technological requirements, and failure models.

4.3 PROCESS CHAINS AND PRODUCT MODEL

A process chain is a sequence of tasks that are connected to perform the product development processes. Process chains that refer to product development workflows or product modelling processes represent the product modelling processes consisting of a set of technical and management functions required to transfer initial ideas to final products (Krause et al. 1993). Process chains and product models are interrelated aspects of product modelling. The components of these aspects are shown in Figure

4.6. One of the most important results from process chains is the product model data. The information technology that is useful for supporting process chains should have the capability to record intermediate and final results as the product model data.

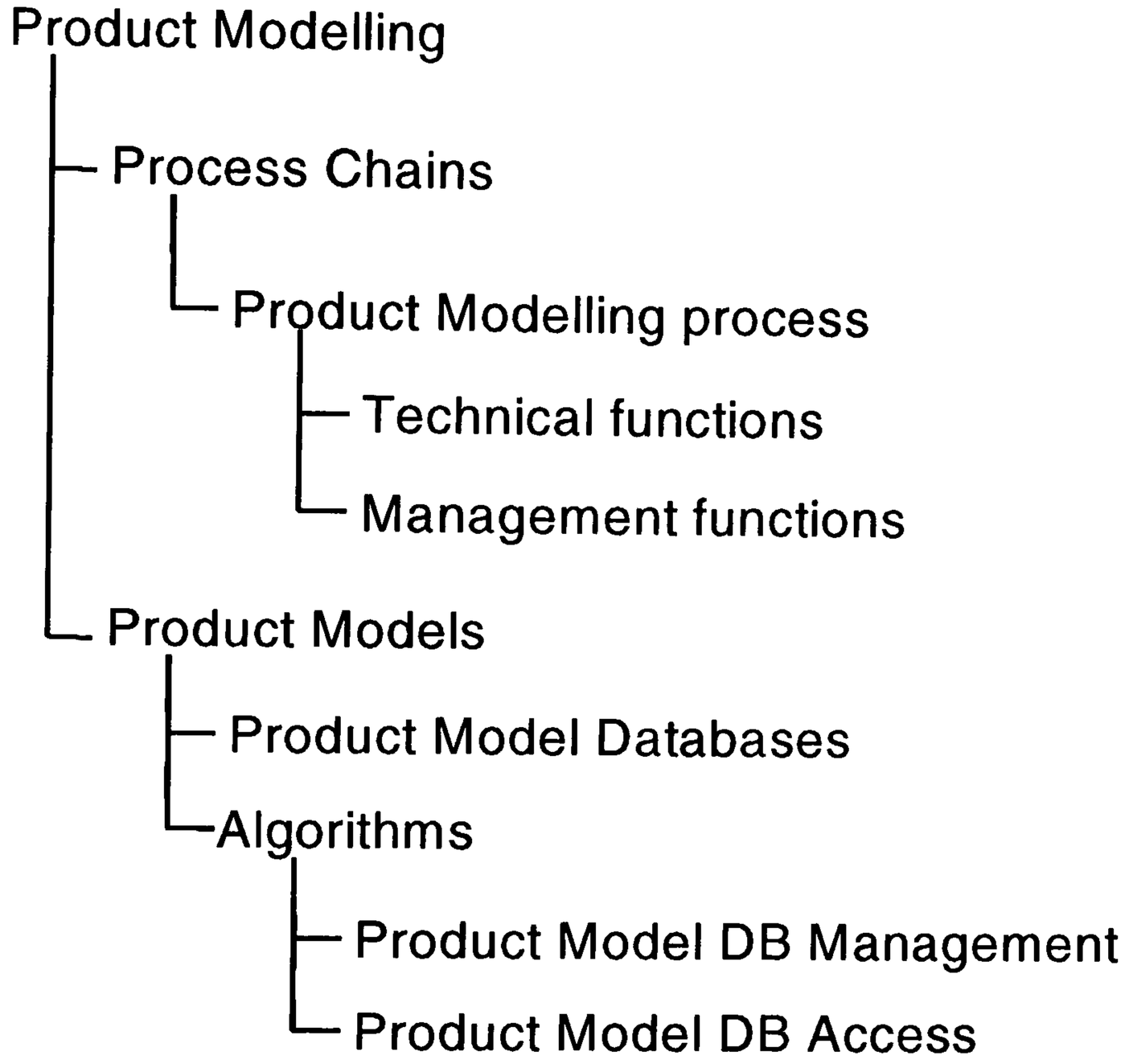


Figure 4.6 Product modelling aspects

Modelling of a product structure has to be considered from various aspects such as structural dependency, functional dependency and geometrical position. Different aspects may produce totally different abstractions of the product structure. A product model can be structured into sub-models, creating so called partial models which contain information of specific tasks from process chains. An integrated product model contains partial models such as: technology model containing material, tolerances and surface condition; planning model containing process plans, assembly plans and

inspection plans; design model containing function, assembly, part and shape (Krause et al. 1993).

Parallel process chains are the most advanced type of process chains. When concurrent engineering is applied in product development, parallel processes happen; these parallel processes need to be linked with the product model. The important requirements of concurrent engineering include parallel problem solving, team work in a distributed way, availability of adequate database/product models and project management systems (Figure 4.7) (Krause et al. 1993).

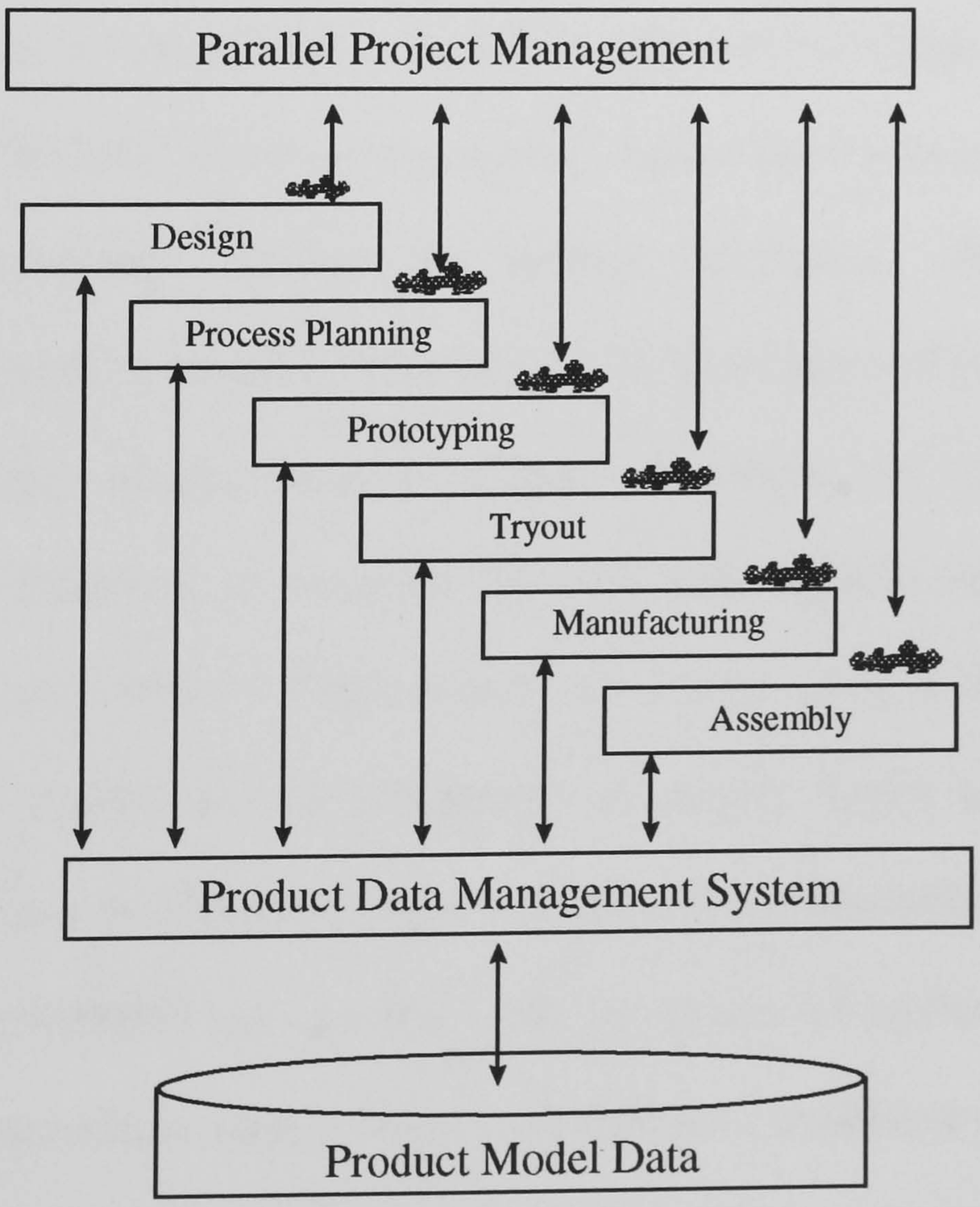


Figure 4.7 Parallel process chains with product model (Krause et al. 1993)

Concurrent engineering in a product introduction process relies upon the ability to share up-to-date engineering product structure data across multiple disciplines for the purposes of analysis, feedback and modification of product specifications. Thus the concurrent process iterates around the product structure, which is critical for enabling effective sharing of product introduction information among the team members involved. The product model is used and updated throughout the PI process cycle that is controlled by the parallel project management system.

The root of the problems in "data sharing" is differences in proprietary data representations i.e. vendor-specific data formats. The vendors of computer applications store the data that is required and produced by their systems in their own proprietary format, and this leads to "islands of automation". Industries have begun to address this problem by developing standards for product information. ISO (International Standards Organisation) standard "STandard for the Exchange of Product model data" (STEP) is a large, complex standard designed to support the needs of different industries and disciplines to exchange product data between different software applications (Fowler 1996). STEP models are written using a data specification language called EXPRESS. An EXPRESS information model is organised into schemata containing definitions of entities, attributes and relationships (Schenck and Wilson 1994). EXPRESS has only conceptual constructs for product modelling, and there is no implementation version or software facility to implement the concepts, nor does it point out how to connect the process chain and the product model. STEP is centered on the concept of "product", especially technical data, and its capabilities are focused on data exchange using files rather than data sharing and data integration, and are limited to static data sets.

4.3.1 Characteristics of product data

Apart from the geometry, the factors that contribute to the complexity of product data are evolution, multiple views, data interdependence, reuse of earlier data, size of information, and version control.

Evolution - The required product (structure) will become available gradually, during the process of engineering and manufacturing engineering. The evolutionary and iterative product introduction process generates the product data, and this data is about evolving, incomplete product design which is constantly changing throughout the product introduction project. There is a subsequent need to maintain potentially inconsistent data until the end of the process, when a single consistent product definition is achieved.

Multiple views - The PI process is multidisciplinary in which each discipline participating in the product introduction process has its own view of the data needed to define that product. The view of a particular discipline optimises the organisation and content of the product data from the perspective of the team members working in that discipline. For example, when designing an aeroengine blade, the view of a project manager emphasises the schedule and resource aspects of the process, while the view of the design engineer emphasises the aerodynamic aspects of the design, and the view of the materials engineer emphasises the stress aspects of the design. Thus, the product information is accessed by the processes, which have their own logical view of the object (information) representation. Therefore, the semantics of the information is known only within the processes that use/generate the information. In addition, these

views must optimise the product data model for use with the design tools used in each discipline. Combining the different semantics in a single framework is difficult.

Dependencies among different structures of a product - Modelling of a product structure has to be considered from various aspects. Different aspects may produce totally different abstractions of the product structure, which are superimposed in the final product. It is necessary to consider the interrelationships between the structures (like physical structure, functional structure, manufacturing structure, material structure of the product) when the processes that access the product data need to be concurrently performed. These interrelationships are very complex to model in a single framework and their representation calls for further analysis of the semantic relationships between these structures.

Reuse of earlier product data - An entire sub-assembly available from an earlier project may be attached to a node of the new product structure tree to record a design decision to use a previously designed sub-assembly. One may then modify a portion of it for the present project. Since industrial products, such as automobiles, may have only 10-15% new components from one model to the next, this re-use of earlier components, sub-assemblies and assemblies is the rule rather than the exception (Cleetus 1995). Thus, most products evolve with time. Reuse of previous product and process data may improve the effectiveness and efficiency of the product introduction process. It is important that they are recorded in such a way that they are easily accessible afterwards and that they can be easily manipulated (Van Veen 1992).

Size of information - In order to minimise the lead time for product development, industries use concurrent engineering techniques which leads to an overlap in engineering, manufacturing engineering, procurement and production activities. For that, product data, which is to some extent final from the engineering viewpoint, must be passed as soon as possible to other organisational functions such as purchasing and manufacturing engineering. It is very important to be able to define the minimum information that must be available to start a particular activity. For actual planning, it is important to be able to distinguish between product data which is to some extent final and product data which may still be subject to change (Van Veen 1992).

Version control - In concurrent engineering environment, information on attributes of a part that may be of value to a downstream function need to be released as soon as they become available, without waiting for complete definition of the part (Roy and Allchurch 1997). Thus, the information would be passed to the downstream functions when it is still going to change or evolve. Well defined procedures for implementing the effects of changes in product specifications are essential for information integrity. This calls for maintenance of different versions of the product data and it is very difficult to retain control over the process.

4.4 EXISTING PRODUCT MODELS

There is a large number of geometric-oriented product models. An integrated product model, which is one logical content that integrates all information concerning a product, is the latest type of product model and is the only one considered for analysis here. From the literature, CEDS - Concurrent Engineering Data Structure (Lindeman

and Wijaya 1992), UPDE - Unified Product Data Model (Cleetus 1995), KOF - Knowledge-integrated Object-oriented Feature-based product definition model (Zhu et al. 1993), LPDE - Leeds Product Data Editor (Erens et al. 1994), PSE - Product Structure Editor (Krause 1989) and Chromosome product model (Andreasen 1990; Andreasen et al. 1996) are described in this section.

4.4.1 Concurrent Engineering Data Structure

Lindeman and Wijaya (1992) argue that the concurrent engineering methodology depends upon the ability to share up-to-date engineering product structure data. A standard format for the product structure data which takes into account structure hierarchy and all the associated parametric data needed for each component of the product structure has been defined in the Concurrent Engineering Data Structure (CEDS). It enables sharing of design information stored in a single database among the design team members (Figure 4.8), and is used and updated from conceptual design through to engineering design approval. The concurrent process iterates around the product structure. CEDS is part of a product engineering project's overall data management solution.

In the implementation version, CEDS mainly concentrates on mechanical, geometric design data and the product structure. Theoretical concepts for structure data creation, maintenance, releases are discussed in the data management system of CEDS but neither designed nor implemented on computers. Evolution of product data is also not implemented, and CEDS does not address product functionality in the data structure.

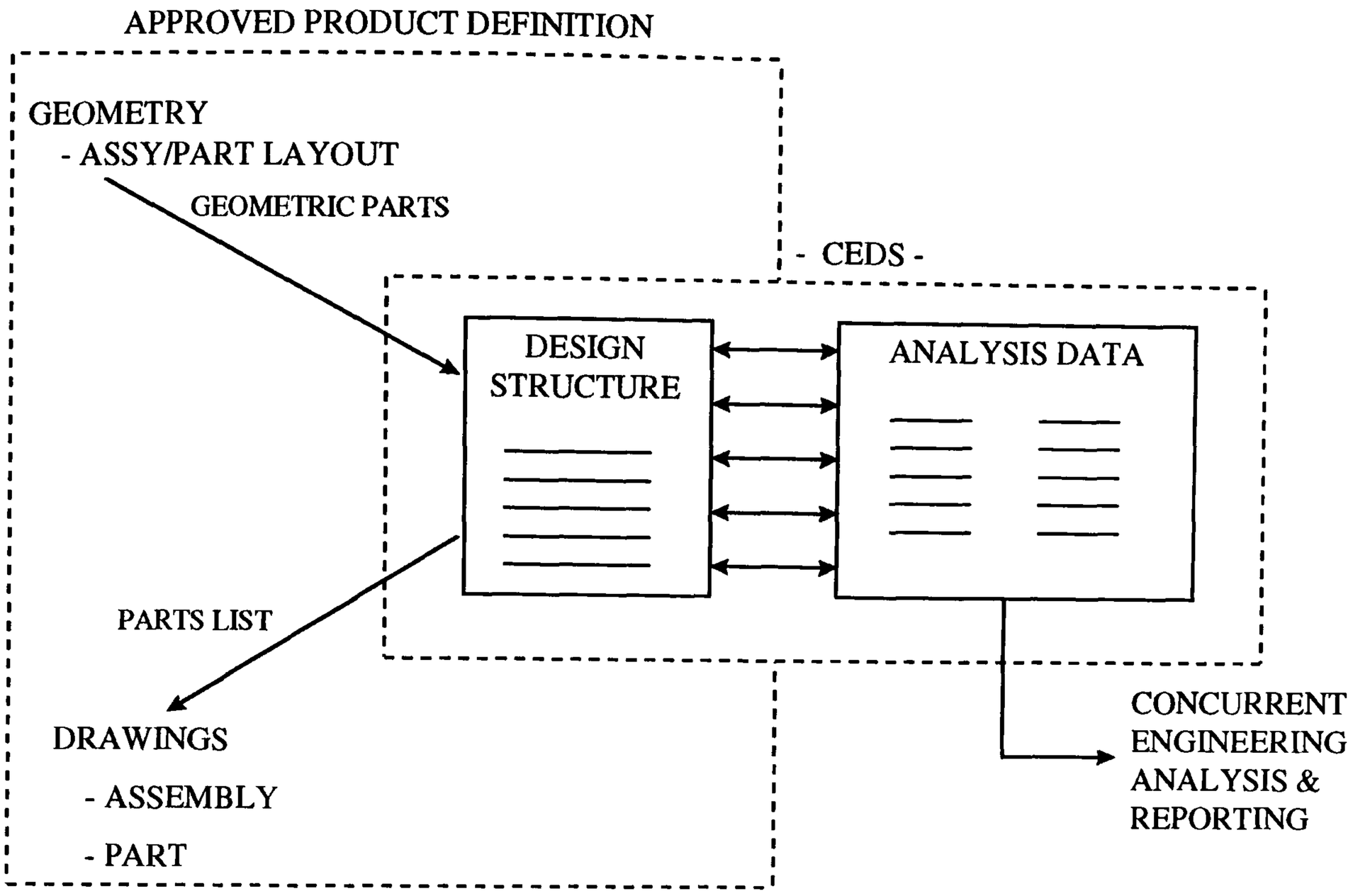


Figure 4.8 Concurrent Engineering Data Structure

(Lindeman and Wijaya 1992)

4.4.2 DICE Unified Product Data Model

DARPA Initiative in Concurrent Engineering (DICE), a US government initiative for concurrent engineering, has focused on how best to share the product information amongst a team. The data sharing in product development must concern not just the finished design, but also the evolving, incomplete design which is constantly changing throughout the project (Cleetus 1995). The unified product data model (UPDM) has been built as a product structure tree with the basic *part-of* relationship to express the successively lower levels of decomposition of the product. The concurrent engineering aspect of the data model consists of the fact that any individual node of the tree representing an assembly, sub-assembly, or component has multiple slots to represent the various perspectives of that node (Figure 4.9). The UPDM is a directory for all the

data that arise in the course of product development and are needed to define the product in all stages of the development, and for all perspectives of the product.

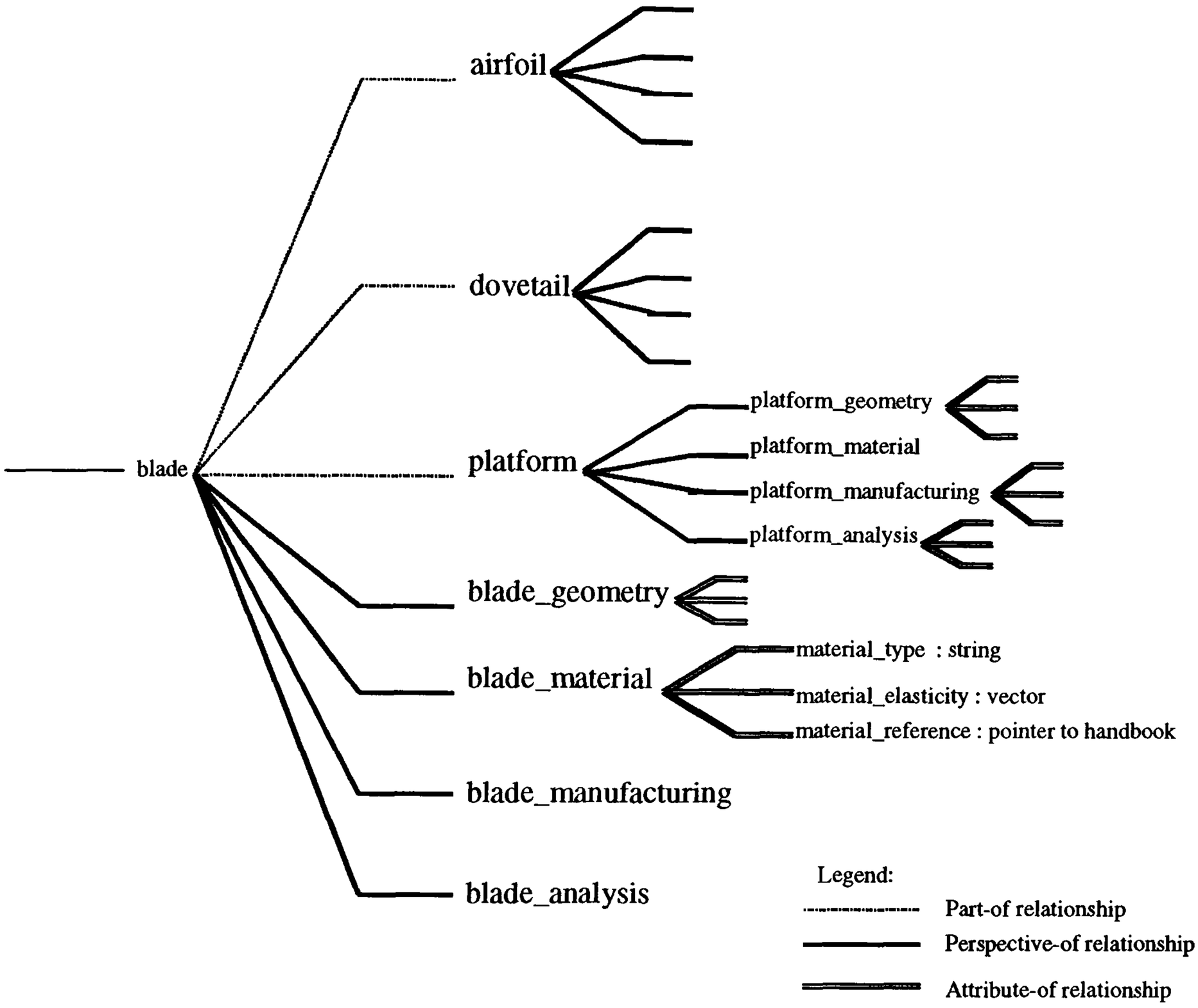


Figure 4.9 Example of a unified product data model of a turbine blade in an aircraft engine (Cleetus 1995)

The model and the accompanying data dictionary are considered as meta-data, i.e. data that describe the definition, structure, form, purpose, units etc., of the data stored about the artefact. Cleetus (1995) proposed that such meta-data be extended to include access methods to obtain the artefact data, and the UPDM should hold pointers to the necessary software routines for accessing the data lying in remote files. Ability to add new *part-of* nodes, new perspectives, new attributes to the product structure tree makes the model dynamic.

Evolution, versioning and reuse of product data are supported in UPDM. UPDM does not include the 'product functionality' perspective but allows the addition of new perspectives. The process by which the product is developed is not stored in UPDM. It does not discuss the interrelationships among the various perspectives (geometry, material, manufacturing and analysis).

4.4.3 KOF Product Definition Model

A knowledge-integrated, object-oriented and feature-based product definition model (KOF) which consists of a feature model, feature process model and equipment model has been developed by Zhu et al. (1993) to support concurrent design.

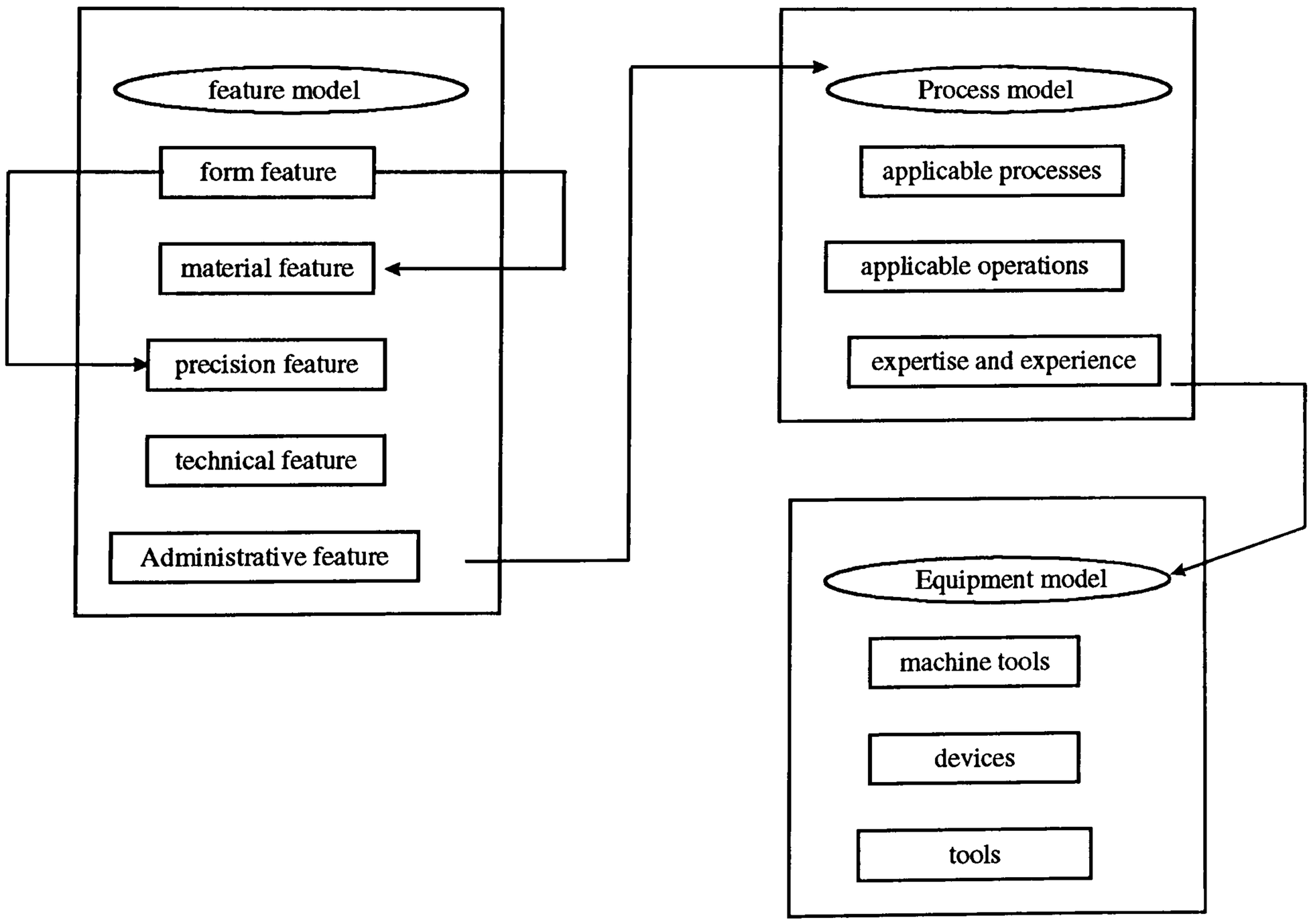


Figure 4.10 KOF architecture (Zhu et al. 1993)

Feature model references the feature process model, at the same time, the feature process model references the equipment model (Figure 4.10). A feature is defined to be a set of information related to a part's description (Shah and Rogers 1988; Zhu et al. 1993). The description could be for design purposes, or manufacturing and inspection or even for administration purposes. The administrative features include part group technology code, designer, data, batch size, etc.

Feature process model contains information about the process knowledge of each form feature i.e. the process motion that generates the form feature, and checks manufacturability of that form feature. It can be used by the process planner, and during the design process to ensure manufacturability.

Equipment model describes equipment needed by the process, such as machine tools. The feature process model references it.

The process mentioned in this model refers to the manufacturing process and evolution of product data is not supported by KOF.

4.4.4 Leeds Product Data Editor

Leeds Product Data Editor (LPDE), developed at the University of Leeds, is a general software tool for handling structured data, such as product data. The LPDE is implemented using a structure editor (SE). Structure Editor allows the creation and editing of the structure and contents of product data. The definition of the structure of data is a meta-structure (or schema in database terms). The meta-structure itself is regarded as a structure and is defined using the SE (Erens et al. 1994). Structures are directed graphs consisting of nodes and relations, and the types of nodes that can be defined using SE are:

- a collection (COL) of other nodes (group)
- a selection (SEL) of other nodes (choice)
- a list of nodes, which are all of the same type
- an atom, which is a leaf in the structure

Figure 4.11 shows the way in which LPDE structures a product 'truck' using SE.

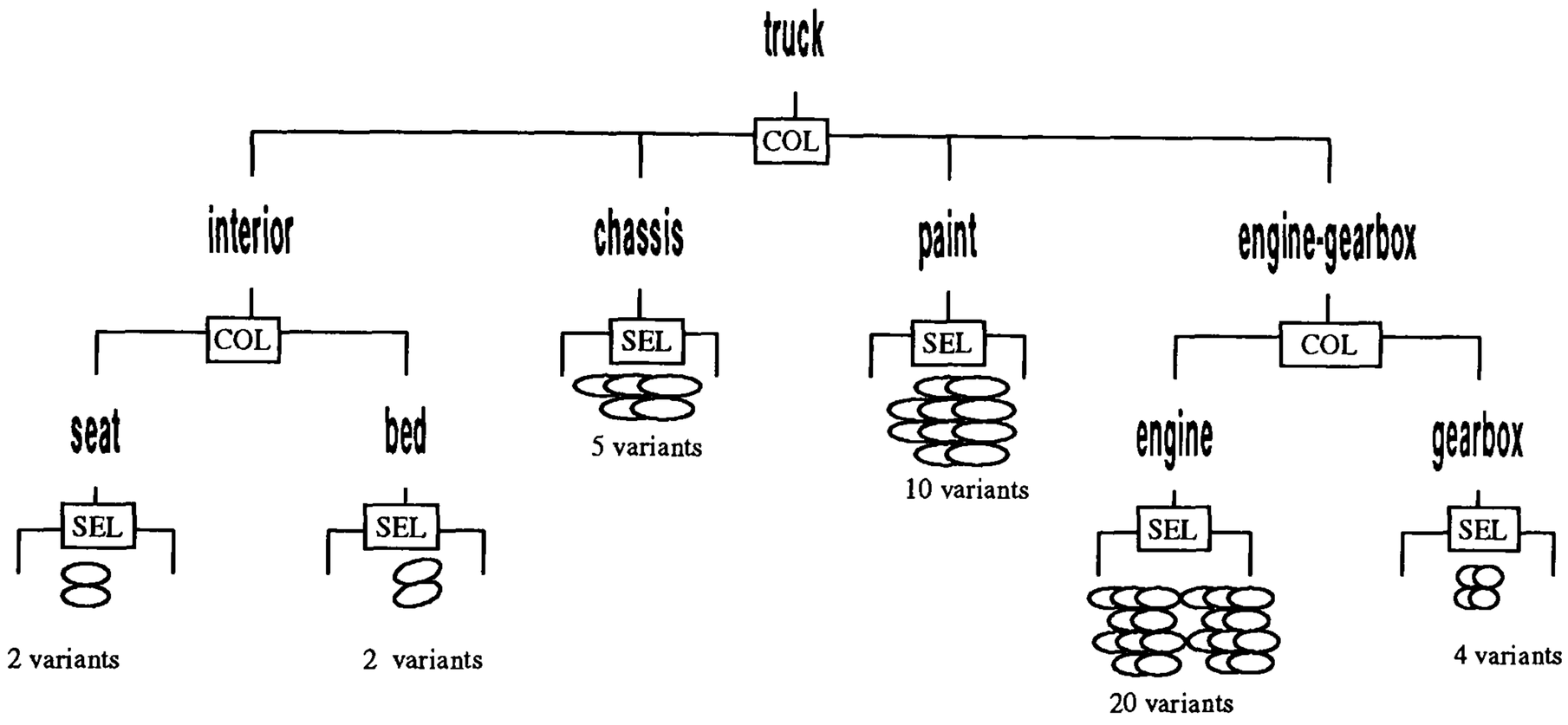


Figure 4.11 Example product structure of truck using SE (Erens et al. 1994)

SE concentrates on static data aspects and does not support evolution of product data, multiple views of the product data and the process that generates the product data.

4.4.5 Product Structure Editor (PSE)

Krause (1989) proposed a knowledge integrated product modelling method for design and manufacture. Features are used as a key integration element. Semantic features are viewed as geometry oriented objects described by data attributes containing static attributes, rule/method attributes defining a specific behaviour of the feature, and relation attributes defining the interdependencies between semantic features or relations to form features. Form features are defined as a structured grouping of geometrical entities without any semantic.

The information basis to store all relevant results of the design process is the product model. Every component of the product within the product model is related to information about the application model. Semantic classification of all information is carried out and information is arranged in different information layers. In between and within these layers, connections of information are realised by information link layers. All information layers are managed and referenced by logical entities of the organisation layer.

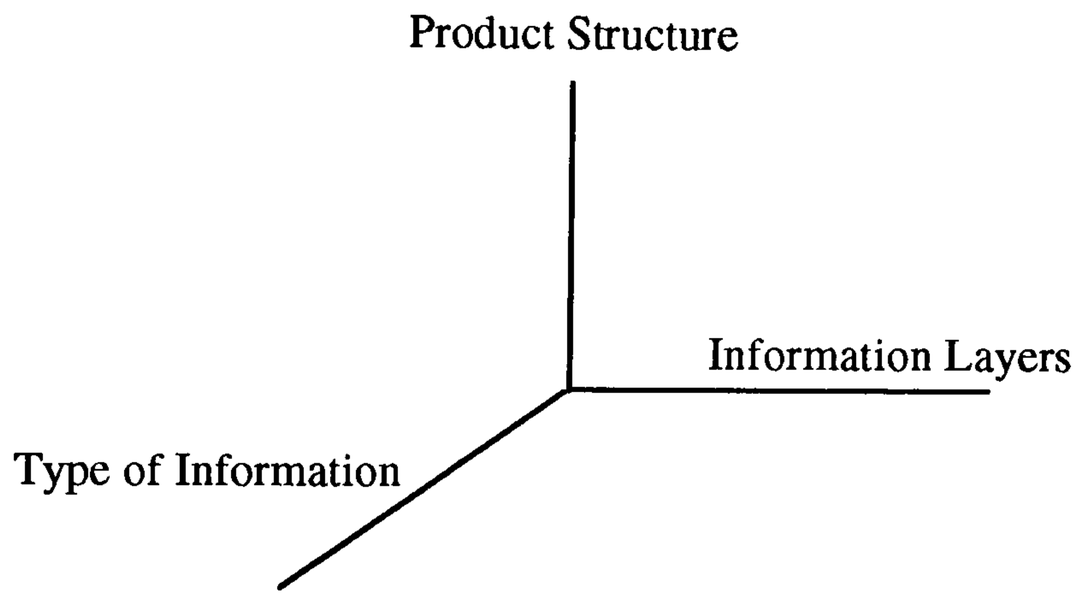


Figure 4.12 Dimensions of product modelling in product structure editor

The concept of the product model is based on product structure, information layers and the type of information (Figure 4.12). Information layers include production planning, sales, product development, process planning, manufacturing and assembly. For example, types of information for the development layer includes concept (function diagrams), 3D models (volumes, surfaces and wireframes), calculation (thermal analysis), 2D models (views, NC geometry). The product structure editor (Figure 4.13) gives the designer the opportunity to graphically set up the product structure. Functional components serve as building blocks for the product structure, and assemblies are defined by grouping functions. The product model at any time also represents the current status of the design work of each element. A relation called

connects is used for representing the relationship among elements that are grouped to meet a given functional requirement, and *contains* is a relation used to represent the physical structure of the product (an assembly contains sub-assemblies, a sub-assembly contains parts). The cost of the assembly consists of the costs of functional components, of additional connectors and of all assembly processes necessary to build up the assembly.

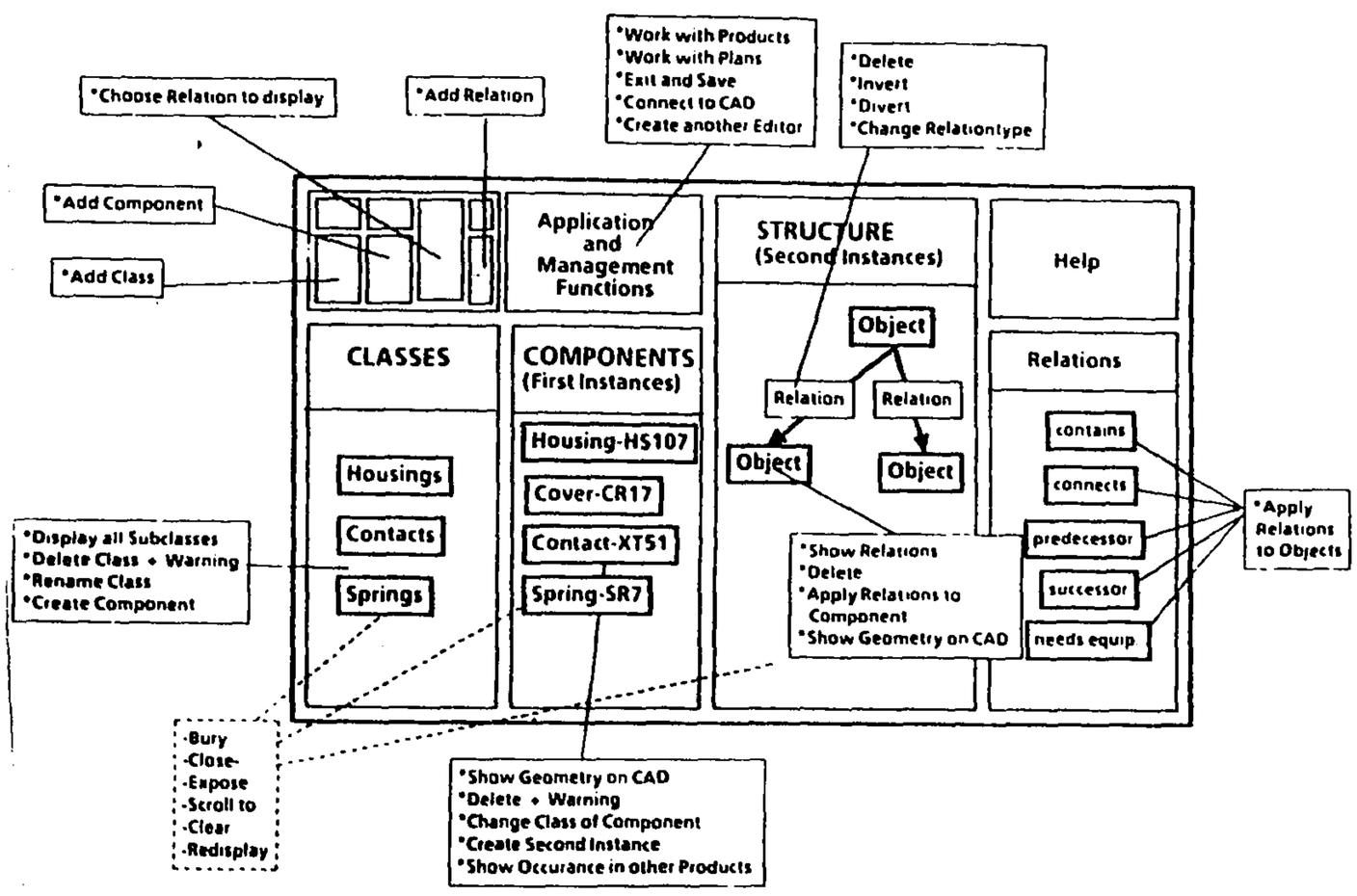


Figure 4.13 Product structure editor (Krause 1989)

Information-oriented integration models provide the basis for various kinds of applications. Product, process, factory and application models allow the storage of all relevant information needed and created during the product life cycle. Evolution of the product data is implemented by addition and deletion of classes, objects and relationships. There is no provision for updating a class or object or relation, nor maintenance of attributes (addition, updating and deletion of an attribute) of a class or relation. The relations considered include: *contains*, *connects* between objects, and the relations represent only one-way progression. It seems data about the structure of a class is not explicitly represented and stored. Thus, evolution is partly addressed in the user-interface. The process that generates the product data, and concurrent

engineering are not addressed in this structure editor.

4.4.6 Chromosome product model

A genetic model for the results of the design task, the Chromosome product model, was designed at the WDK School, Technical University of Denmark. This model views a mechanical product from four different system views: process, function, organ and construction (parts) (Figures 4.14, 4.15). The process system describes the appropriate transformation (of material, energy or data) which takes place in the product, the function system describes the effects which the product is to create, the organ (functional carriers) system describes the entities which create the effects/functions, and the constructional system describes the way in which the organs are realised (Andreasen 1990; Andreasen et al. 1996; Mortensen and Andreasen 1996).

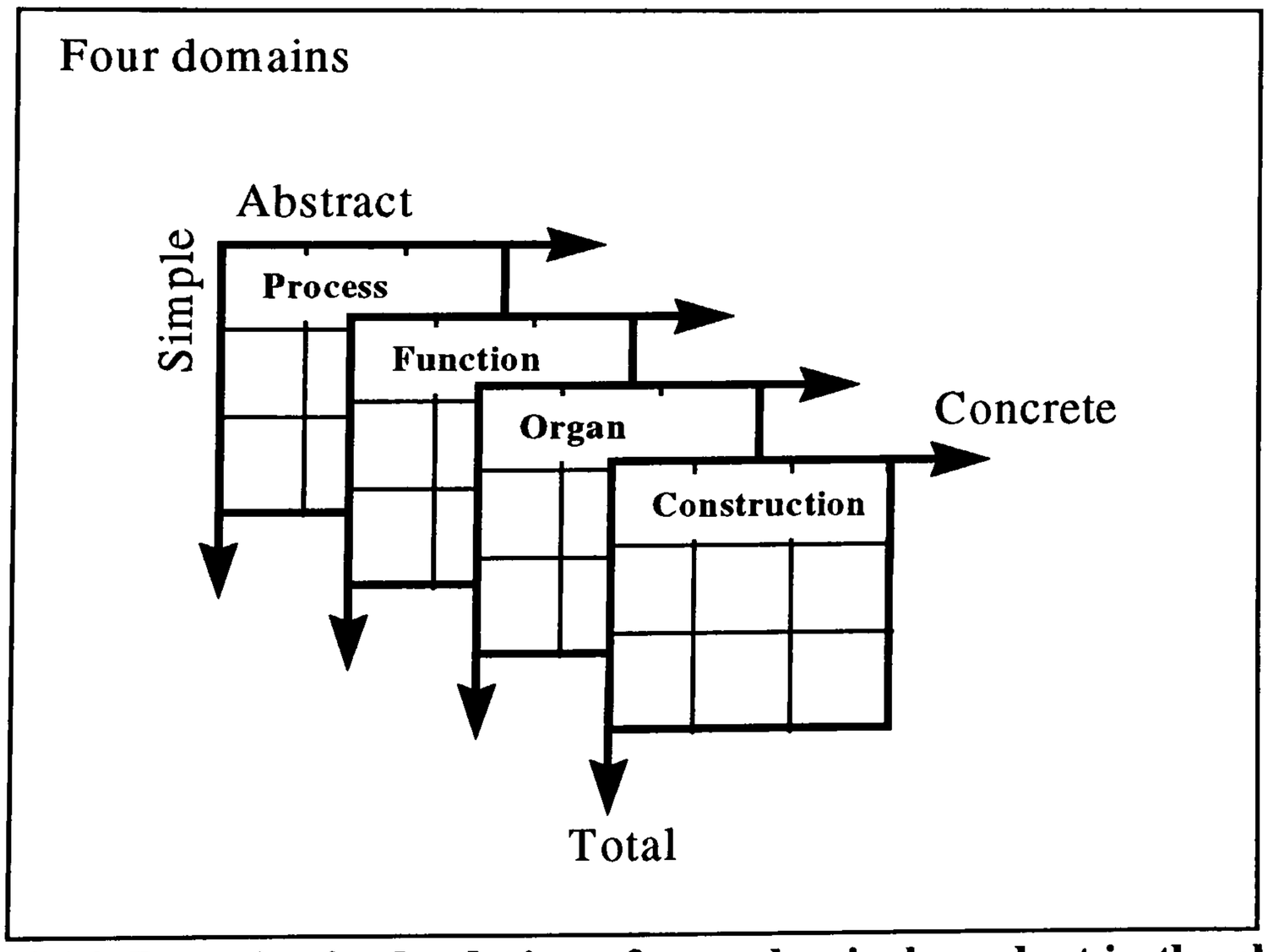


Figure 4.14 Domains in the design of a mechanical product in the chromosome product model (Andreasen 1990)

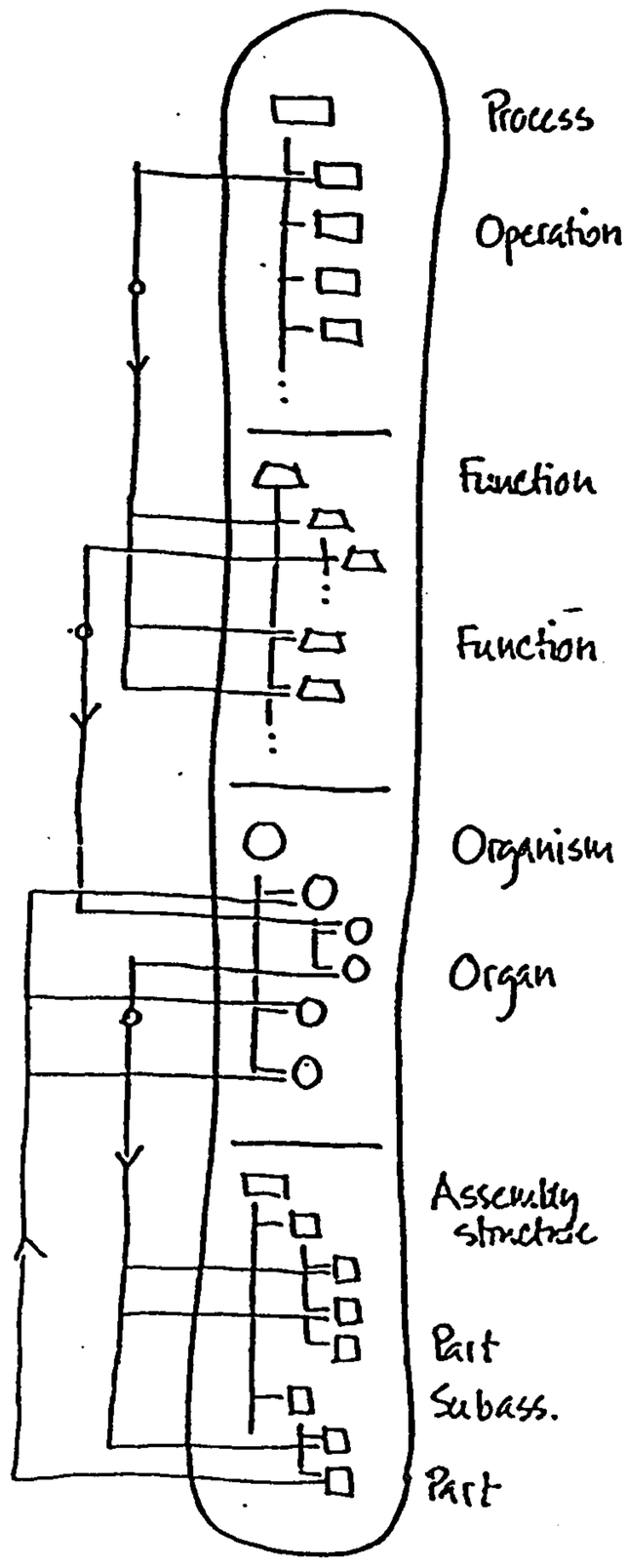


Figure 4.15 Chromosome product model (Andreasen 1990)

Functions are realised by organs and organs are materialised by parts. Decomposition within the organ and part domain involves functions (i.e. function/means law). The goals (functions) and means may be seen as a hierarchical structure where means seen top down becomes goals seen bottom up. The function/means tree is a hierarchical arrangement of function levels and means levels connected with lines that correspond to causal relations (Figure 4.16), and it does not show the relationships between the means.

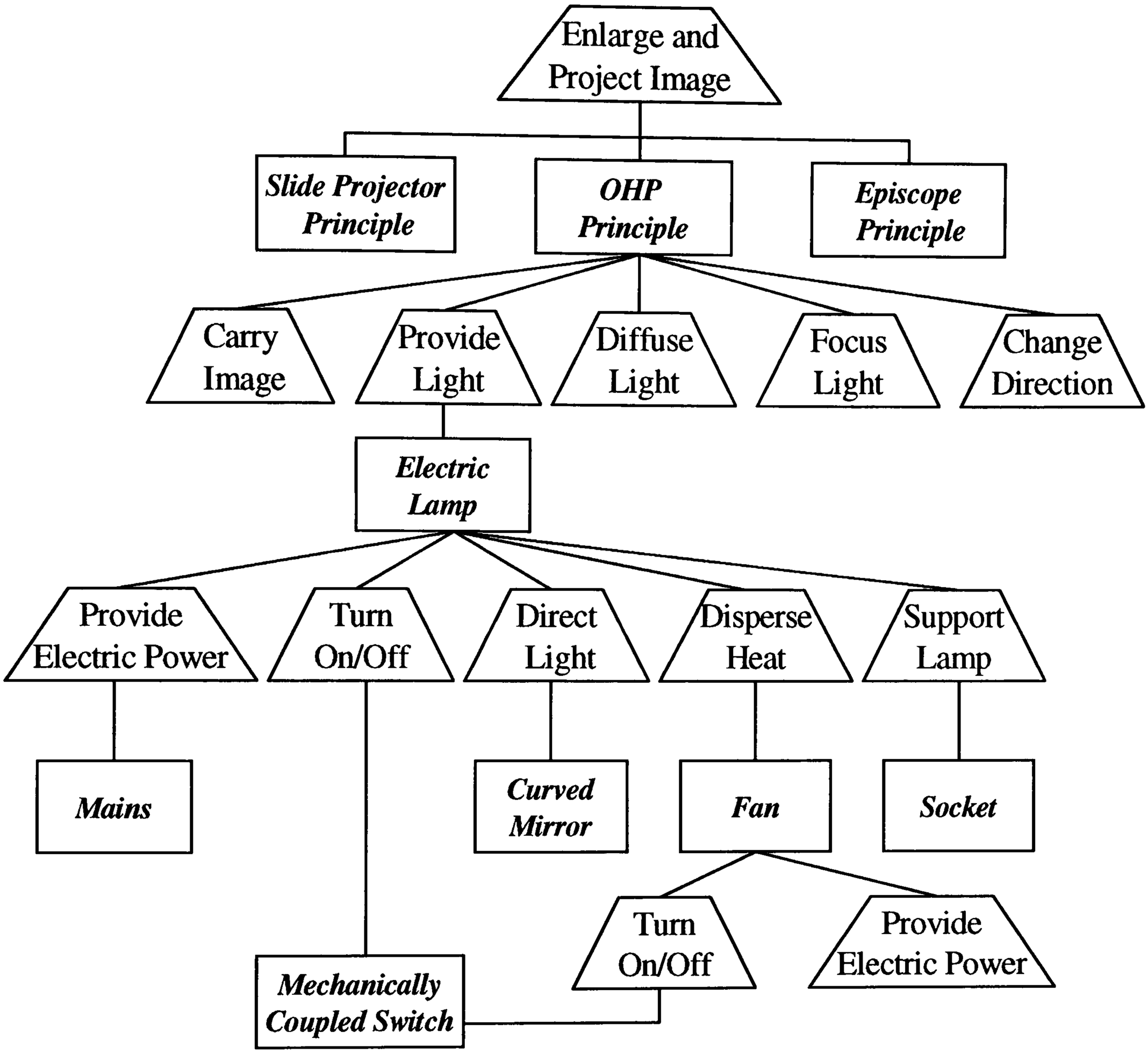


Figure 4.16 An example of a function/means tree (Andreasen 1998)

The chromosome model is at a conceptual stage and uses relational concepts. There is no computer support model or software developed based on this model yet. Mortensen and Andreasen (1996) propose an object-oriented implementation of the model. It does not address evolution of product data and concurrent engineering. The process that is addressed in the model refers to the technical process which describes the transformation (of material, energy or data) that takes place in the product, not the process that generates the product data.

4.5 COMPARISON OF PRODUCT MODELS

The above product models can be compared based on the data structure, relationships used in the model, their support for evolution of product data, reuse of historical data, versioning, concurrent engineering, link to the process that generates the product data, and the perspectives of the product data included in the model. The results of the comparison are shown in Table 4.1.

Design structures are hierarchical in nature. Most of the product models use hierarchical tree structures. Object-oriented concepts relate objects to one another in a hierarchical form. Objects inherit the characteristics of its parent object whether or not the objects are of the same type. As there are many different types of data in a product structure (mechanical, electrical, etc.), and as there are complex relationships among these different categories, researchers usually propose an object-oriented implementation.

Even though evolution or dynamic capability is identified as an essential feature of product models, there are only proposals in the literature, and not much effort has been made at the design and the implementation level of the product models. It is a similar case with reuse of historical data and versioning of product data for change management.

Product model data is one of the most important results from process chains, and process chains use the product model data. When concurrent engineering technique is to be applied in the product introduction project, it is necessary to see what is the

Table 4.1 Comparison of product models

Product Model	Area of Application	Data Structure	Model / Software	Relationships	Evolution	Reuse	Versioning	Support for CE	Process (Product Data Generator)	Link to Process or Project	Perspectives						
											Functionality	Structure	Geometry	Design	Material	Manufacturing	Analysis
CEDS	Detailed Design	Hierarchical	Model	Selection Collection List	Proposed at conceptual level	NO	Proposed at conceptual level	YES	NO	NO	NO	YES	YES	YES	NO	NO	YES
UPDM	Concurrent Engineering	Hierarchical	Model	Part-of Perspective-of Dependent-on	Proposed	YES	YES	YES	NO	NO	NO	YES	YES	NO	YES	YES	YES
KOF	Concurrent Design	Object-Oriented	Model	Available but not detailed	NO	NO	NO	NO	NO	NO	YES	YES	YES	YES	YES	NO	NO
LPDE	Structured Data Editing	Hierarchical	Software Tool (Structure Data Editor)	Selection Collection	Proposed at conceptual level	NO	NO	NO	NO	NO	NO	YES	NO	NO	NO	NO	NO
PSE	Design and Manufacture	Object-Oriented and Relational	Software	Contains Connects Predecessor Successor Needs-equipment	Partly Addressed*	NO	NO	NO	NO	NO	YES	YES	YES	YES	YES	YES	NO
Chromosome	Machine Design	Relational	Model	Part-of Consists-of Effected-from Effecting-on Connected-to	NO	NO	NO	NO	NO	NO	YES	YES	NO	YES	NO	YES	NO

CEDS – Concurrent Engineering Data Structure
UPDM – Unified Product Data Model
KOF – Knowledge-integrated, Object-oriented, Feature-based product definition model
LPDE – Leeds Product Data Editor
PSE – Product Structure Editor

* Addition and deletion of class, object and relation are addressed at the user-interface level. Updation of class is not addressed. Maintenance (addition, updation and deletion) of attributes of a class or relation is not addressed.

status of the results of the project/process (i.e. the product) in order to manage the overall project. Even though it has been identified that the process chains should be the driving factor in the development of product models to support product modelling application and research efforts, existing product models do not store the information about the process or a link to the process that generates the product data.

Product functionality is a driving factor for the success of the product introduction project, but very few models (KOF and Chromosome model) address product functionality. KOF product definition model represents functionality as a form feature. The concepts of the chromosome product model are developed only to assist product design. The chromosome model is available only at the conceptual stage i.e. there is no software or implementation version of it, and the link between the product functionality, process that generates the product data and other perspectives of the product are not defined.

4.6 SUMMARY

No definite and commonly agreed product modelling approaches exist to date. Depending on the application, definition of a product model varies. Geometry alone is insufficient to support many engineering activities. The superimposed nature of several structures in a product, and the type and complex relationships within and among these structures makes product modelling a complex process. Formal information models for product information are beginning to appear, but the literature does not yet address the ways in which databases should provide structurally complex information to application processes that use the product information. The semantics of the

information stored in product models are often known only within the application processes that use the product information. Some of the existing product models include process models, but these process models represent the physical processes which are required for representing product behaviour and manufacturing processes, and not the process that generates the product data. Even though process-chain driven approach to product modelling is identified as an important future research topic in database and related information technologies, there is no model that details how to link the process in the process chain that generates the product data with the product model data.

CHAPTER 5

5. INTEGRATED INFORMATION MODELLING

5.1 INTRODUCTION

Information related to the product introduction (PI) process can be viewed from multi-dimensions leading to various semantics of the information. In order to design an information model that supports the management of the product introduction process, the following fundamental research questions need to be addressed: which dimensions exist for modelling information? and how to represent the evolving information?. The focus of this chapter is on the analysis of the product introduction information in order to identify the basic elements, relationships and methods for integration. The results of the analysis, i.e. the interrelationships of the elements of the product introduction information are used to address the above questions. The demands on the database technology to develop such an integrated information model are discussed. Finally, the requirements for information management to support concurrent engineering in product introduction are presented.

Usually a three phase approach consisting of ① basic elements, ② relationships and attributes, and ③ constraints is recommended for information modelling. For a large project, a fourth phase called model integration is recommended.

5.2 BASIC ELEMENTS OF THE PRODUCT INTRODUCTION PROCESS

The information modelling process starts by identifying the real world things that are to be represented in the model and naming them. Executing the product introduction process involves management of the resources and management of the technical information generated. Basic elements involved in the management of the product introduction process include product functionality, product introduction process (aggregation of activities), PI resource and product (aggregation of assemblies) (Figure 5.1). For example, 'blade design' and 'blade weight evaluation' are activities in an aeroengine introduction process. The functionality is 'weight of the blade should be x grams', product element or component is blade, resources are a team of individuals with the necessary skills, and output information is technical information such as weight, life of the blade.

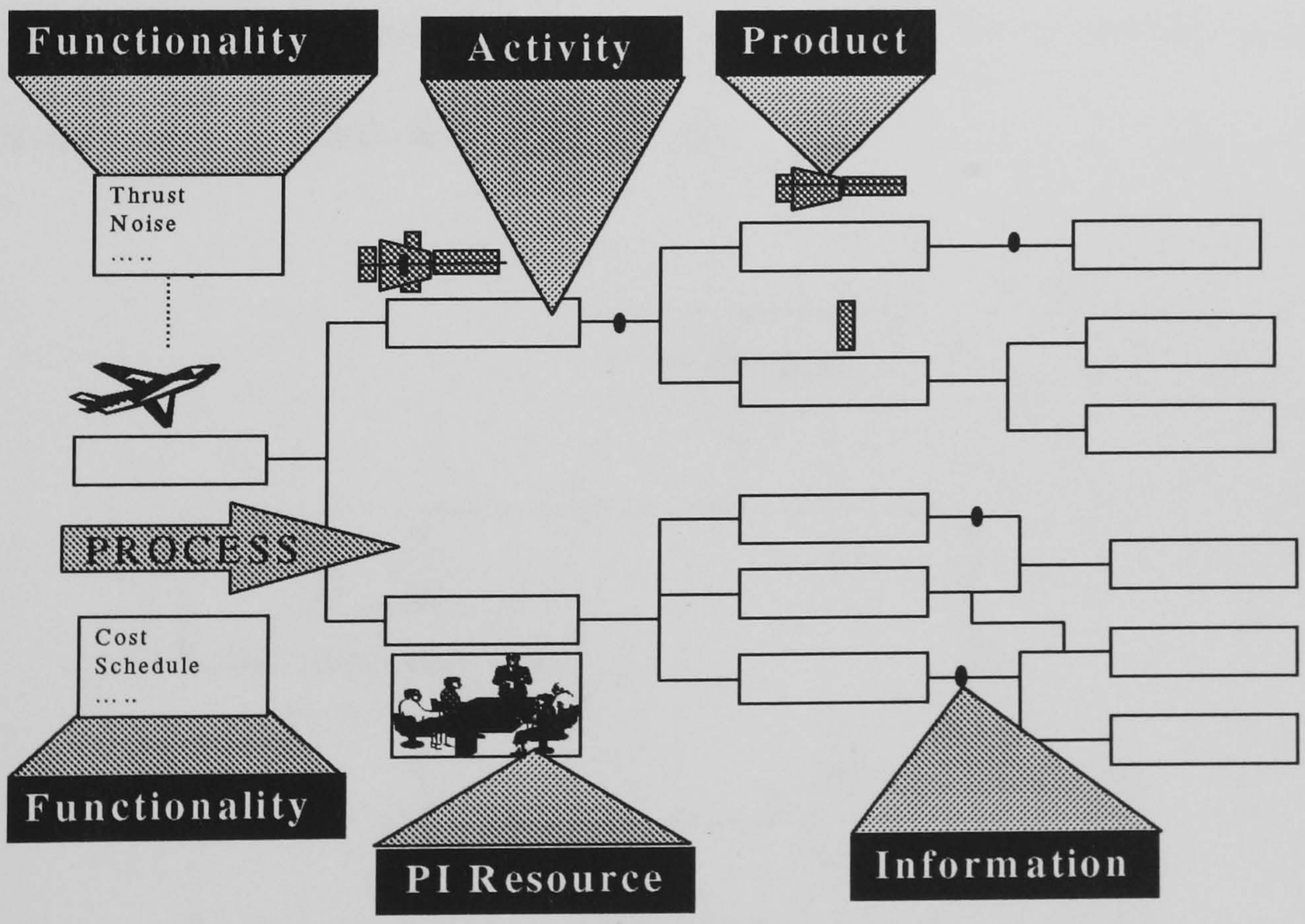


Figure 5.1 Information in the context of the product introduction process

5.2.1 Product introduction process

The business process 'product introduction' can be regarded as a project. From project management point of view, the business process would need to be split up hierarchically into sub-projects at various levels and the resulting structure could be represented as a multi-level project management network and the aim is to achieve the product functionalities. The process involves both top-down and bottom-up procedures as the goal of achieving the product functionality is decomposed into specific assembly, sub-assembly and component design requirements and passed top-down, and the component, sub-assembly, assembly design definitions are directed bottom-up for integration in order to provide a product definition that can be shown to meet the overall project requirements. A process can be defined in terms of other processes, referred to as its sub-processes (or sub-projects). Thus, the product introduction process can be defined as an aggregation of sub-processes, which in turn can be defined as an aggregation of activities (Figure 5.2). The elementary node in the product introduction process is thus, an activity.

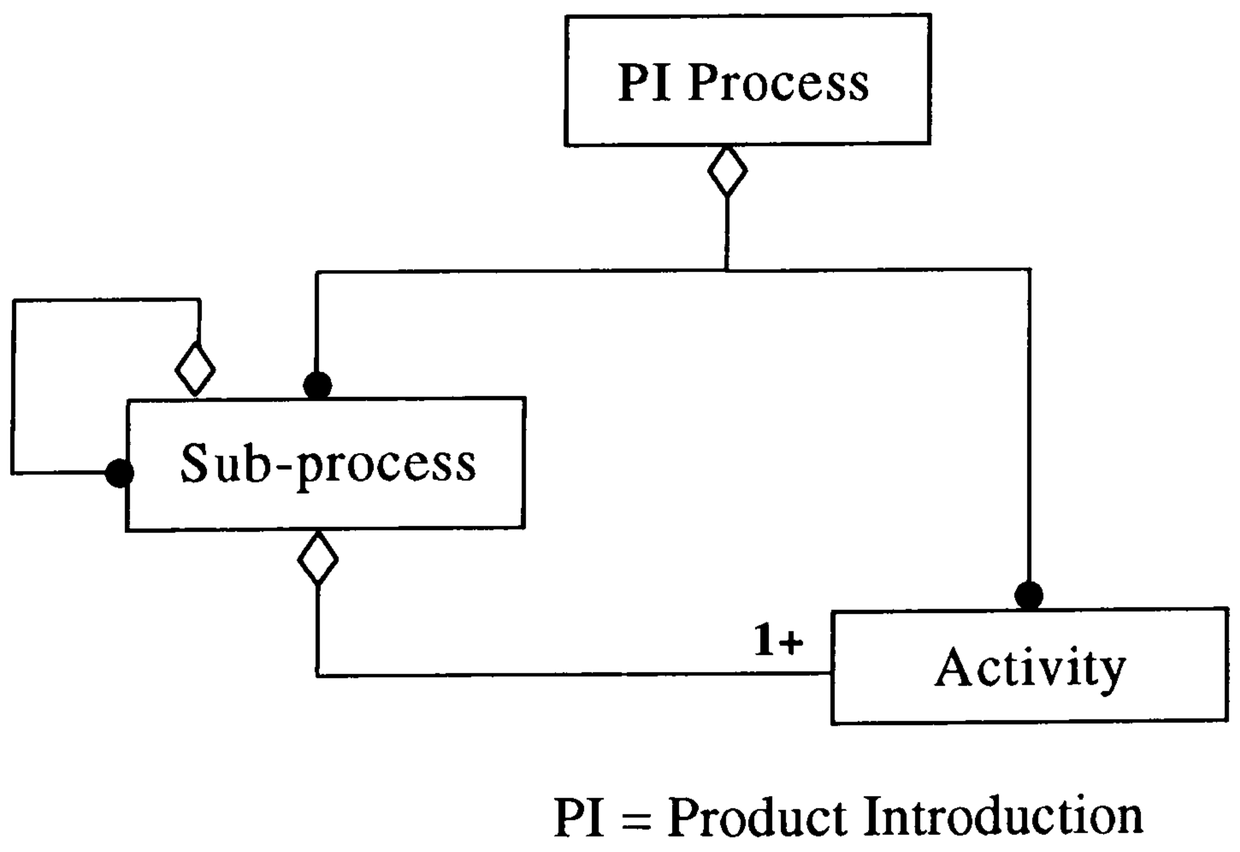


Figure 5.2 Object model of the product introduction process structure

Executing the product introduction process involves linking two different, though not disjoint, processes along the various stages of the project's life cycle. They are the technical processes (e.g. conception, design, prototyping, testing) and the managerial processes (e.g. identifying the required skills, identifying the required human resources, workflow management, etc.) as shown in Figure 5.3. The managerial process monitors the resources, coordinates the activities of the parties involved, manages the communication and information flow, and supports the technical process via decision making and data management. The technical process creates and shapes the features of the project's outcome (i.e. product) (Thirupathi and Roy 1997b).

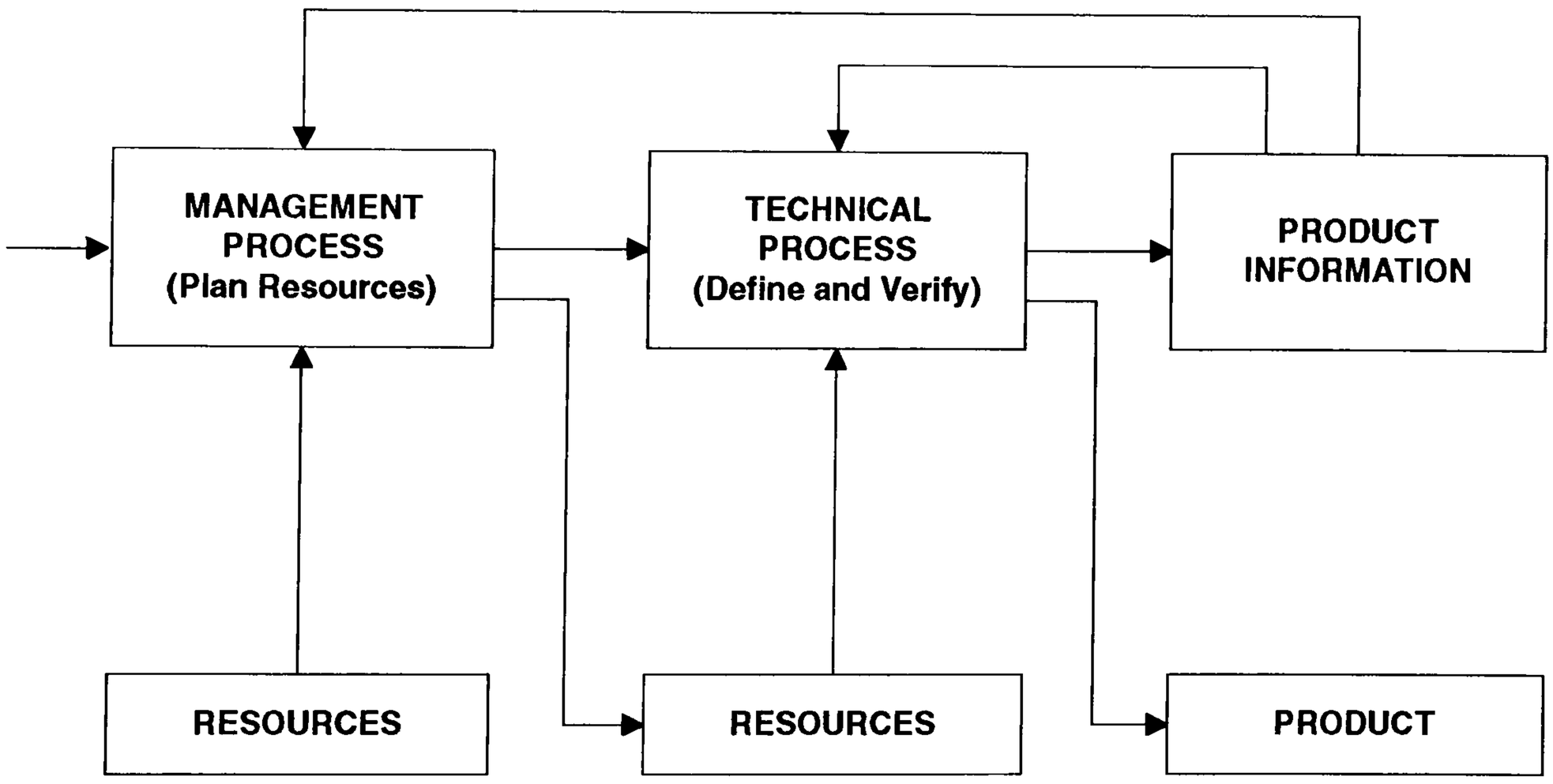


Figure 5.3 Relationship between management process and technical process

In case of activities of the type test, depending on the result of the test, the control may go to one of the previous activities at the same hierarchical level or to an activity at the previous level in the same project or to an activity in a different project. For example, consider the 'weight evaluation' activity that evaluates the weight of a product unit say 'blade' (Figure 5.4). The result information 'design weight' from the

activity would be compared with the required weight; this comparison would provide the information 'overweight' or 'within the required weight range'. Based on these information, the control of activities may enter into a feedback loop. Thus, the order of execution of the activities depends on the information that is generated from the activities. In order to find a representation for the activities and the dependencies among them, the relationships between activity and information need to be analysed in detail.

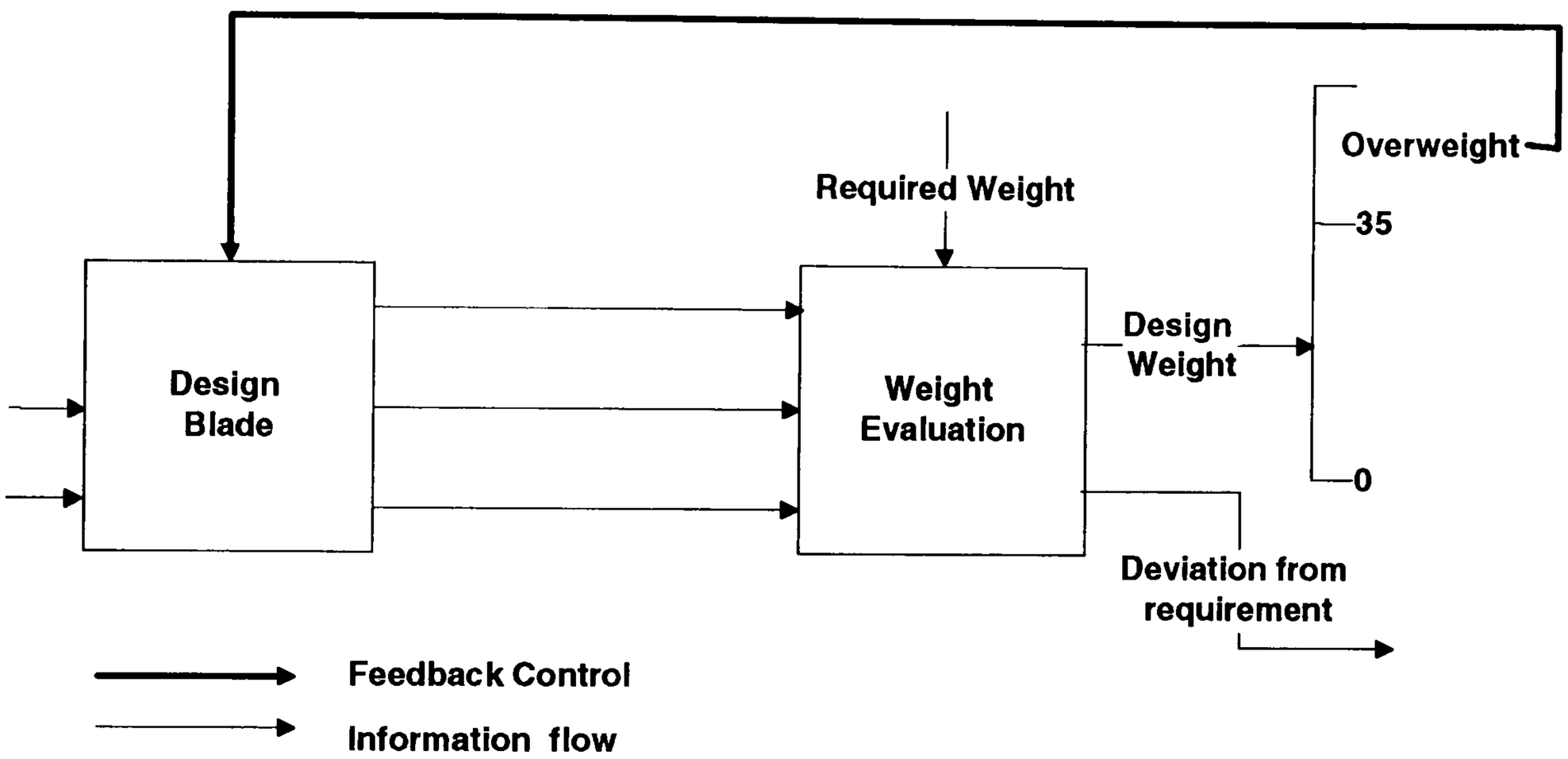


Figure 5.4 An example of an activity

The activities of the product introduction process generate information about the product, and this information is the glue binding the managerial and technical process. Apart from this, concurrent management of project time (and cost) and project objectives (product functionalities) necessitates link across the project and product information. But this link calls for special issues in information modelling. In order to explain one of the special requirements of the information model for the management of the product introduction process, the following example is given.

Consider the product introduction framework for the product aeroengine. Project managers will be using the framework for planning and management purposes, while individual engineers will use it for storing, retrieving and manipulating the technical data (design specifications). For the process of designing a blade which is used for the development of aeroengines, project managers will be mainly concerned with the development time and resources, and will need database entities for activities, responsibilities, resources, start and end-dates, etc. If an object-oriented database is used in which the entities are implemented as classes and attributes, then "blade" comes at the data level in the '*input*' and '*output*' attribute of the '*blade evaluation*' activity (Table 5.1). When the process is completed, information on the design of the blade is available, and the data is entered in a class "blade" for use by engineers, with the necessary attributes (Table 5.2) for the different blade variants, life cycle, a reference to a CAD drawing, a list of components, production method, etc. It can be seen that "blade" is information at both:

- the data level (object "blade" of class "aeroengine development", Table 5.1) and
- the schema level (class "blade", Table 5.2)

where a schema defines the data structure (records) in the database, the elements they contain and the relationships.

Table 5.1 Some of the attributes and data objects of the class "aeroengine development"
(Project management view)

Id	Activity Name	Responsibility	Resources	Start	Finish	Input
1	Stage 2 Evaluation	R.French	100 man years	1998-3	2000-6	{Blade,..}
2	Blade Evaluation	H.David	20 man years	1998-5	1999-7	{Blade}

Table 5.2 Some of the attributes and data objects of the class "blade" (Technical view)

Design Variant	Variant name	Life cycle	CADDs_file_PD	CADDs_file_TD
1	Version 1	10000 cycles	Lk12345_PD	Lk12345_TD
2	Version 2	12000 cycles	Lk12346_PD	Lk12346_TD
3	Version 3	14500 cycles	Lk12347_PD	Lk12347_TD

Thus, information that exists at the data level in the project view has to be at the schema level in the technical view. Existing frameworks find it difficult to accommodate the same information at both these levels because the traditional database technology allows modelling at only two abstraction levels (schema and data). Thus it creates a barrier to linking the project management view and technical view in a single framework. Apart from this, they lead to heterogeneous databases as the *blade* is stored as an *attribute value* in *project database* and is an *entity* of its own in the *product database* and the semantics (*input* in the *project database* and *part* in the *product database*) of the blade information differ in different databases.

5.2.2 Product functionality

Product functionality means the purposeful action of a product (Akiyama 1991). Concurrent Engineering requires that participants are able to share a common understanding of their underlying goals and ideas. Achieving a product that satisfies the required product functionalities is the goal of the PI process. Functions are usually defined by statements consisting of a verb and a noun. (Pahl and Beitz 1984). For example, 'provide thrust' is a function of an aeroengine, 'attach root to aerofoil' is the function of the platform of a blade. The 'why' and 'how' in the functional structure

(Figure 5.5) represent why the function exists and how the function occurs respectively.

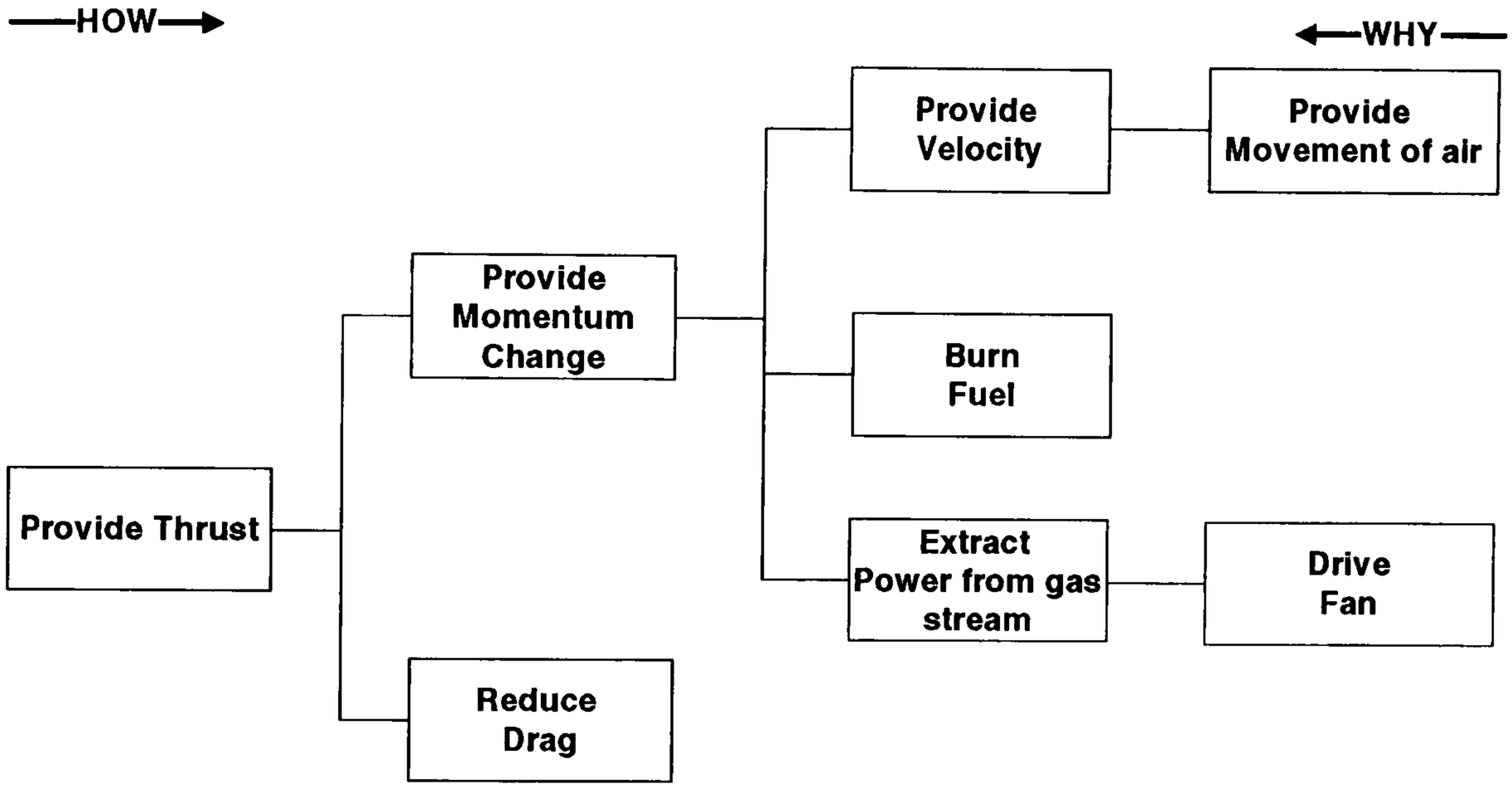


Figure 5.5 A portion of the function analysis structure diagram of the aeroengine

The product functionality of a complex product can be broken down into sub-functionalities at various levels and the functional decomposition would involve functional sub-systems. For example, the functional system of an aeroengine can be broken down into five sub-systems : fuel, oil, heat management, air (primary, secondary) and electronic system (Figure 5.6). The overall (or higher level) functionality of the product would be realised by major parts of the product that belong to different functional sub-systems.

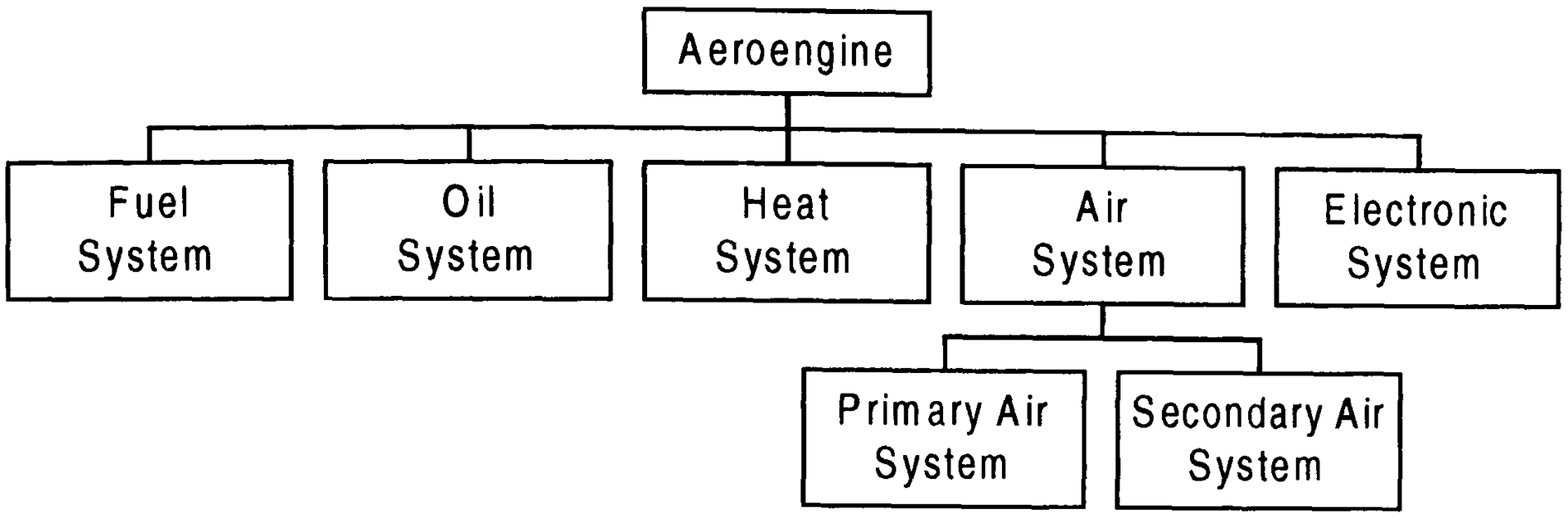


Figure 5.6 Functional subsystems of an aeroengine

Relationship with physical structure - Product introduction can be seen as a process of converting the requirements into drawings, and of assigning required functions to physical products (Figure 5.7). Thus, the PI process creates the means of achieving the product functionality, i.e. the physical structure of the product, and the product. The functional decomposition results in a number of physical components, and these components will together realise the function; however, this set of components may not be assembled as one physical unit, but distributed over the physical structure of a product. The relationship between functionality and product is many to many as each functionality may use part of many components and each component may serve many functions.

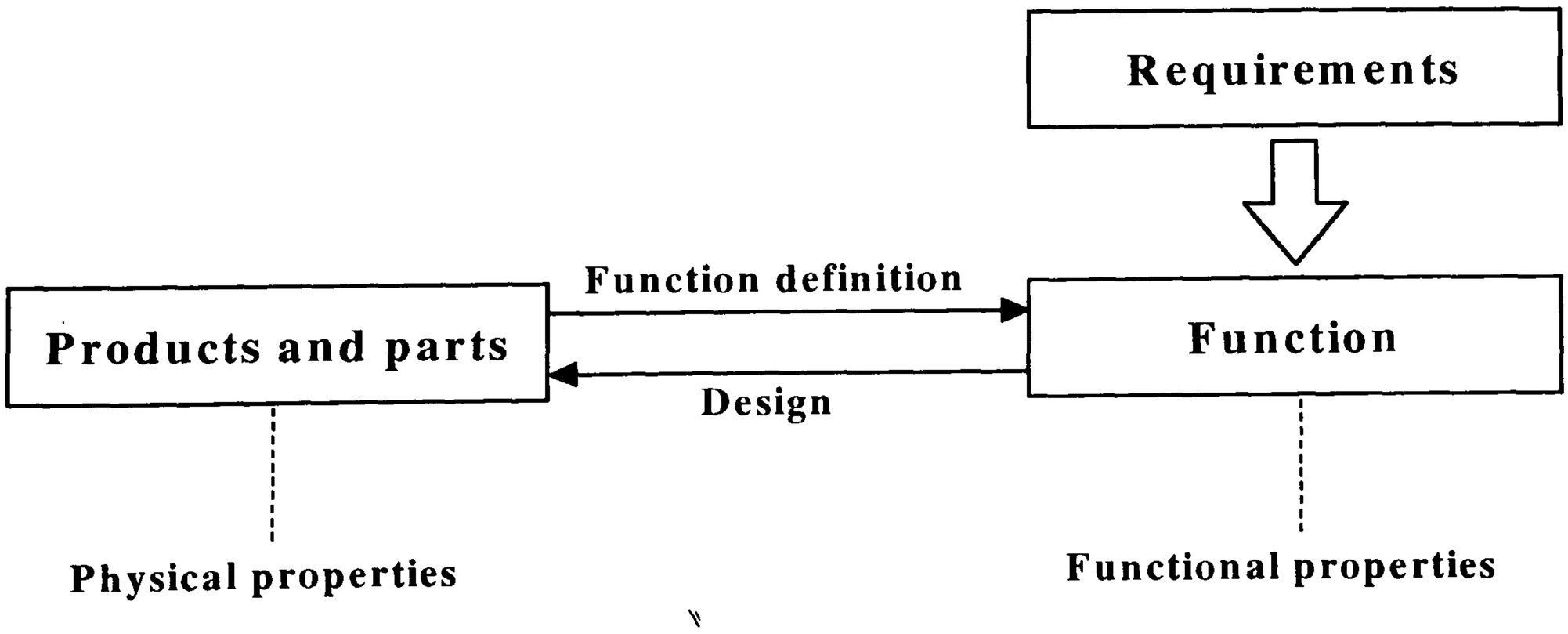


Figure 5.7 Converting requirements into physical products (Akiyama 1991)

Associated elements - A function is associated with the following elements:- ① a list of input parameters that enable the function to operate, ② responses that may be used to measure the function and ③ failure modes associated with the function. A response will probably have more than one failure mode at each of the upper and lower limits. There may be more than one response to monitor in order to evaluate the function. The output response must be of some attribute that is measurable and comparable to the engineering specification.

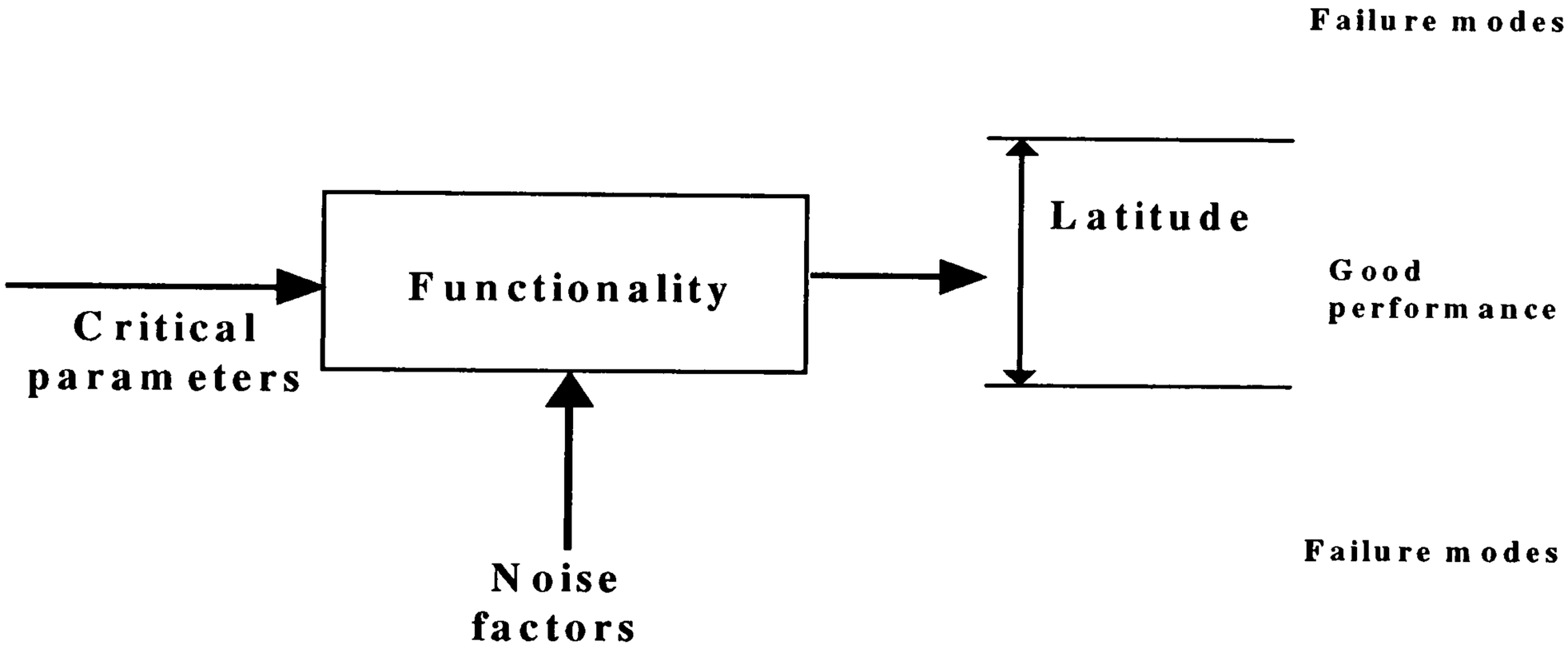


Figure 5.8 General function diagram (Fox 1993)

The function diagram (Figure 5.8) ties together the relationships between the parameters that control the function, the function itself and the outputs from the function and their associated latitude and failure modes (Fox 1993). The list of critical parameters, their nominal values and their tolerances serve to represent the design intent. In the PI process, these parameters evolve and the set of data (target values, actual values) of the parameters is continuously updated as optimisation of the product design is carried out. An example of the function 'provide design weight' and some of

the activities of the 'blade development' project that aim to achieve this functionality are shown in Figures 5.9 and 5.10 respectively.

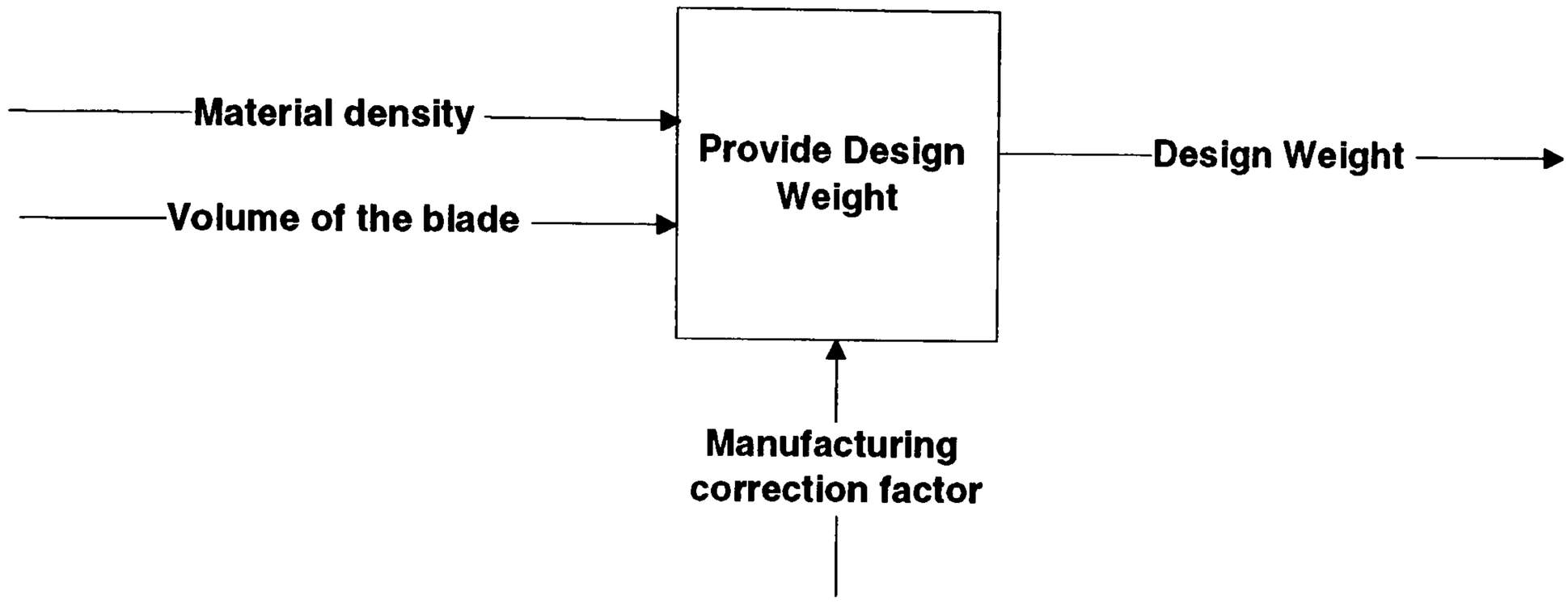


Figure 5.9 An example of a function diagram

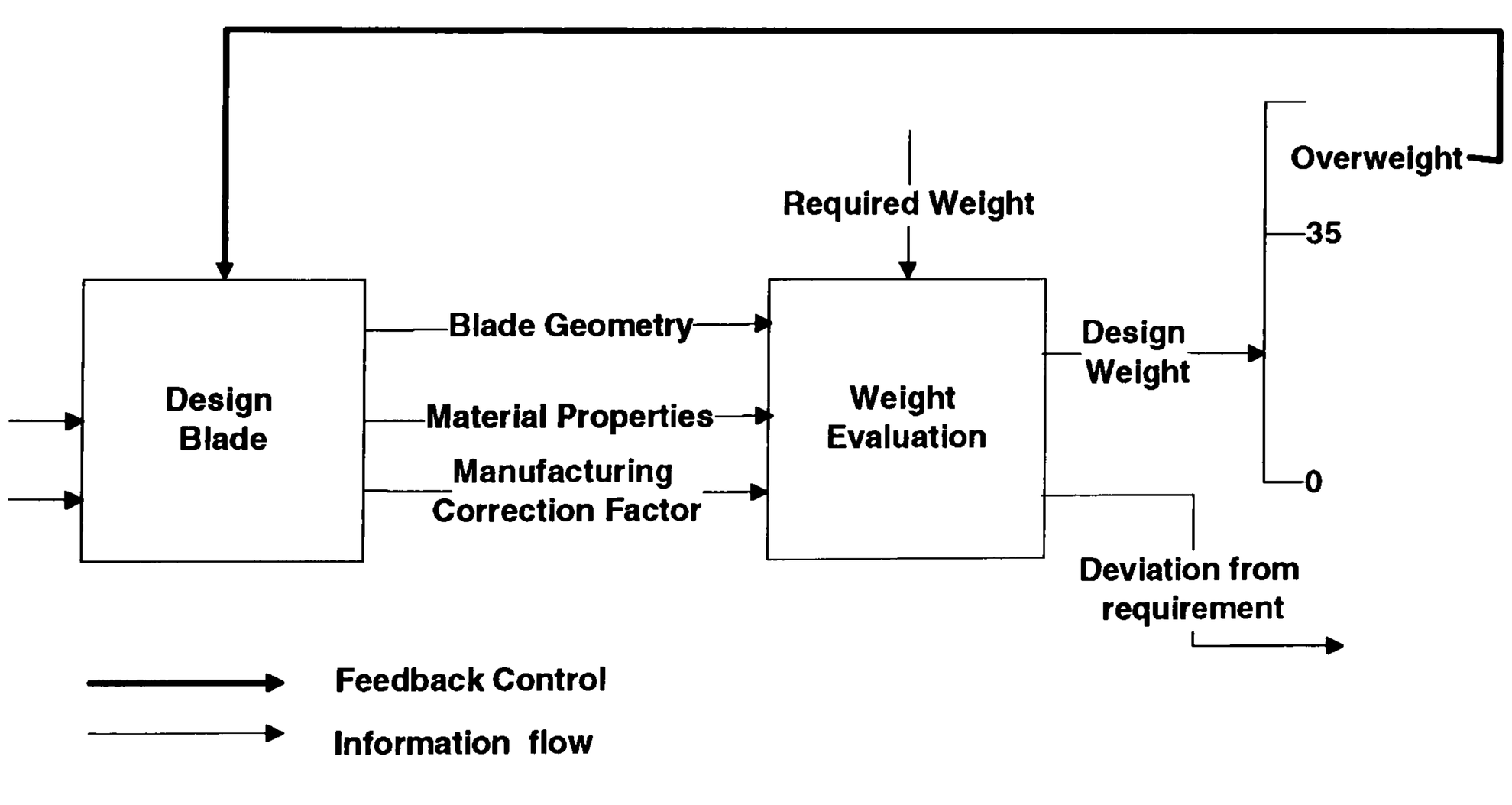


Figure 5.10 Activities and product functionality

Functionality of the product plays an important role in the product introduction process. At any stage of the process, it is necessary to see whether the product functionalities have been achieved, i.e. the status of the product functionalities have to be checked by comparing the actual values with the target values of the parameters

associated with the product functionalities. Based on this, the decision about the direction of the project, the effort and resources required to achieve the product functionalities would be planned. Thus, the activities are managed by the results of the activities and, hence, the functionality of the product is treated as a separate element from the product. The parameter 'weight' associated with the functionality 'provide design weight' would be an attribute of the component blade. Thus, functions during PI process are achieved through specifying and identifying changes in the attributes of product elements and relationships between the product elements. Over the entire design process, typically 35% of all information generated about the component are of the relational type (Ullman 1993). Ullman's model is based on the realisation that it is the relations / interactions between product elements that model functionality. According to his model, new or refined relations model the evolving functionality and behaviour of a system. It was found that 63% of customer requirements and 73% of the engineering requirements were in terms of relationships between product elements. The remainder of the requirements were attribute values of the product elements.

Baxter (1994a and 1994b) proposed a functional data model capable of representing the functional description of a product that can be used by engineering applications and defined it using EXPRESS, an information modelling language. A prototype framework for capturing information related to product function was defined using the attributes: name, inputs (set of measures), outputs (set of measures), has_need_of and performed_by (assembly, component or feature). The functional data model stands on its own and is not linked with the applications that it can support. The model represents the input and output values, but do not have any mechanism to

supply information to the applications that need this information. Thus, the link between the functional data model and its usage by an application is missing.

It can be seen that the relationships between the product elements play an important role in representing product functionality. Various researchers (Reed 1993; Gui and Mantyla 1994) have proposed and developed representation of product functionality as a functional model. Even though it is identified that common goals (product functionalities) are to be shared among the CE team members, nothing is mentioned about the representation of the link between the product functionality and the process that generates the product with the required product functionalities, and, product functionality and the PI resources.

5.2.3 Product introduction resources

In order to provide the most value for the least cost in the least elapsed time, organisations get people working together simultaneously on a PI project. A task force needs to be created which would be responsible for management, communication, involvement of the participants and other activities needed for product introduction. Linkage is defined as the extent of communication and co-ordination among the resources i.e. the participants, and linkage in PI process includes both vertical and horizontal linkages.

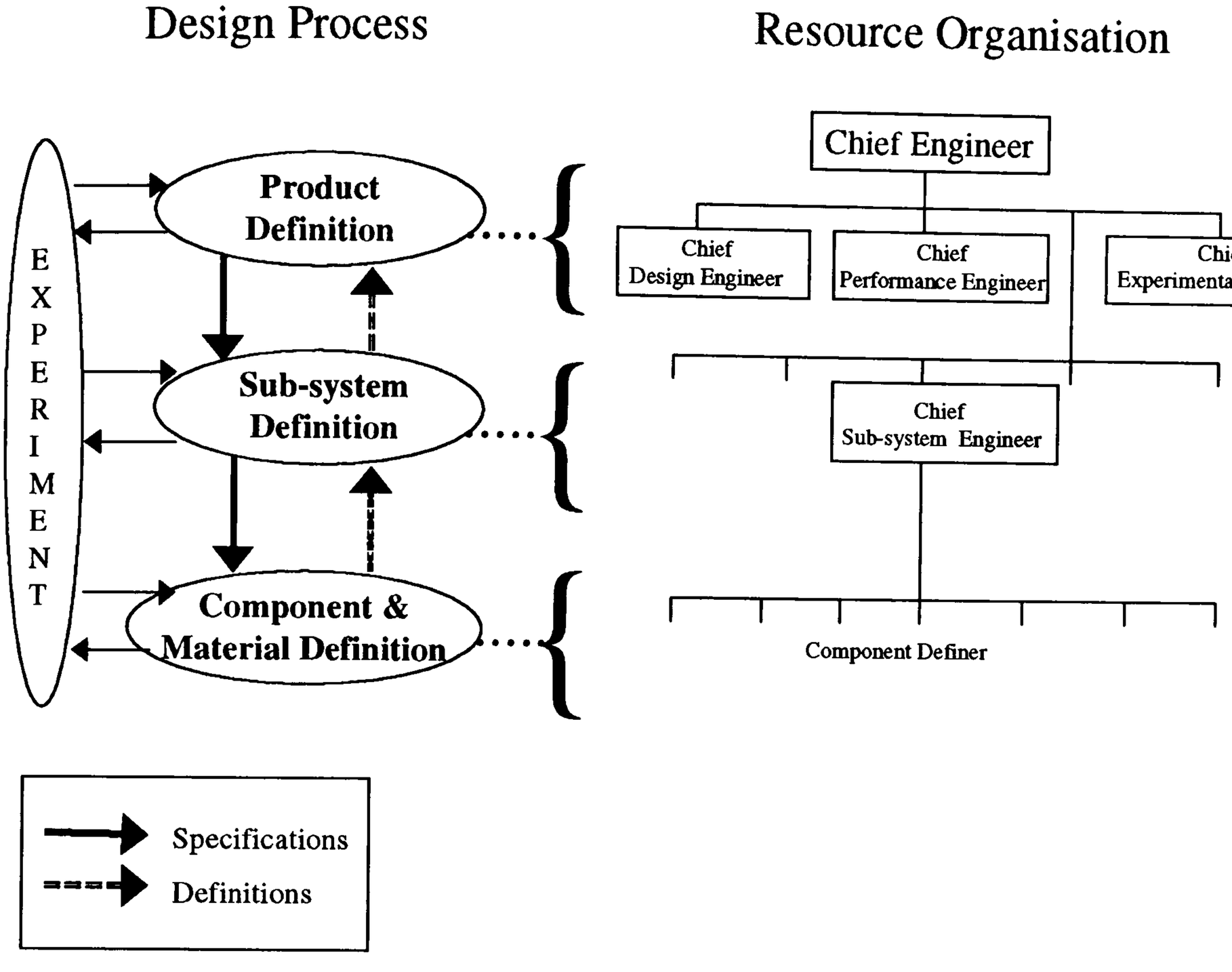


Figure 5.11 The design process and resource organisation

Vertical linkages - This type of linkages is used to co-ordinate activities between various layers of an organisation. The vertical lines represent the chain of command or reporting relationships and show who reports to whom (Figure 5.11). Employees at the lower levels carry out the activities consistent with top-level goals of achieving product functionality, and the higher level resources rely on the lower level resources to provide feedback on how well the product that is being defined meets the required product functionalities and other objectives such as time and cost. In case of problem solving, the problems can be referred up to the next higher level in the hierarchy, answer or method is passed back down to the resources at the next lower level for trial and verification. From technical innovation point of view, the PI process structure involves bottom-up flow of ideas. Thus, the link between the lower level and the next higher level of the resource structure in either direction becomes important, and the

vertical linkage involves both top-down and bottom-up information flow (Figure 5.12), and the lines of the resource organisation chart act as communication channels. The attributes that are necessary to represent such relationships are *manager* (or reports_to), and *manages* (Figure 5.13) in the data structure *person.*

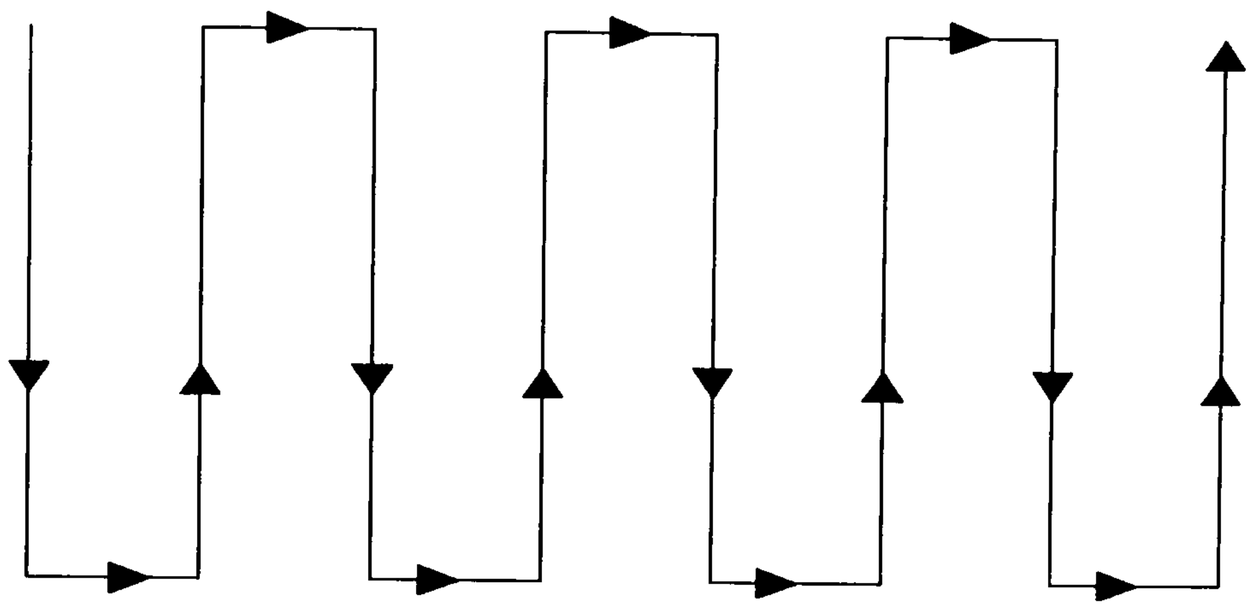


Figure 5.12 Information flow through vertical linkage

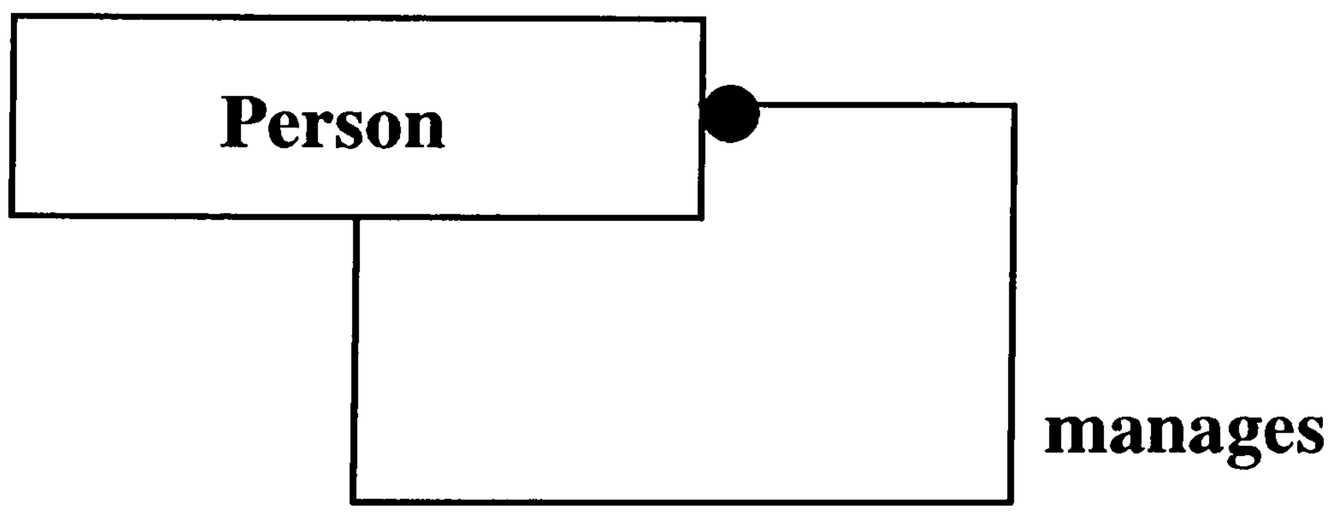


Figure 5.13 A representation of the association among persons

The work breakdown structure defines the accountabilities or responsibilities required to carry out the PI project (Figure 5.11). Performance engineer is responsible for the product functionalities. Experimenter verifies not only the design but also the method of manufacturing. Design and manufacturing definition of the sub-systems are the responsibility of the respective chief engineers. The sub-system chief engineers are in

turn responsible for the individual component definitions that make up their sub-system.

Horizontal linkage - In the case of complex products, the PI process requires a team arrangement to achieve the following project objectives:- product functionalities, specified project time, project cost and product cost. The team arrangement is between all the disciplines at all levels of the product structure and thus bringing together the business and technical skills from marketing, manufacturing, procurement and suppliers together with engineering. So horizontal linkage is an essential part of the product introduction process. The members of the team are distributed geographically across different organisations involved in the project. For example, consider the process 'blade design and development' in the introduction of an aeroengine; the disciplines involved in blade design process are: mechanical design, metallurgical support, air systems, aerodynamics and cooling design, engine performance, manufacturing engineering, mechanical test, detail drawing, vibration and stress analysis (Figure 5.14). Hence the integrated team that carries out the project includes design engineer, stress engineer, design software engineer, weight evaluation engineer and a team leader. Apart from this, the PI team would include representatives from maintenance, finance, outside suppliers and customers. This horizontal linkage helps to connect with overall goals and adequately share goals and co-ordinate with one another. The flow of information through horizontal linkage is shown in Figure 5.15 and an object model representation of the resources is given in Figure 5.16.

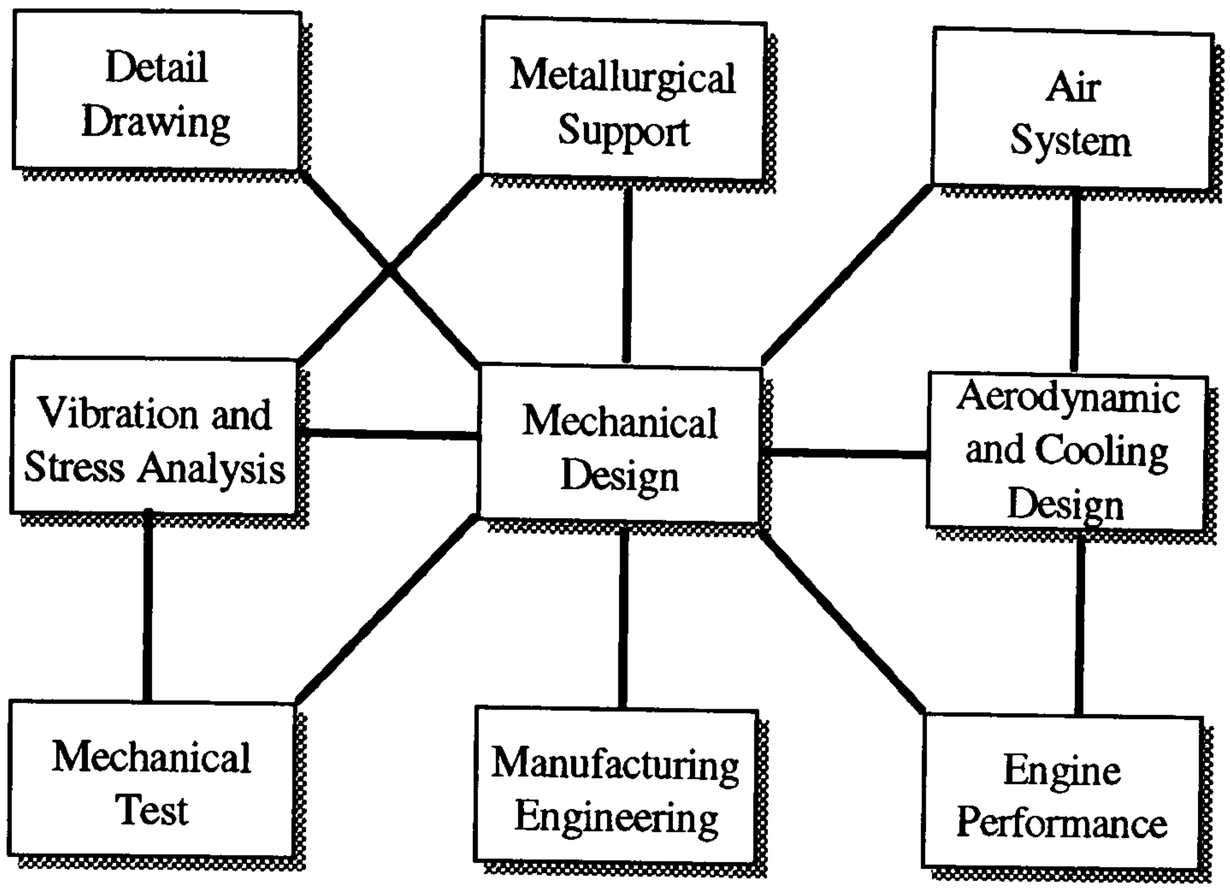


Figure 5.14 Disciplines involved in high-pressure gas turbine blade design (Murdoch 1993)

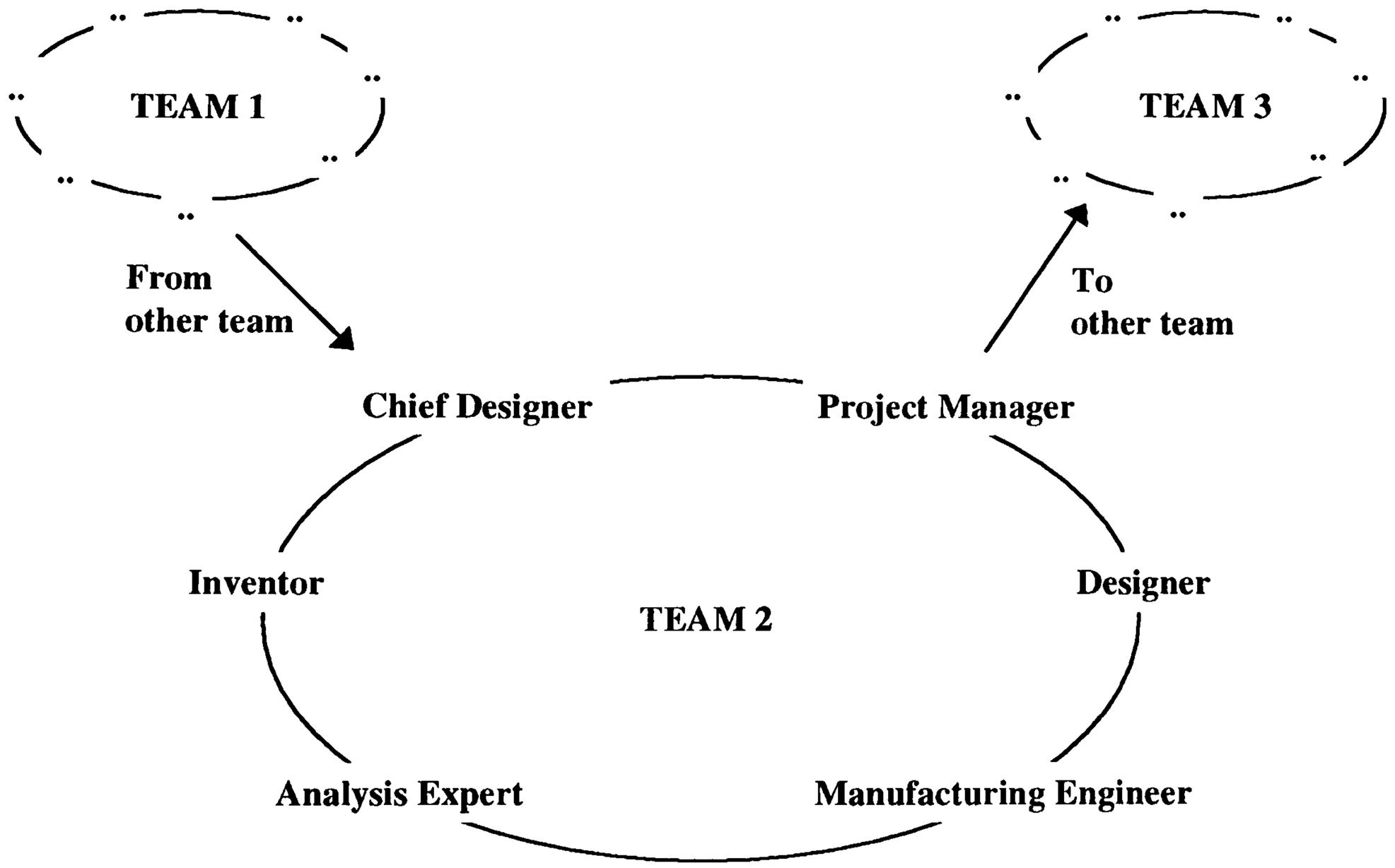


Figure 5.15 Information flow through horizontal linkage

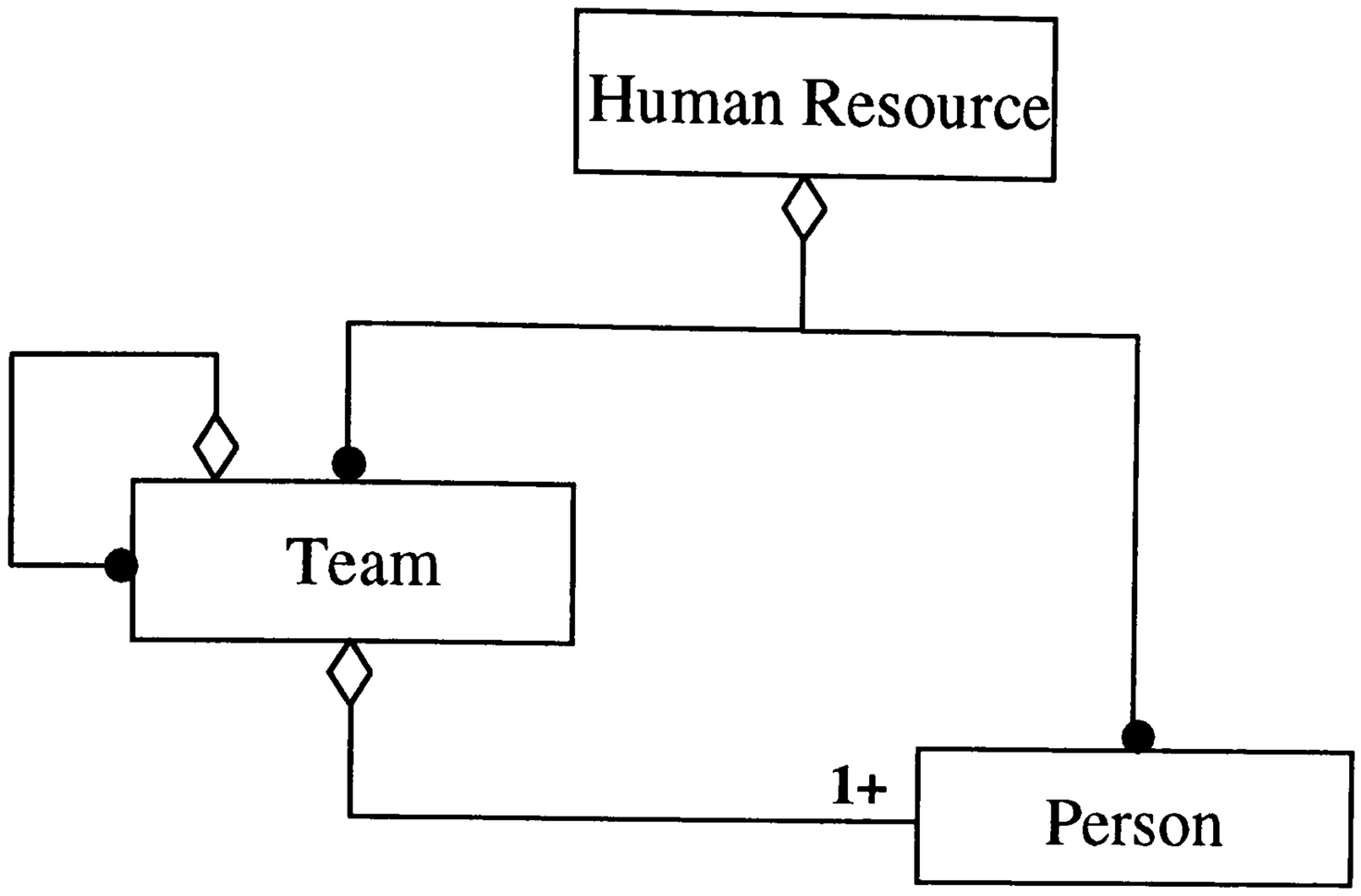


Figure 5.16 A representation of the resource structure

Thus, the resource structure involves both hierarchical (vertical i.e. top-down and bottom-up) and network (horizontal) structures making the representation of them within a single framework a complicated process. The overall organisation of the PI process and resources must be designed to encourage information flow in both vertical and horizontal directions necessary to achieve the overall objective of the PI process (Figure 5.17).

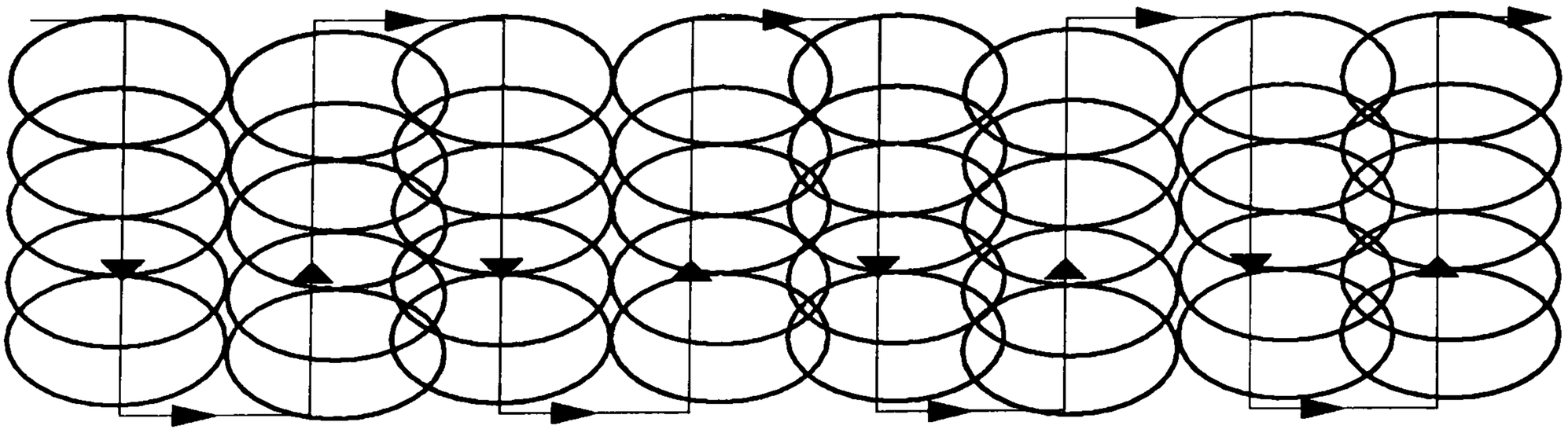


Figure 5.17 Overall information flow

Team building is an important activity in the PI process. The process model would represent the identified skills needed at various stages of the process (Table 5.3). The

relationship between the PI process and the required skills to perform the process can be represented by a separate class (Figure 5.18). The attribute *level* is an attribute of this relationship representing the level of expertise in the skill required to perform the process. For example, the resources required to analyse the blade for having the required weight includes a weight engineer with the skills of add up, read drawing and operate the design software. The skills required to design and develop the blade to have the required weight and required stress includes design, weight evaluation, stress analysis, aerodynamic analysis and leadership skills. The skills possessed by a person (Table 5.4) can be represented as an association between person and skill (Figure 5.19). The skills of the available resources would be compared against the required skills and the resources would be selected and allocated. The resources allocated to perform the process would have a role to play with respect to the process. The role, i.e. required grouping of skills to perform the activity, would be a relationship between the resource and the activity (Figure 5.20). In case of complex products, due to the significant duration of the PI project, the roles and the organisation of the team would change. This is due to the change in the direction of the PI project which in turn would change depending on the information generated by the activities of the PI project.

Table 5.3 An example for process, skills required, level and corresponding roles

Process	**Skills required**	**Required level in skill**	**Role**
Blade design and development	Design	2 years of experience	Designer
	Weight evaluation	0.5 years of experience	Weight evaluation engineer
	Aerodynamic analysis	1 year of experience	Aerodynamist
	Stress analysis	2 years of experience	Stress engineer
	Leadership	Trained	Team Leader

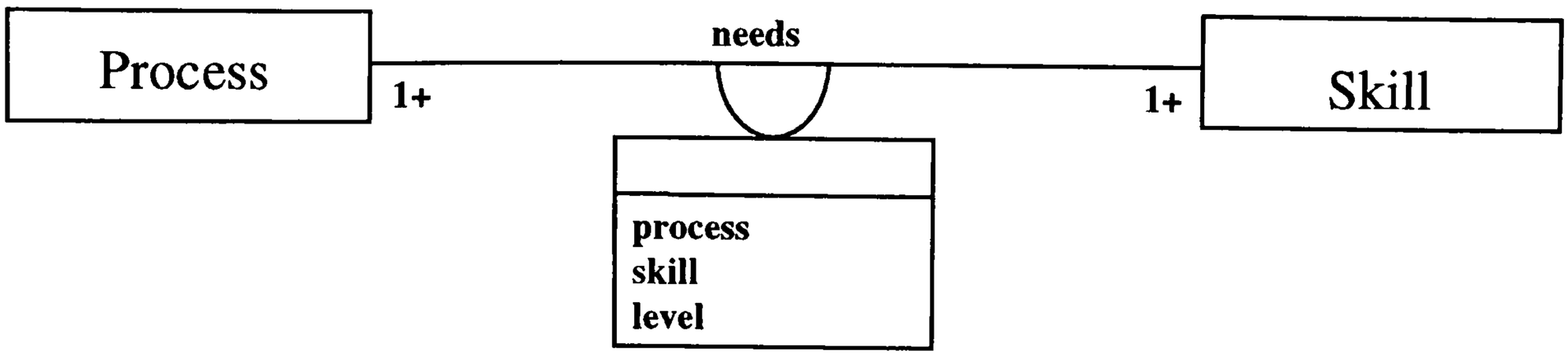


Figure 5.18 A representation of association between process and skill

Table 5.4 Examples of skills possessed by a person

Person	Skills possessed	Level of expertise in skill
Fred Shenhar	Design Weight evaluation	2 years of experience 1 year of experience
James Renier	Weight evaluation	3 years of experience
David Laufer	Aerodynamic analysis Leadership	1 year of experience 3 years of experience
Eves Thursfield	Stress analysis	2 years of experience
Peter Edwards	Stress analysis Leadership	2 years of experience Trained

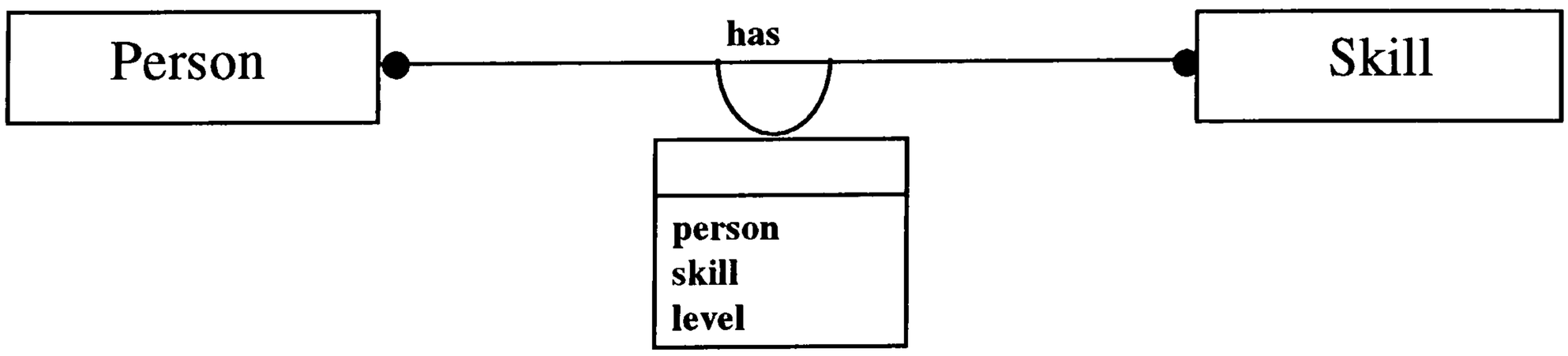


Figure 5.19 A representation of association between person and skill

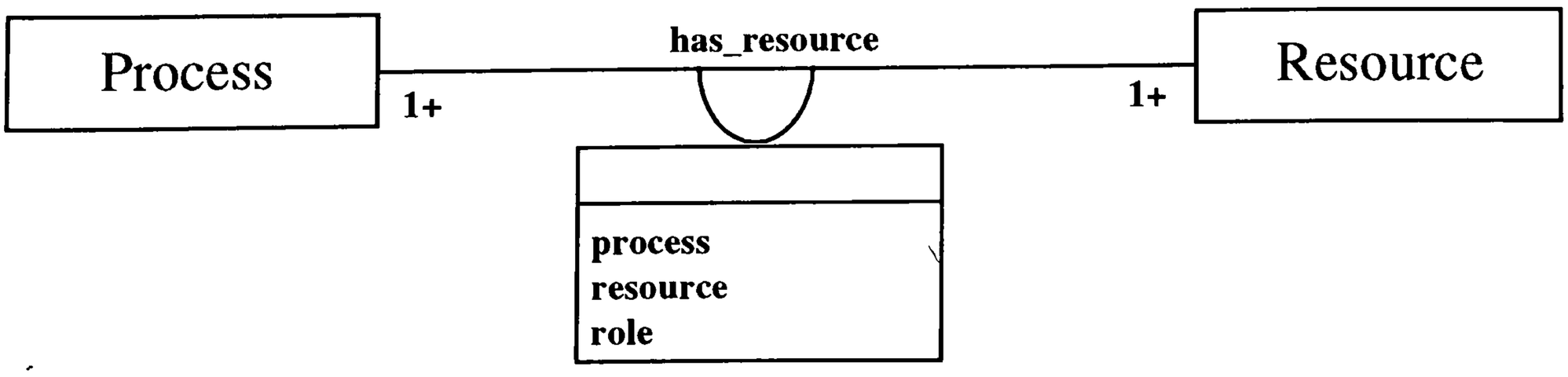


Figure 5.20 A representation of association between process and resource

5.2.4 Product

The PI process can be thought of as "product requirements specification and product definition integration" where the specifications are derived from the required product functionalities, and the product that is being defined is aimed to achieve the product functionalities. In order to achieve the product functionalities, overall product performance parameters would be set up, and the performance parameters would be broken down to sub system level and component level. These parameters are the attributes of the elements of the product. For example, 'weight' is a performance parameter of the product 'aeroengine', and it is also an attribute of the aeroengine and all its subsystems and components. The parameters evolve during the PI process and, hence, there would be addition, modification and deletion of attributes in the product information class at all levels. This would necessitate the evolution of the data structure that represents the product information. The relationship between product and product functionality would be represented by an attribute *why* in the data structure *product* (Appendix B.5).

The physical structure of the product is a description of the physical realisation of a product. A product (e.g. an aeroengine) could be decomposed into a set of assemblies (e.g. fan, compressor, turbine), each assembly could be decomposed into a set of sub-assemblies (e.g. stage1, disc, stage2) and each sub-assembly could be further decomposed into a set of components (e.g. locking blade, blade). The decomposition process can be continued until no part can be decomposed into smaller elements. An object model representation of the physical structure of the product is shown in Figure 5.21. The physical structure of a product, which is represented

generally using Bill of Materials, necessitates the following attributes : *name, description* for identification purposes, *has_parts, is_part_of* to represent the decomposition and aggregation relationships among the physical structure of the product. A link to the technical information such as drawings, production methods, and materials would be provided in the data structure *Product* (Appendix B.5).

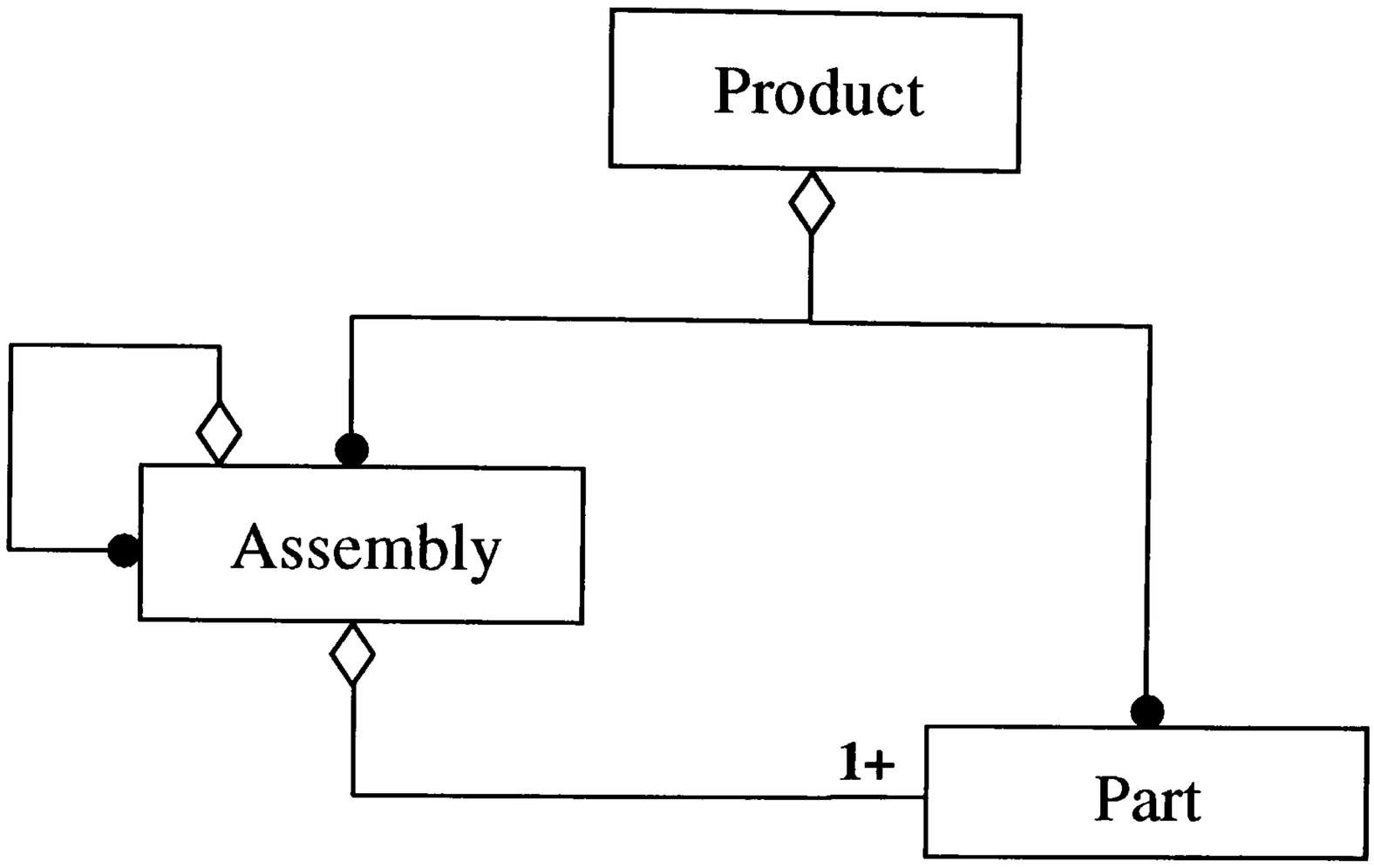


Figure 5.21 A representation of the physical structure of the product

5.3 RELATIONSHIPS

The product introduction process involves complex nested entities such as PI project model (Figure 5.22), product functionality tree, team structure and Bill of Materials (Figure 5.23). The relationships among these entities and within these entities take many forms. For illustration, if we consider the relationships associated with an activity of the product introduction project, they include relationship between the activity and the associated information (associated information relationships) and relationship between the activity and the project (aggregation relationship) in the project structure.

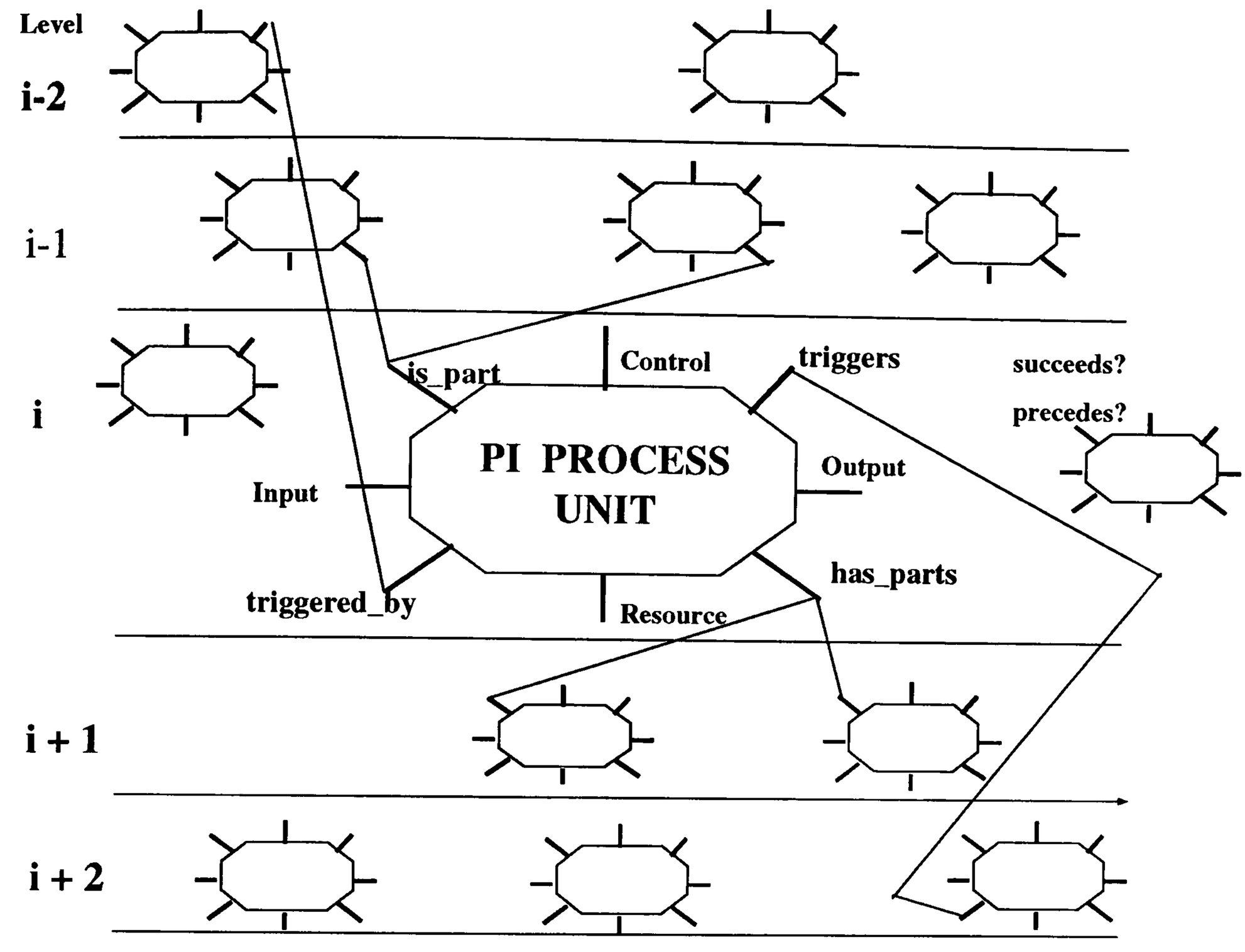


Figure 5.22 Complex associations in the PI process

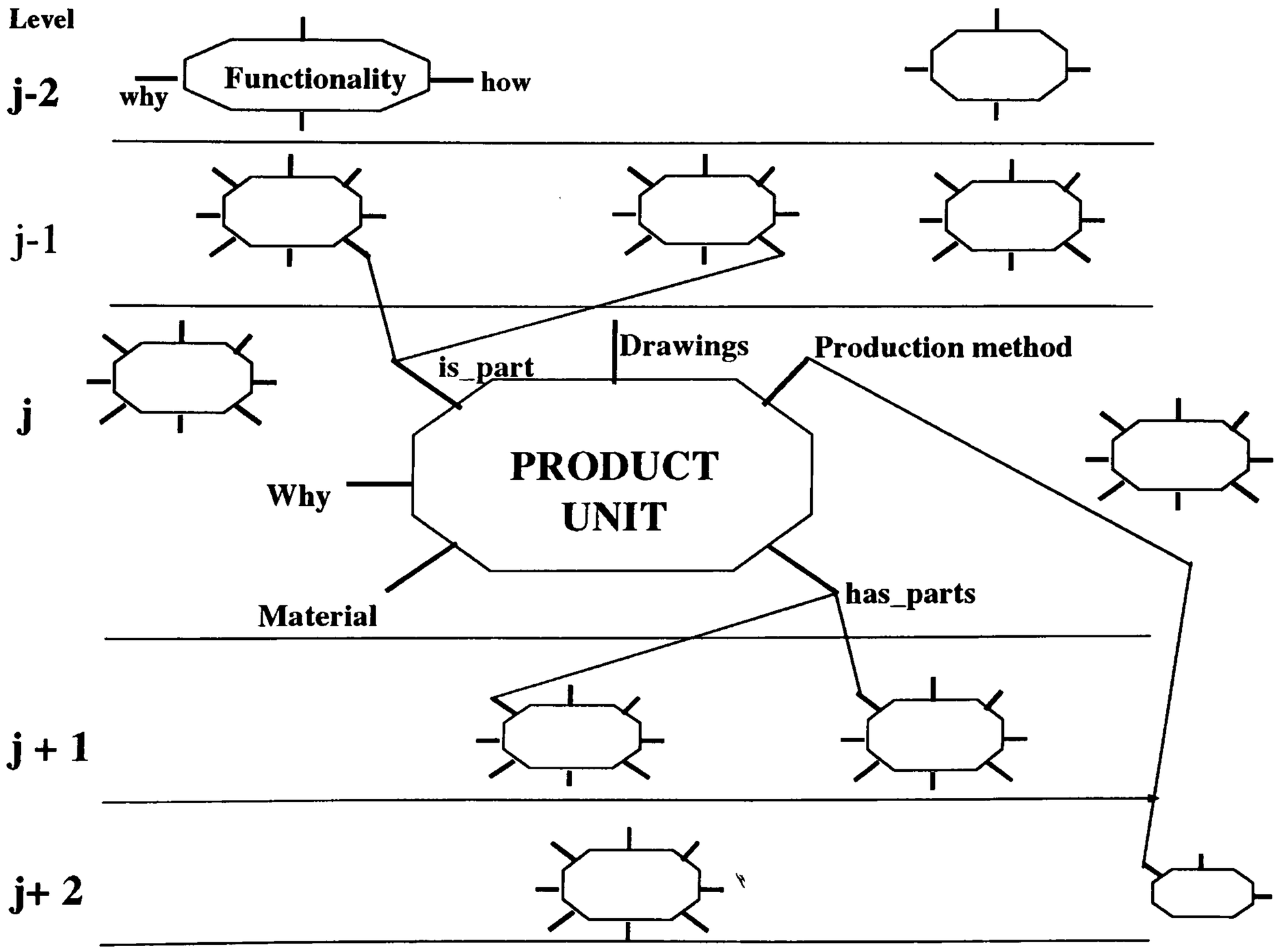


Figure 5.23 Complex associations in the product structure

The associated information relationships between the activity and the information on entities such as product, resource are of the type inter-entity relationships. In the project entity, the decomposition relationship between project and activity, and the aggregation relationship between activity and project are of the type intra-entity relationships.

Inter-entity relationships

In order to develop the reference model, it is necessary to clarify the forms of interdependencies existing between the elements of the product introduction process. For this purpose, the interdependencies have been modelled using the object modelling method (Figure 5.24). The complexity of the system can be best understood by the fact that all the relationships between the entities (or objects) are of the type 'many-to-many' and the relationships between the PI process and product information is not unique.

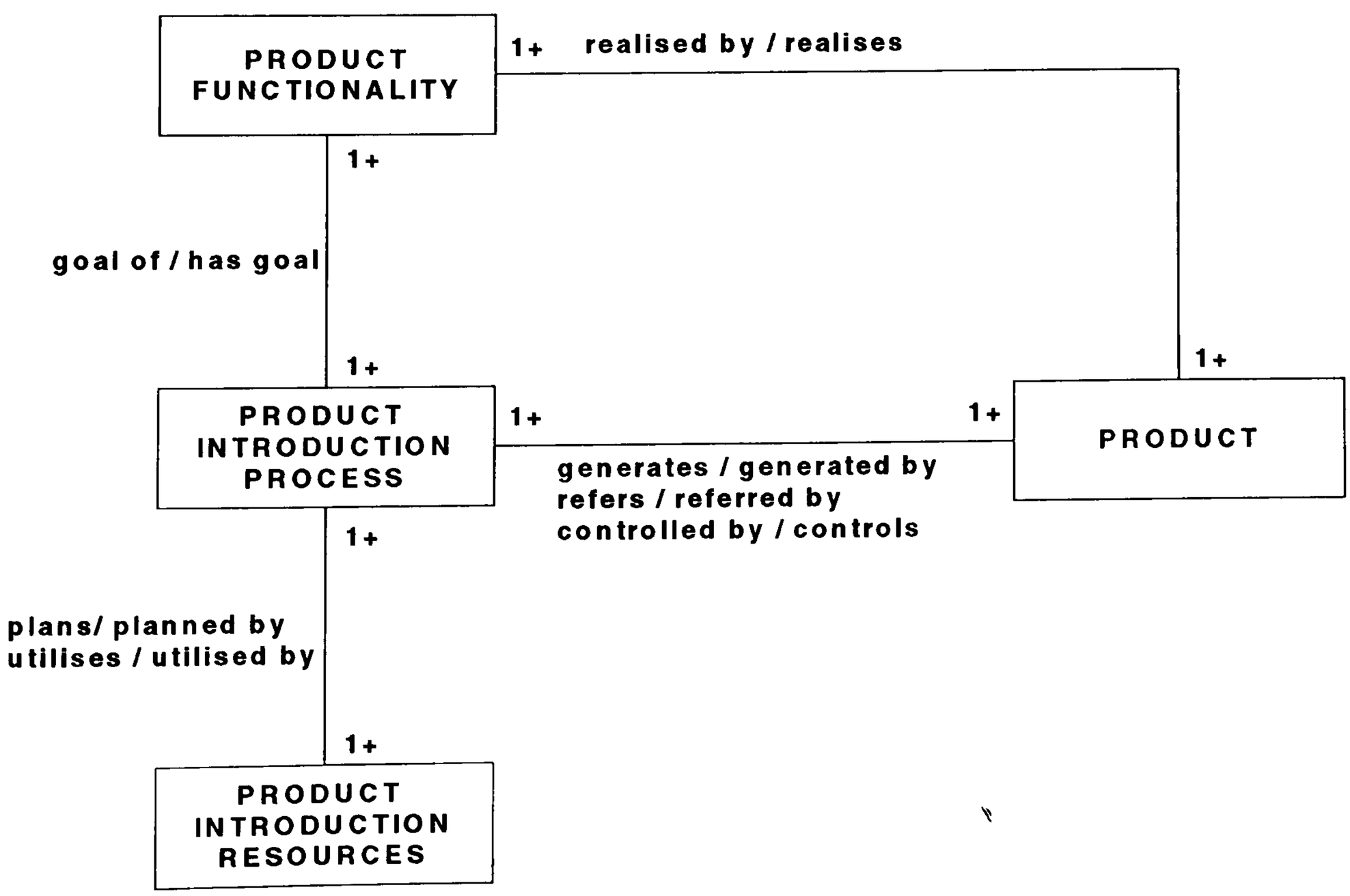


Figure 5.24 Object model of the product introduction process

Activity and its associated information - Information (product definition) is generated from the product introduction activities, and is also a control measure to control the process flow (via feedback / iteration) and an input (changes in product specification). Thus, an activity may use and generate product information which take many roles (input, output) in relation to the activity, depending on their usage. Figure 5.25 shows the general representation of an activity to analyse the dependencies among the PI process and information. The first activity in Figure 5.25 is an information transformation activity. Inputs to the activity are the information, materials etc. that are changed within the activity. Controls are the factors that constrain the activity. Means / resources are the people, tools, equipment that support the activity. Outputs are the results generated by the activity in the form of information. The inputs, controls, resources and outputs of the activities are information looked upon from different directions. Information generated from the decision-making activities (Figure 5.25) must be channelled to the intended destination to initiate control action.

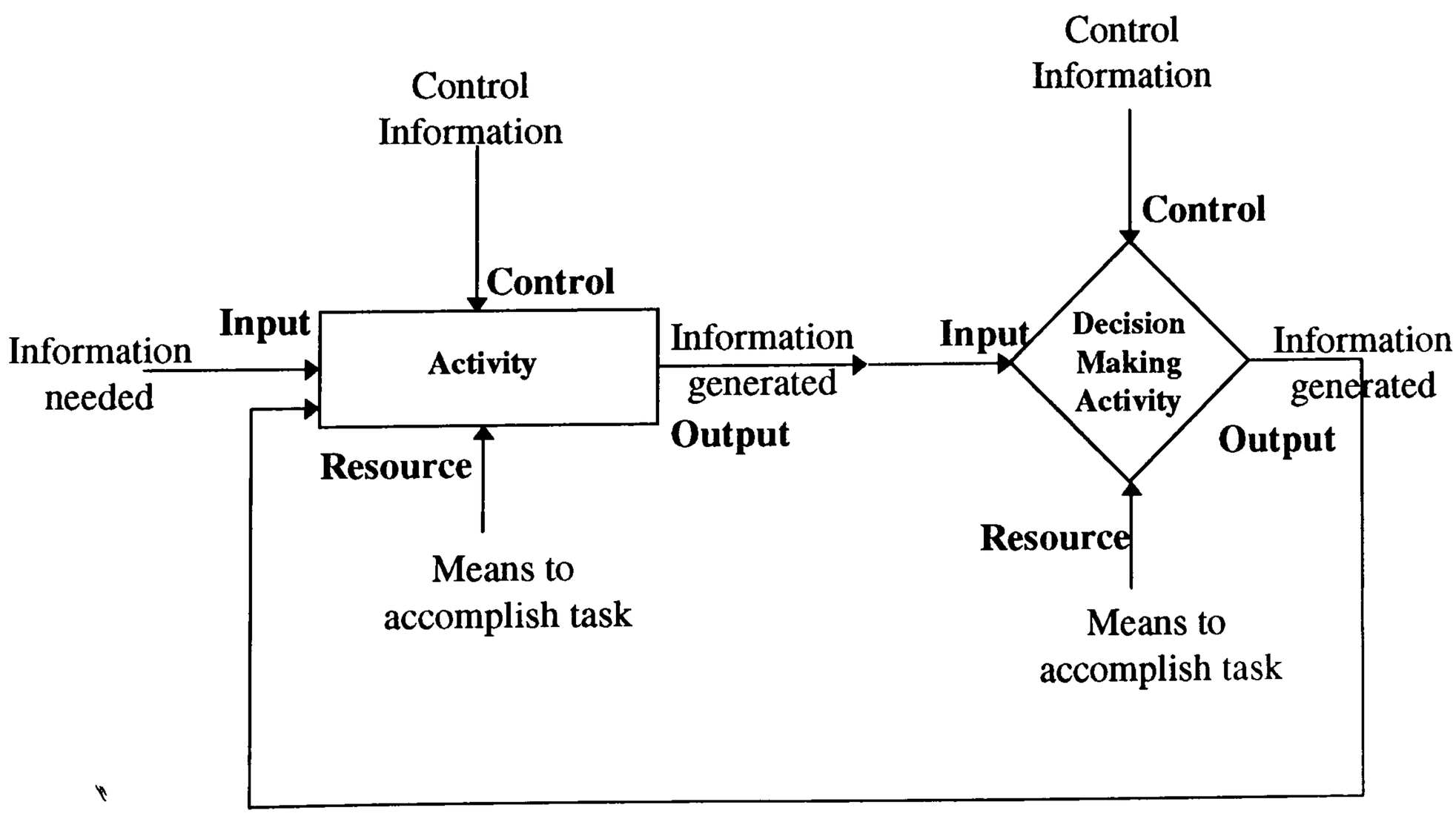


Figure 5.25 A general representation of activities of the PI process

For example, consider the activity 'weight evaluation'; the collection of input information includes blade geometry, material properties, manufacturing correction factor, that of output includes design weight, deviation, and that of goal includes required weight (Figure 5.26). The structure of this information as required by the activity is shown in Figure 5.27. Product information is structured in the form of bill-of-materials. Input to the activity is a collection of information taken from the product information database which is manipulated by the activity, and the activity generates a collection of information about the product.

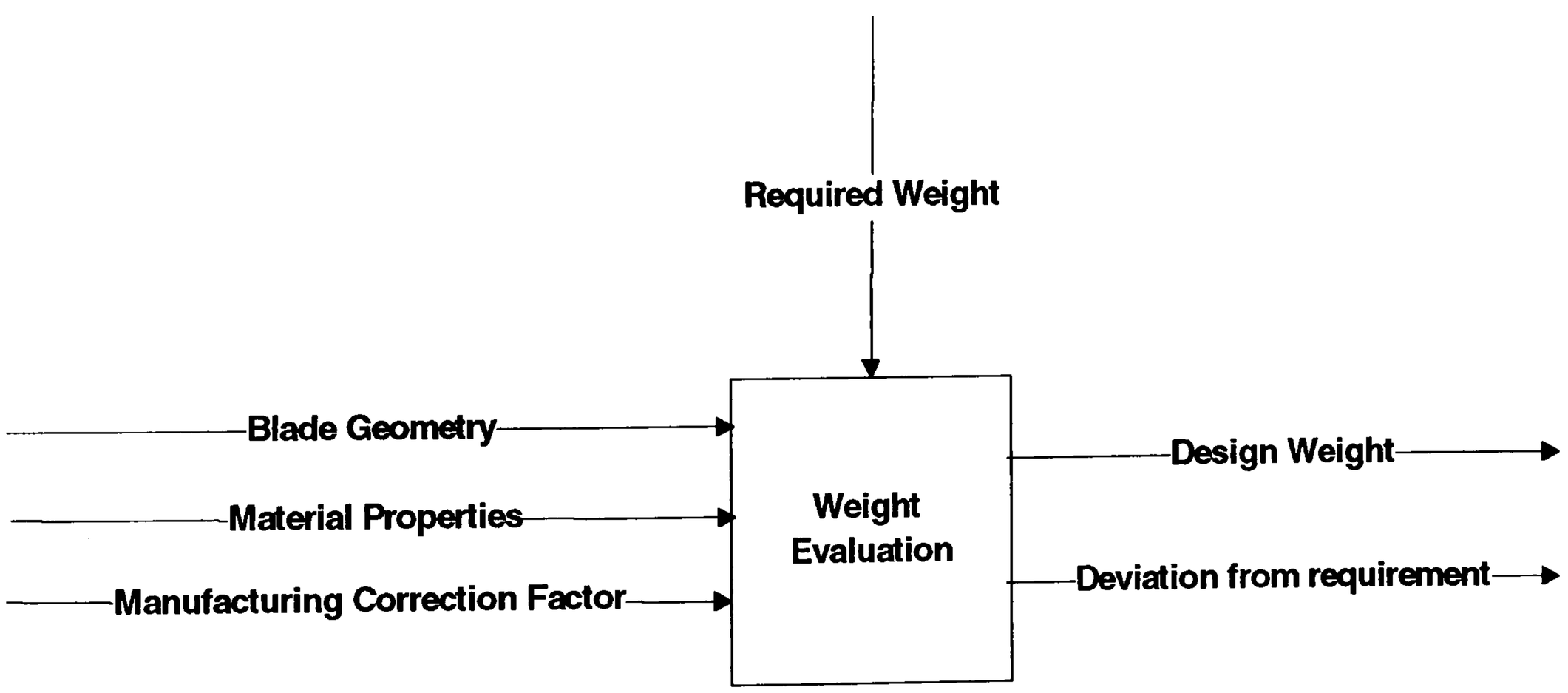


Figure 5.26 An example of an activity with its input, control and output

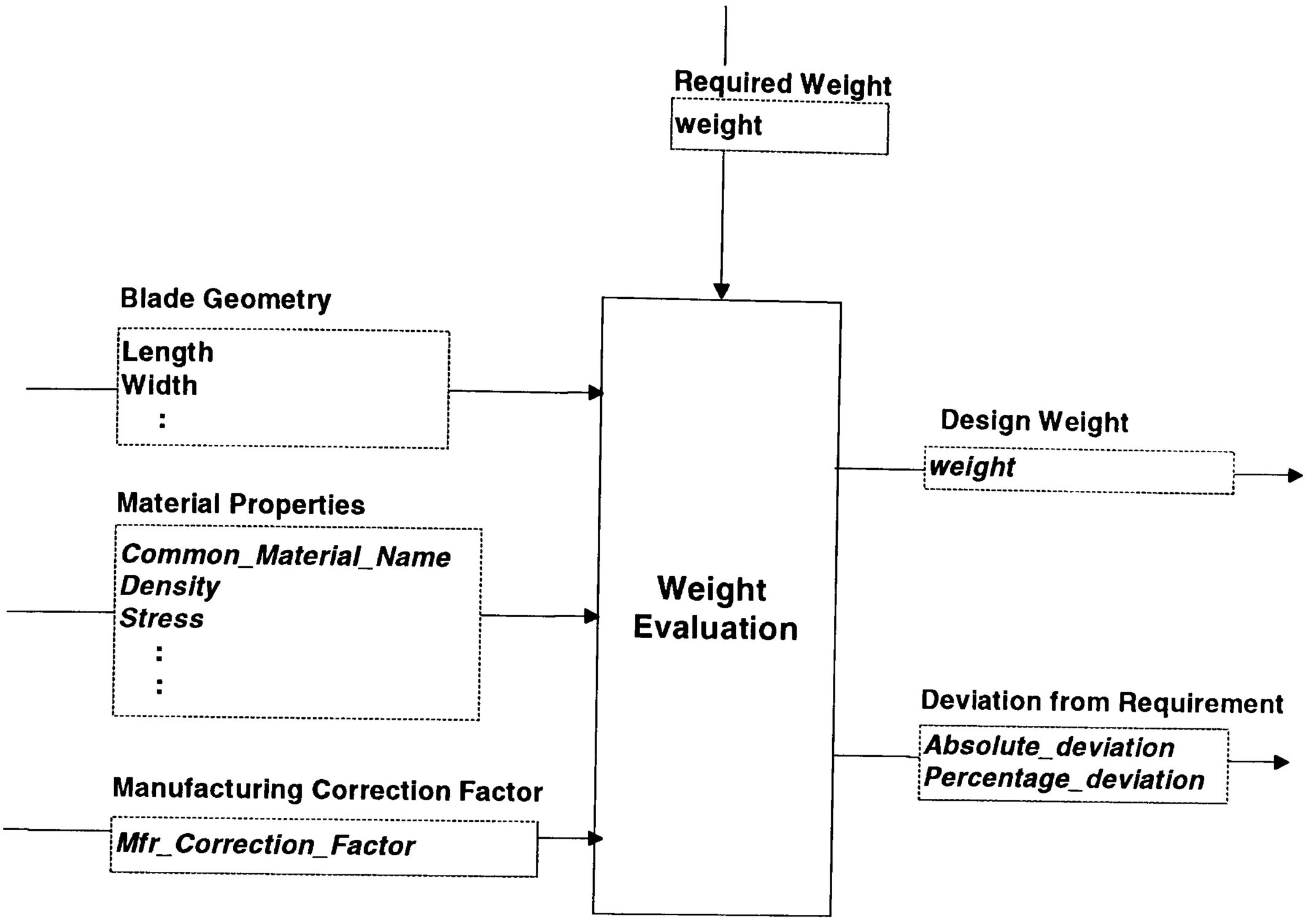


Figure 5.27 An example of an activity with the structure of its input, control and output information

If the collection of information that forms the input is available in the required structure, it can be copied and used by the activity. Most of the time, it needs to be collected before and by the activity; this process of collection involves the mapping between the structure in which the information is available at present and the structure in which it is required. For example, the input information 'blade geometry' is collected from the design drawing data of the blade, and material properties are collected from the material database. The activities of the PI process undergoes revisions and, hence, these associated information and their structures undergoes evolution. There should be some way of representing the information about the required structure (class name, attribute names (Figure 5.27) and their domains) and information as instances of it, of the database containing the information and its

structure, and of the relationships between them. Thus, the relationships between the activity and the associated information involves structures and instances intertwined making the representation of the relationship a complicated process.

Table 5.5 Attributes and sample values

Class name	**Attribute name**	**Sample attribute value**
Activity	*name*	'blade weight evaluation'
	description	'Compute the expected production weight from the CAD model'
	is_part_of	{*'design_blade'}
	input	**{*'Blade_geometry.2', *'Material_property.1',** *'Mfr_correction_factor.1'} // map from blade information
	control	{*'Required_weight.1'}
	output	{*'Designed_weight.2', *'Status_weight_deviation.2'} // map onto blade information
	duration	5
Blade_Geometry	*Length*	6
	Width	2.5

* denotes a pointer to an object.

Table 5.6 Domain of the attributes

Class name	**Attribute name**	**Domain**
Activity	*name*	Text
	description	Text
	is_part_of	Set_of Activity_Project
	input	Set_of Heterogeneous Structures and Instance Ids
	control	Set_of Heterogeneous Structures and Instance Ids
	output	Set_of Heterogeneous Structures and Instance Ids
	duration	Numeric
Blade_Geometry	*Length*	Numeric
	Width	Numeric

Table 5.7 Blade_geometry

Instance Id	**Length**	**Width**
1	5	2
2	***6***	***2.5***

Table 5.8 Material_property

Instance Id	Name	Density	UTS	Proof_Stress
1	*Titanium*	*4000*	*1000*	*350*

It can be seen from the attribute *input* in Table 5.5, and Tables 5.7 and 5.8 that the structures belonging to the input set are not of the same type having the same attributes. For example, *'*Blade_geometry.2'* points to the instance with instance id 2 of the *Blade_geometry* class (Table 5.7); and *'*Material_property.1'* points to the instance with instance id 1 of the *Material_property* class (Table 5.8). Thus, the *input* is a set of pointers pointing to the classes, and instance identification codes pointing to the instances of the respective classes (Table 5.6). The relationships between an activity and the associated product information that include the above mentioned heterogeneous information structures can be represented using a concept called information mapping (Figure 5.28). The entity 'information' in Figure 5.28 represents the product information consisting of information on the physical structure of the product, product functionalities, drawings, etc. The data structures that would be necessary for associating the process with the information maps are shown in Appendix B.2 and Appendix B.6.

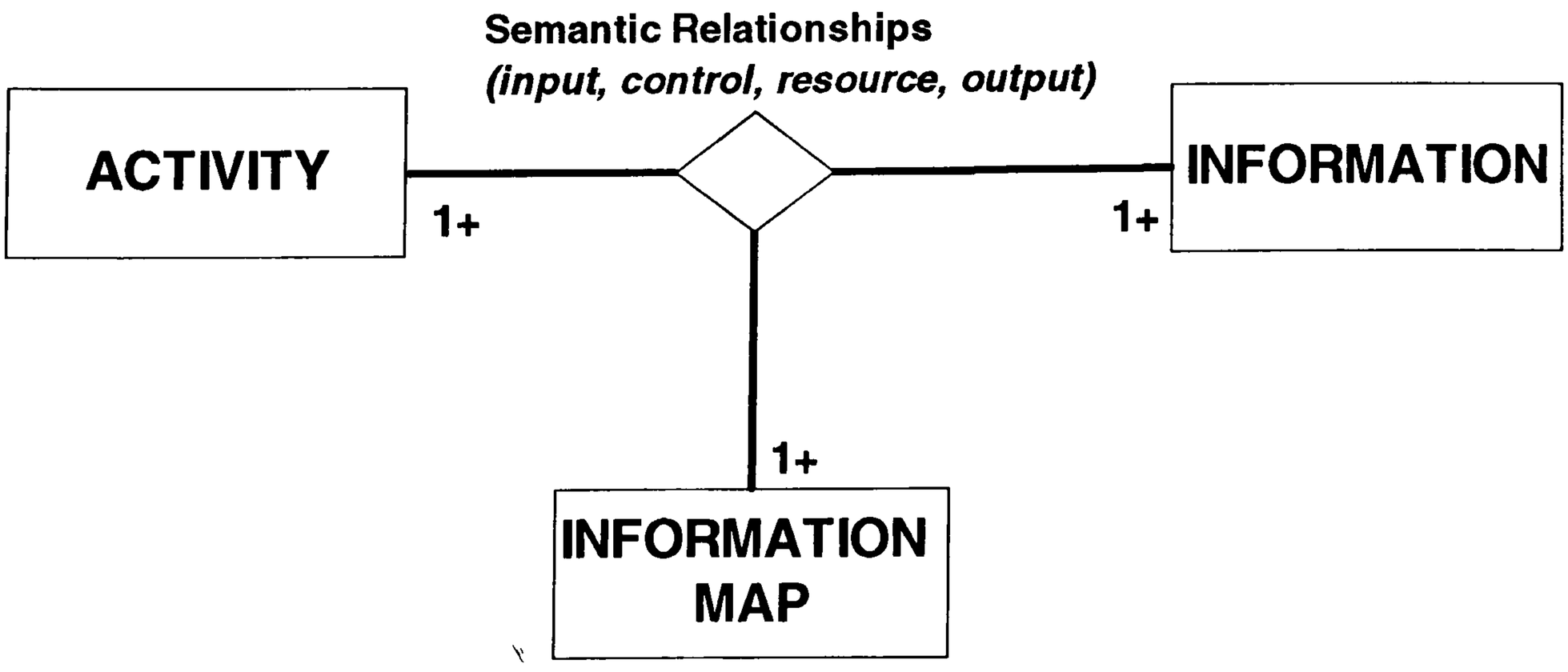


Figure 5.28 Relationships between activity and product information and its representation

As the product introduction process is a problem solving process, it involves enormous testing and iteration, thus following Test Operate Test Exit (TOTE) procedures. This set of heterogeneous information structures evolve with time and require an early transmission of information to following activities and a feedback of information from the test activities to the necessary activities within the process chain of product introduction.

The managerial process does the resource planning for the technical process necessary to develop the product that meets the required functionalities. The resource information is an output at the managerial level of the product introduction process, and is resource information at the technical level of the product introduction process. Hence, the relationship between process and resource information can be argued in a similar way and represented using information maps.

Table 5.9 Semantic relationships between activity and information

Class name	**Attribute name**	**Domain**
Activity	*name*	Text
	description	Text
	is_part_of	Set_of Activity_Project
	input	**Set_of Activity_InformationMap**
	control	**Set_of Activity_InformationMap**
	resource	**Set_of Activity_InformationMap**
	output	**Set_of Activity_InformationMap**
	duration	Numeric
Blade_geometry	*Length*	Numeric
	Width	Numeric

Thus, the semantic associations of input, control, resource and output between the activity and information can be represented by collections of associations between the activity and information maps (Table 5.9) where an information map class is a mapping

defined between the required information structure (as required by the activity) and the available information structure (as available in the product information model). An information map is simply an instance of the information map class.

Intra-entity relationships - From a problem solving perspective, initially the problem of achieving product functionalities is decomposed into smaller sub-problems at various levels; once the individual solutions of these sub-problems are obtained, the sub-solutions at the lower levels must be aggregated to reach the overall solution. In this process, decomposition of the entities of the PI process (overall functionality into sub-functionalities, project into activities, workforce into teams and then to individuals, product into assemblies and then parts) would take place and result in a hierarchical structure. The planning of the PI process should be top-down starting from the product functionalities, and working down through the workflow, but its implementation should be bottom-up, starting from the downstream activities and working up towards the project goal "achieving the product functionalities" (Figure 5.29). From a management perspective, the product introduction process of a complex product can be viewed as a large scale multi-level project network management problem. An information map at a higher level could contain pointers referencing lower level information maps, thus forming an information map hierarchy. For example, an output (information map) of a higher level activity could be a report aggregating the outputs (information maps) of the lower level activities. Thus, each of the entities (product functionality, PI project, PI resources, product - Bill of Materials, and information map) follows a hierarchical organisation that can be represented as shown in Figure 5.29. It can be observed that three factors would become necessary

to represent the models from intra-entity relationships perspective. Thus, the factors that contribute to the hierarchy of network structures of each of these elements can be generalised into: node, relationship among nodes, and the organisation of nodes and relationships (Figure 5.29) at various levels.

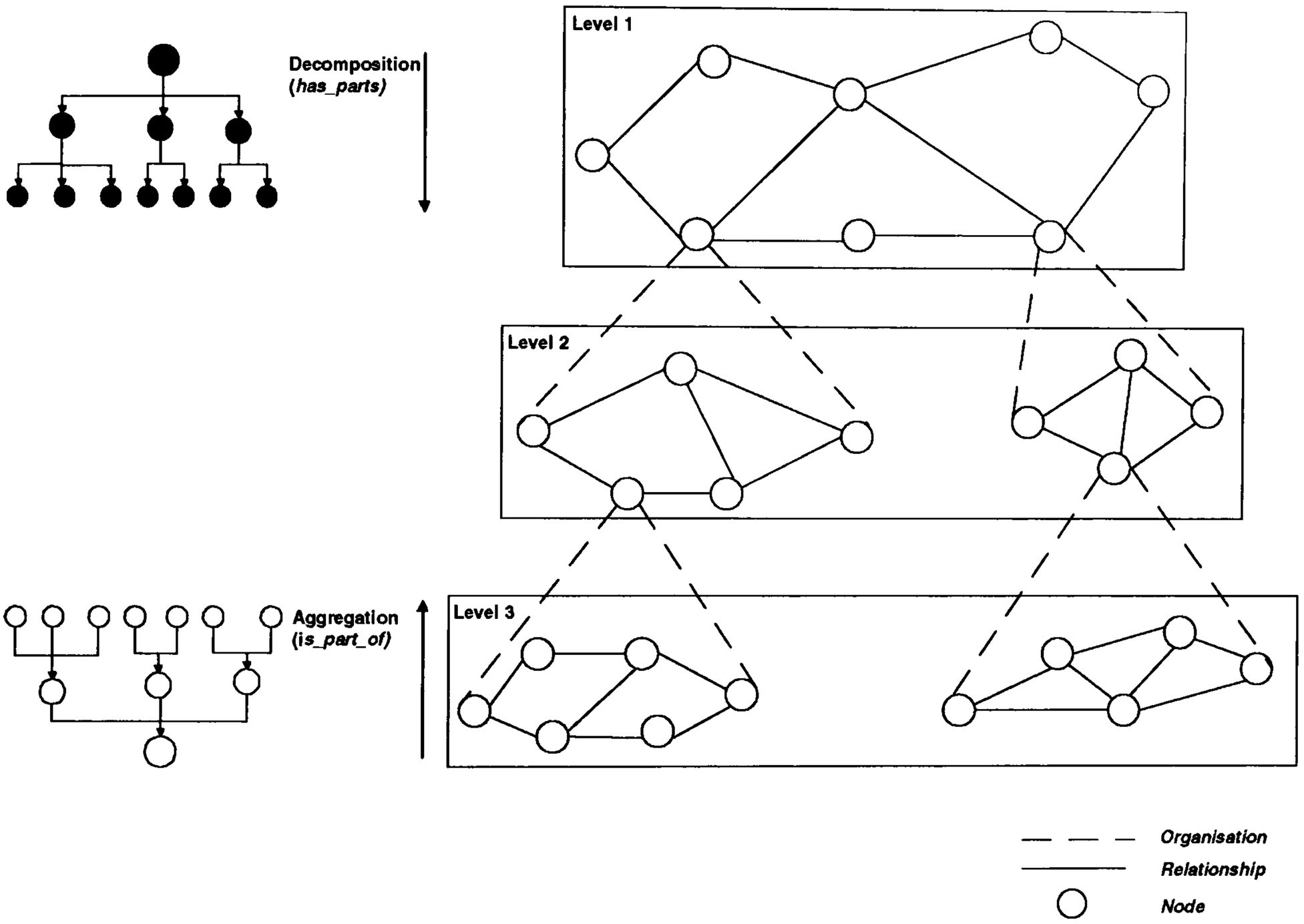


Figure 5.29 Entity structure (hierarchy of networks)

The decomposition relationship between the object that is decomposed (e.g. project at the higher level) and the unit that is obtained due to decomposition (e.g. activity at the lower level) would be represented in the 'organisation' layer using the attribute *has_parts* defined in classes *product_functionality, pi_process, pi_resource* and *product.* When the sub_solutions at the lower level need to be aggregated to reach the overall solution at the higher level, it would be necessary to know the higher level

(problem) node the sub-solution contributes to or has to be sent to for aggregation. Thus, it becomes necessary to represent the aggregation relationships between the nodes at the lower level and the nodes at the higher level, and this relationship would be represented in the 'organisation' layer by an attribute *is_part_of* that would be defined in classes *functionality, activity, person* and *part.*

Within each level of the hierarchical structure, the relationships, such as precedence relationship based on the relative start date among activities and means-end relationship in product functionality analysis, lead to a network structure. Such relationships among the entities at the same level of hierarchy of each of the entity models could be represented by the 'Relationship' layer. The nodes of these individual networks (e.g. an activity in the project network, a part in the product network, a functionality in the product functionality network, and a person in the resource network) form the basic elements which when aggregated provide the higher level nodes; these nodes would be represented by the 'Node' layer. Thus, a node corresponding to the product functionality data model would represent 'a functionality'; that of project data model would represent 'an activity' or 'a project'; that of resource data model would represent 'a person' or 'a team'; and that of product data model would represent 'a part' or 'sub-assembly' or 'a product' depending on the level of decomposition. Corresponding to the nodes at the lowest level, the classes *'functionality', activity', 'person'* and *'part'* need to be defined.

5.4 MODEL INTEGRATION

The overall information model would include the following sub models : PI process model, product functionality model, PI resource model, product model and information

map model. Each of these models when drawn with all possible relationships follows a network architecture. The intra-entity integration or intra-model integration i.e. the interdependencies of elements within a model would form a dimension of the information modelling. This intra-layer integration ultimately leads to a "hierarchy of network" architecture.

As the PI information model includes sub-models and is large, structured integration is followed. Structured integration is concerned with ① discovering general concepts underlying the sub-models and ② developing interfaces between the sub-models. Information map model would be the interface integrating the entities of the PI process management (product functionality, PI process, PI resource and product). From an intra-entity perspective, each of the models leads to a hierarchy of network architecture. The necessary interfaces (*has_parts, is_part_of*) would be defined to represent the hierarchical relationships among the various levels within an entity model. In order to integrate the above mentioned submodels, it would be necessary to define a meta-model. The meta-model would provide the information about the sub-information-models by defining ① the information about the structure of the entities, relationships stored in each of the sub-model and ② information about the relationship between the sub-models. In order to achieve the first step, it is necessary to define the classes MetaDatabase, MetaClass, MetaAttribute, MetaAttributetype in the meta-model (Appendix B.7). In order to achieve the second step, it would be necessary to define the dependencies among the classes and the attributes of the databases involved.

5.5 DISCUSSION

In this chapter, the interdependencies between the elements of the product introduction process have been analysed. Information (product definition) is generated from the product introduction activities, and is also a control measure to control the process flow (via feedback / iteration) and an input (changes in product specification). The link between the managerial and technical information is essential as it is the output of the activities, i.e. technical information that controls the decisions in managing the project. As the lead time and project cost are information related to the activities of the PI project, and product cost and product functionality are information related to the product, i.e. results of the activities, the design of an integrated information model would be based on linking the activity plan with the results of the activity using information maps. Apart from this, concurrent management of project time, cost and project objectives (product functionalities) necessitates this link. But, the representation of the link places a challenging and novel set of demands on database technology. These involve ① dynamic evolution of the conceptual schema to support the evolution of information, ② development of data representation techniques to represent information maps, and ③ linking the conceptual schemata behind the distributed and heterogeneous databases to present the relevant information to the activities of the PI process.

Dynamic data modelling - The information generated by the product introduction process has a special characteristic of moving from abstract to concrete state in steps when the process moves from initial to the final stage. This information includes the information on physical structure of the product, functional structure of the product,

resource structure, product (drawings, production method, materials), product functionality, resources, relationships among these. In order to capture the evolving structural and instances information that evolves out of the dynamic product introduction process, the information model for the product introduction should evolve. Towards this end, a meta-model concept that supports evolution of the data model needs to be used (Figure 5.30).

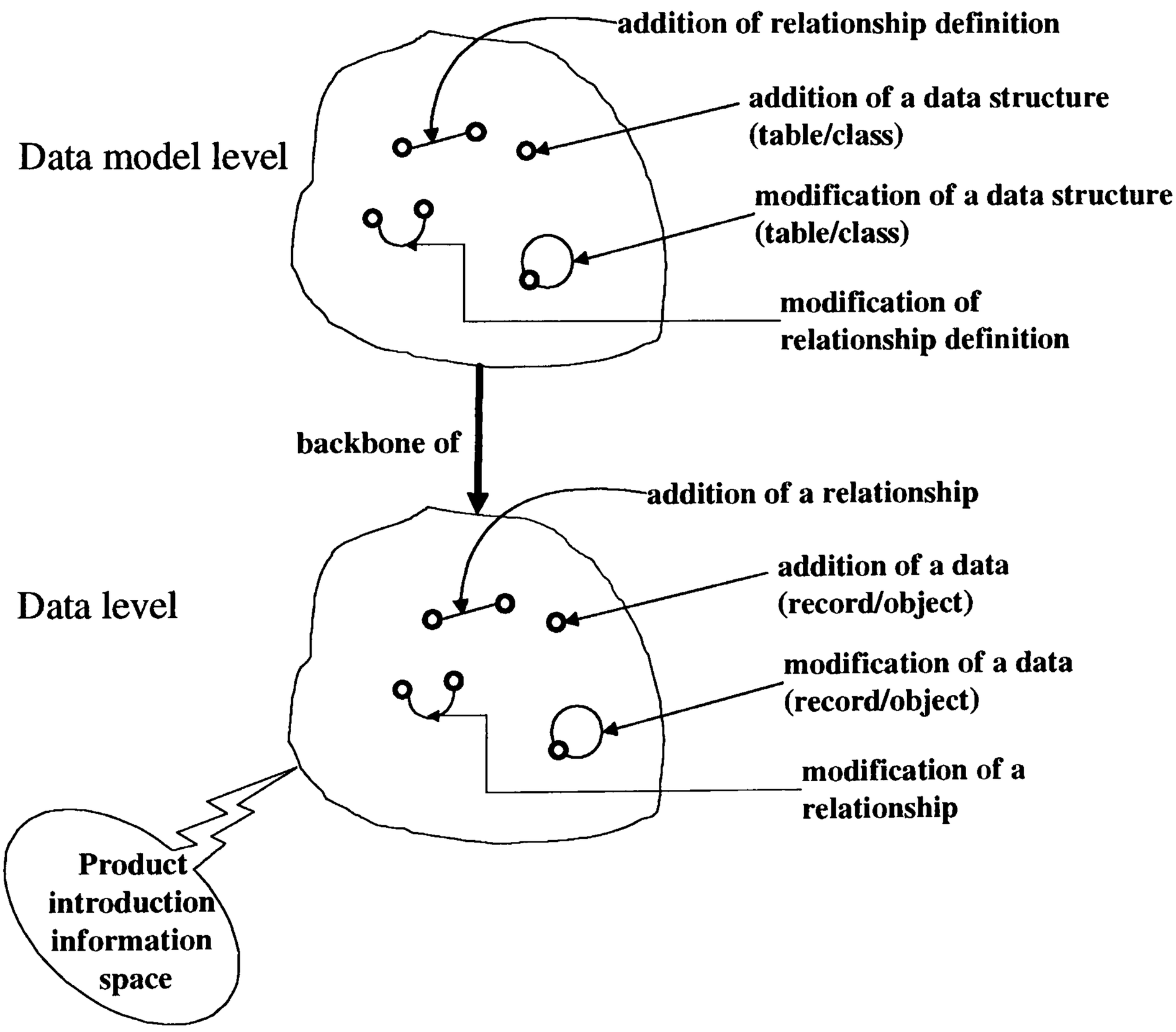


Figure 5.30 Addition, deletion and modification of a concept in the product introduction information space

The special features of information associated with the product introduction activities are :- the structural (meta-) data is dynamic and is intertwined with repetitive (instance-) data, and the amount of structural information is large compared to the size of the information content. In existing database systems, there is a strict borderline between structural data (classes) and repetitive data (objects). Instantiation is used to link the class and the instance levels. An extension to the instantiation concept, to consider classes as objects would become necessary to represent the data that is used both at schema level and data level within a single framework. This would necessitate ① defining a *meta_class* that represents the structure of the data structures, i.e. classes and ② providing the dynamic routines to handle the creation of a class, maintenance of the definition of a class (adding an attribute, modifying an attribute, deleting an attribute) and relationships among classes. Thus, the 'meta-model' concept becomes essential in designing the required integrated information model.

Development of data representation techniques - The structural aspects of the various entities involved in the product introduction process, and the complex associations among them demands a uniform framework for the treatment of arbitrary user-defined data types such as *'set_of Activities'* and *'set_of Activity_InformationMap'*. Representation mechanisms of such data becomes necessary for the development of the information model.

Linking the heterogeneous structures - the representation of the product introduction information calls for integrating heterogeneous structures, which is a complex process. The decomposition-aggregation relationship that exists in the vertical dimension in the product introduction project is used to define the *flow of*

control relationship between modules at different levels of abstraction and it is of hierarchical type. The information dependency relationship (input, constraint and output) that exists in the horizontal dimension in the product introduction project is used to define the *flow of information* relationship between modules at the same level of abstraction, or between modules which are not related by a direct flow of control from one to the other, and it is of network type. The representation of these heterogeneous structures within a single framework is a challenging task.

Linking the distributed, heterogeneous databases - The product introduction process may be thought of as a process generating a collection of various types of interrelated semantic models and their eventual instantiation and population. The semantic models represent product introduction entity networks (goals, activities, resources, outcomes). The information mapping is a mechanism proposed that would allow access of the information represented in these models by defining the appropriate mappings between the usage (input, control or constraints, resource, output) of the information and the description of information. The information accessed by an activity of the PI process is stored in distributed, heterogeneous databases where heterogeneous implies that each local database may be managed by a distinct database management system. An essential feature of information mapping would be to enable the access of information from several proprietary formats (databases) and bring them into the required format. This necessitates the storage of information on databases and what each database represents. Thus, a class called *meta-database* would become essential in the meta-model schema. A data translation needs to be done in the background which the user will not be aware of and that will make the information access more transparent.

Dimensions of information modelling - A detailed analysis of the product introduction information resulted in the identification of the three modelling dimensions for product introduction information. They are entity-relationship of the product introduction information, elements that lead to the intra-layer integration and evolution (Figure 5.31). Considering the entity-relationship details of the product introduction information as the first dimension of the information model, the following five layers are identified:- ① PI process that generates / uses information, ② product functionality which is the goal of the product introduction process, ③ product introduction resources that carry out / support the PI process, ④ product information generated / used by the PI process and ⑤ information map to represent the semantic relationships between the above layers. In the intra-layer integration dimension, three layers - node, relationships and organisation are identified, and in the evolution dimension - meta-model, data model and data are identified. The meta-model describes the structure of the data model. In the proposed integrated information model, the meta-model would incorporate and unify the following data models:- PI process data model, product functionality data model, PI resource data model, product data model and information map data model. These data models describe the schemata of the following databases :- PI process database, product functionality database, PI resource database, product database and information map database.

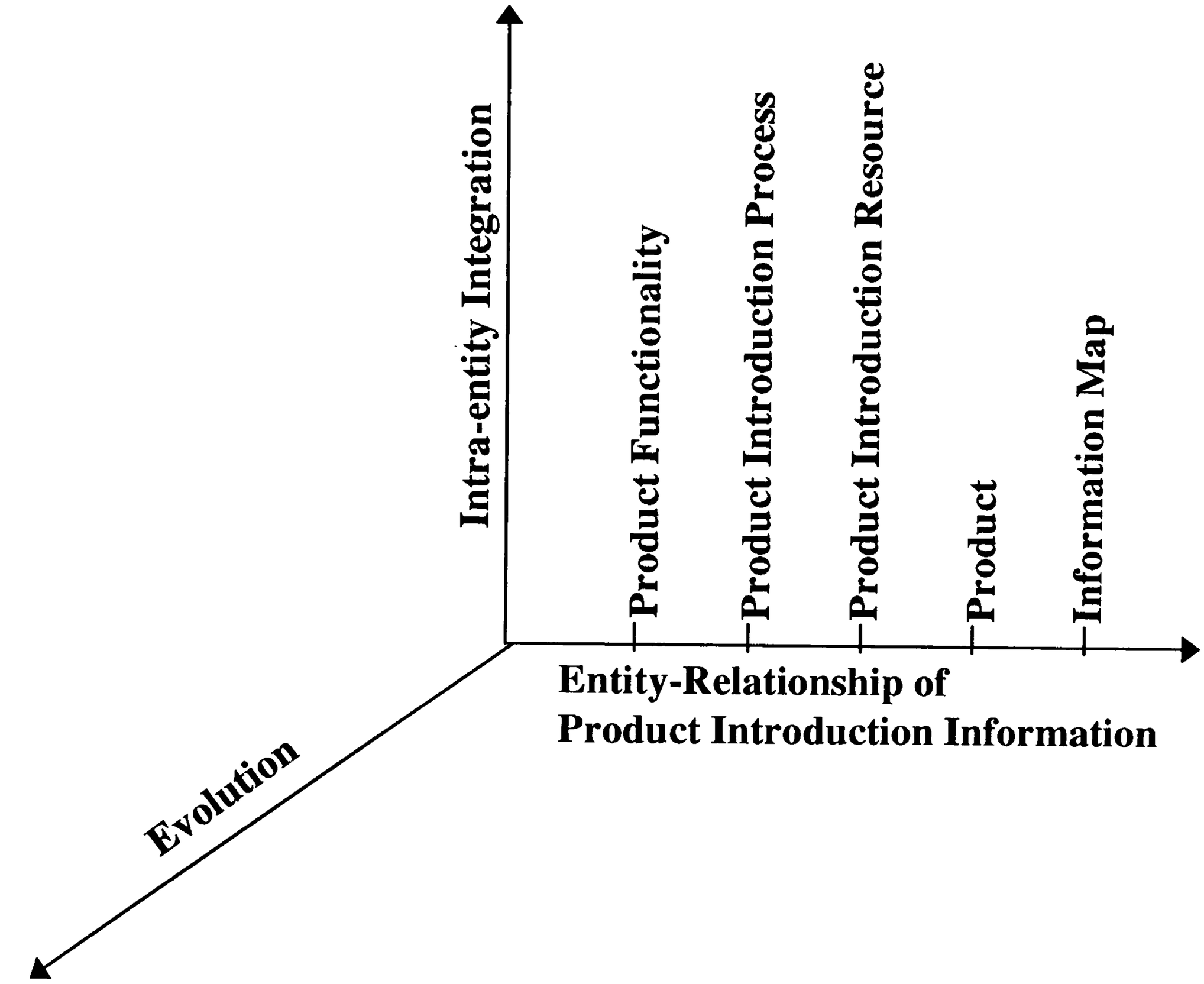


Figure 5.31 The dimensions of information modelling

Requirements of information management - No single representation can address the representational needs of the information relevant to the product introduction process; this can only be satisfied by a collection of representations that are related to each other. Information is the "glue" that connects the elements. It has been found that the following models need to be designed and developed in order to support the management of the product introduction process (Thirupathi and Roy 1997b):

1. a model of goals (product functionalities) and goal networks to represent the goals of the product introduction process
2. a model of activities and activity networks to configure and manage the product introduction process

3. a model of data objects of varied representations: text, drawings, databases, etc. (to represent product information)
4. a model of human and computational agents to represent the resource structure in the product introduction process
5. an information model that integrates the above models based on the definition of semantic relationships between the models; these semantic relationships can be represented using information maps. The link between the process and the data is shown in Figure 5.32.
6. a model management system that allows for continual evolution of the integrated information model mentioned in the previous step, and
7. a data management system that will take care of the evolving data that is generated by instantiating the information model.

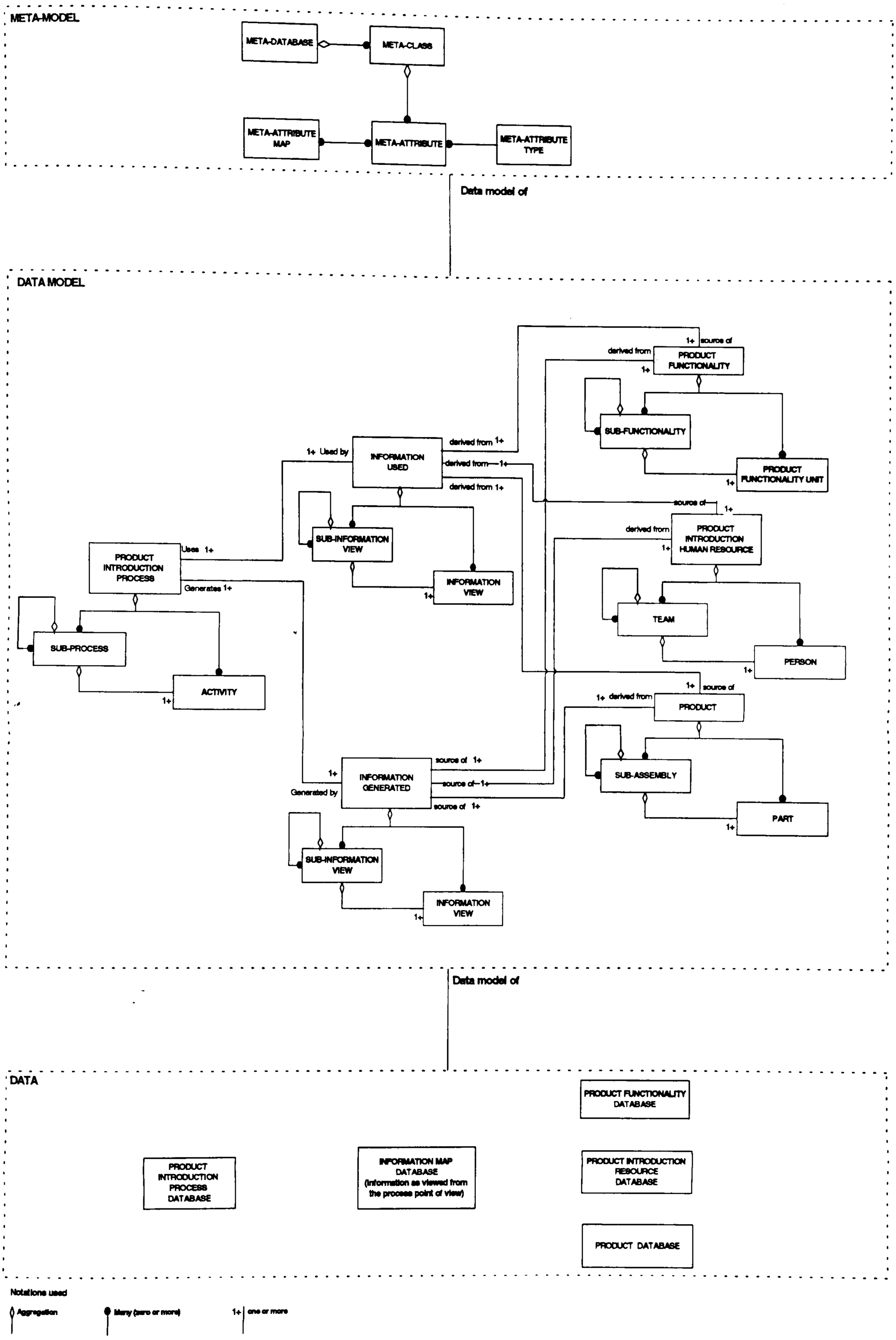


Figure 5.32 Link between the process and the data

CHAPTER 6

6. INTEGRATED INFORMATION MODEL

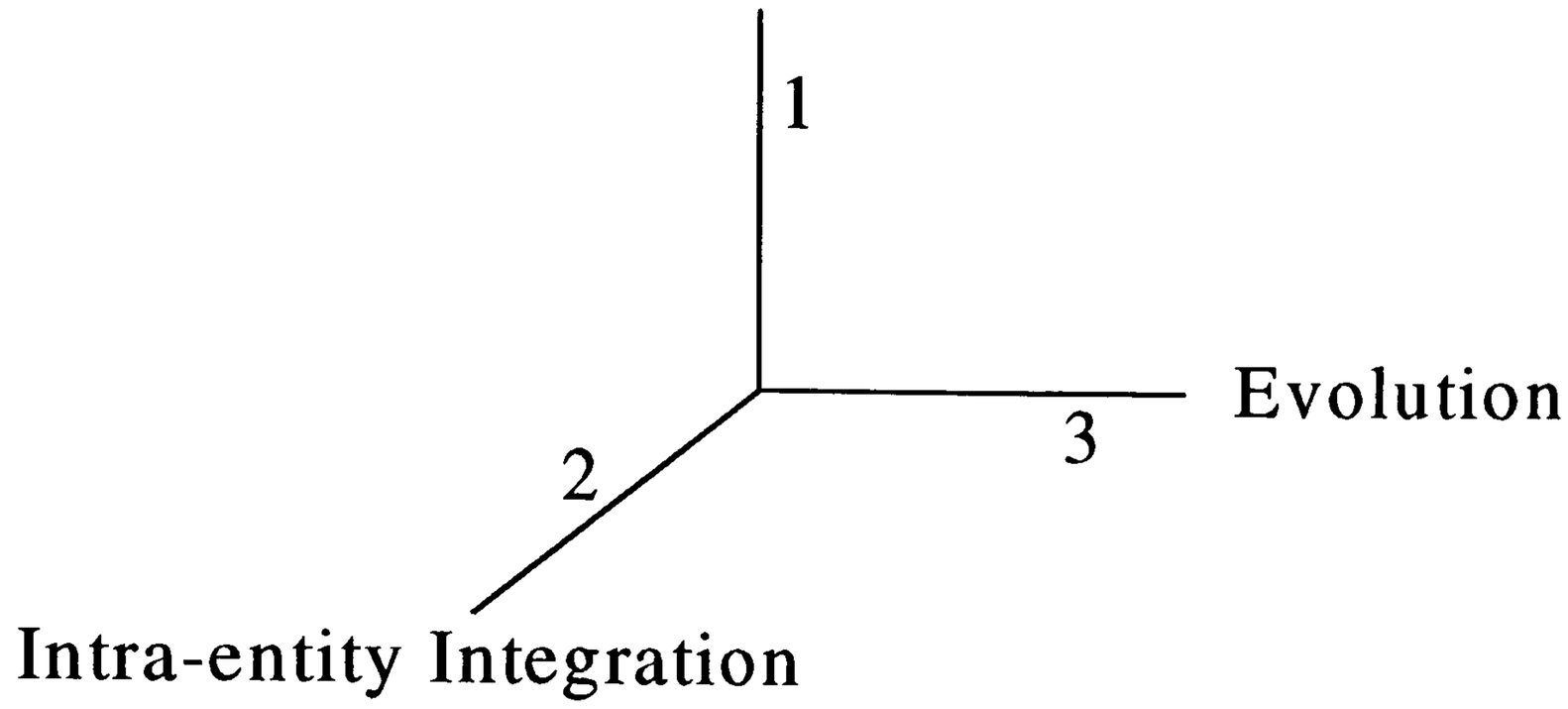
6.1 INTRODUCTION

Analysis of the product introduction information has recognised three dimensions for product introduction information modelling. The focus of this chapter is the architecture of the information model designed using these dimensions. First, an overview of the architecture of the information model is presented. Then the data structures that constitute its building blocks are detailed. Finally, the important characteristics of the architecture are highlighted.

6.2 ARCHITECTURE OF THE INFORMATION MODEL

The architecture shown in Figure 6.1 is based on three modelling dimensions - entity-relationship of the product introduction information, intra-entity integration resulting in a network architecture, and evolution. In connection with the entity-relationship of the product introduction information, there are five layers representing the product functionality, product introduction process, product introduction resource, product and information map. Information maps are the "glue" of this architecture, providing semantic access paths between the entities.

Entity-Relationship of Product Introduction Information



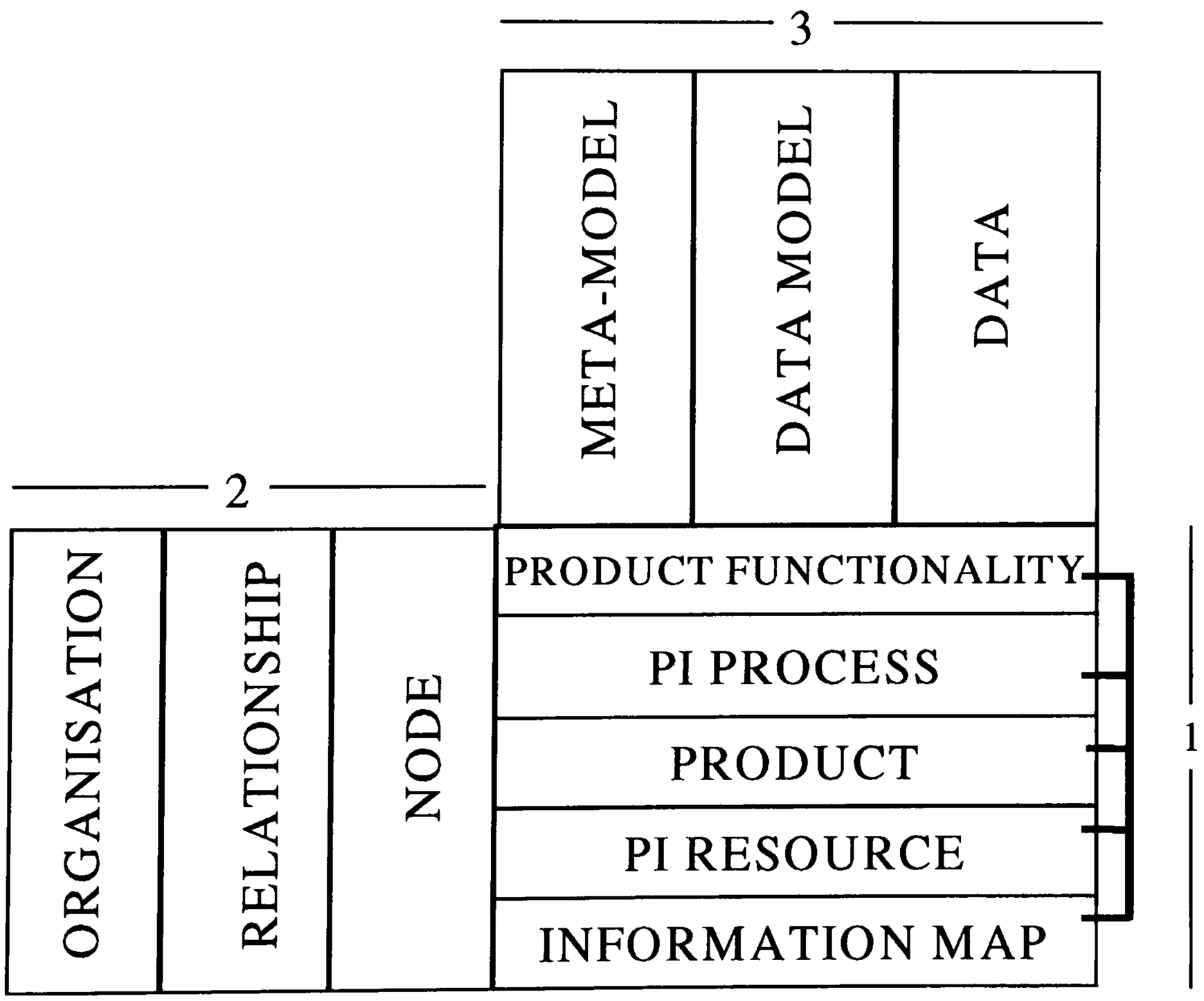
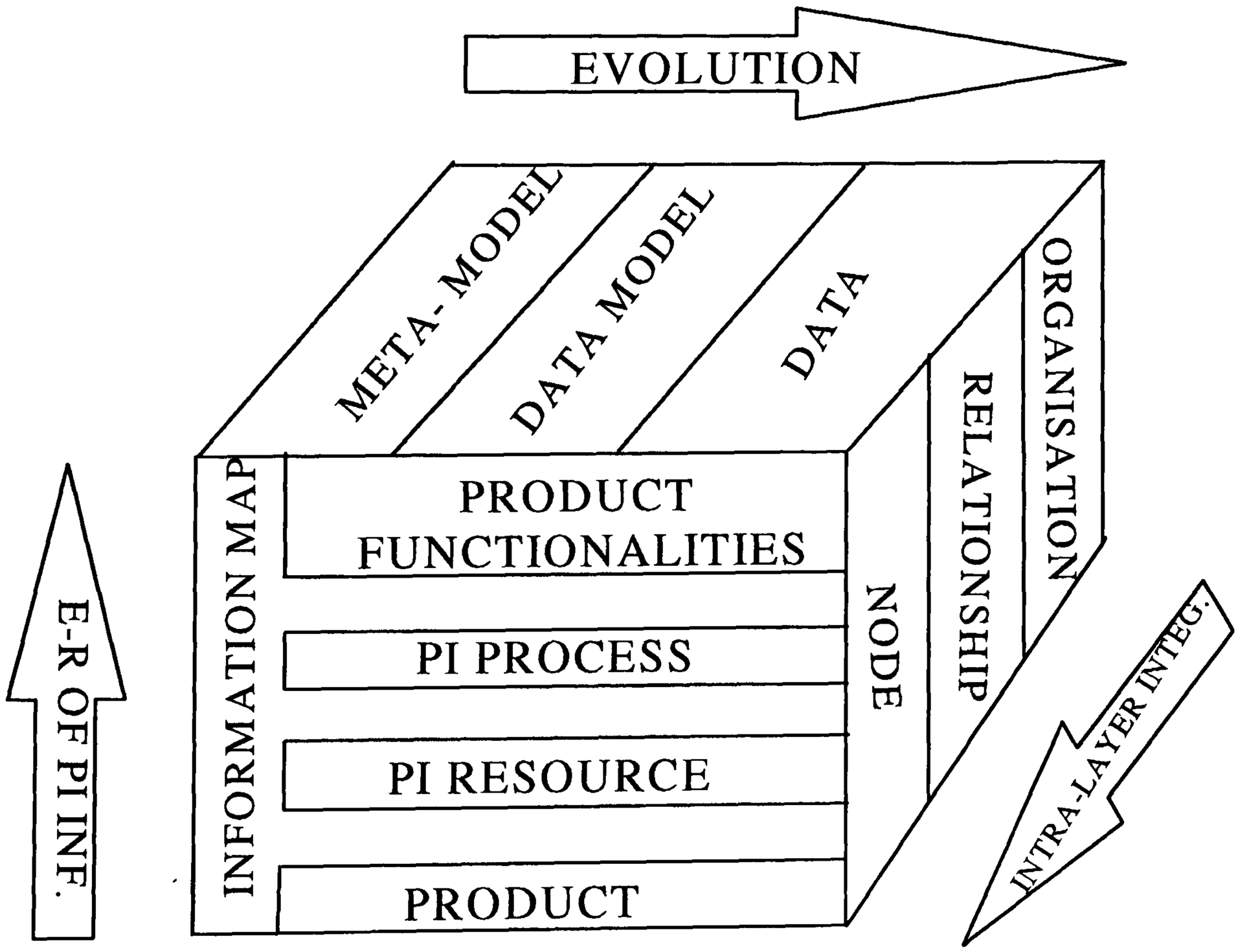


Figure 6.1 The dimensions and the layers of the information model

In order to represent the intra-entity integration within each of the layers (product functionality, product introduction process, product introduction resource, product

and information map) mentioned in the first dimension, elements of the hierarchy of a network architecture i.e. node, and relationship among the nodes of the network would form the layers in the second dimension; it would be also necessary to organise the nodes and the relationships in the networks and, hence, there is a third layer called the 'organisation'. The dimension which consists of the three layers - meta-model, data model and data - takes care of the continual evolution. The meta-model and data model are the modelling layers. The data layer is obtained by instantiating the data model in the data model layer. An overview of the architecture of the product introduction information model is given in Figure 6.2.



PI Product Introduction

Figure 6.2 The architecture of the product introduction information model

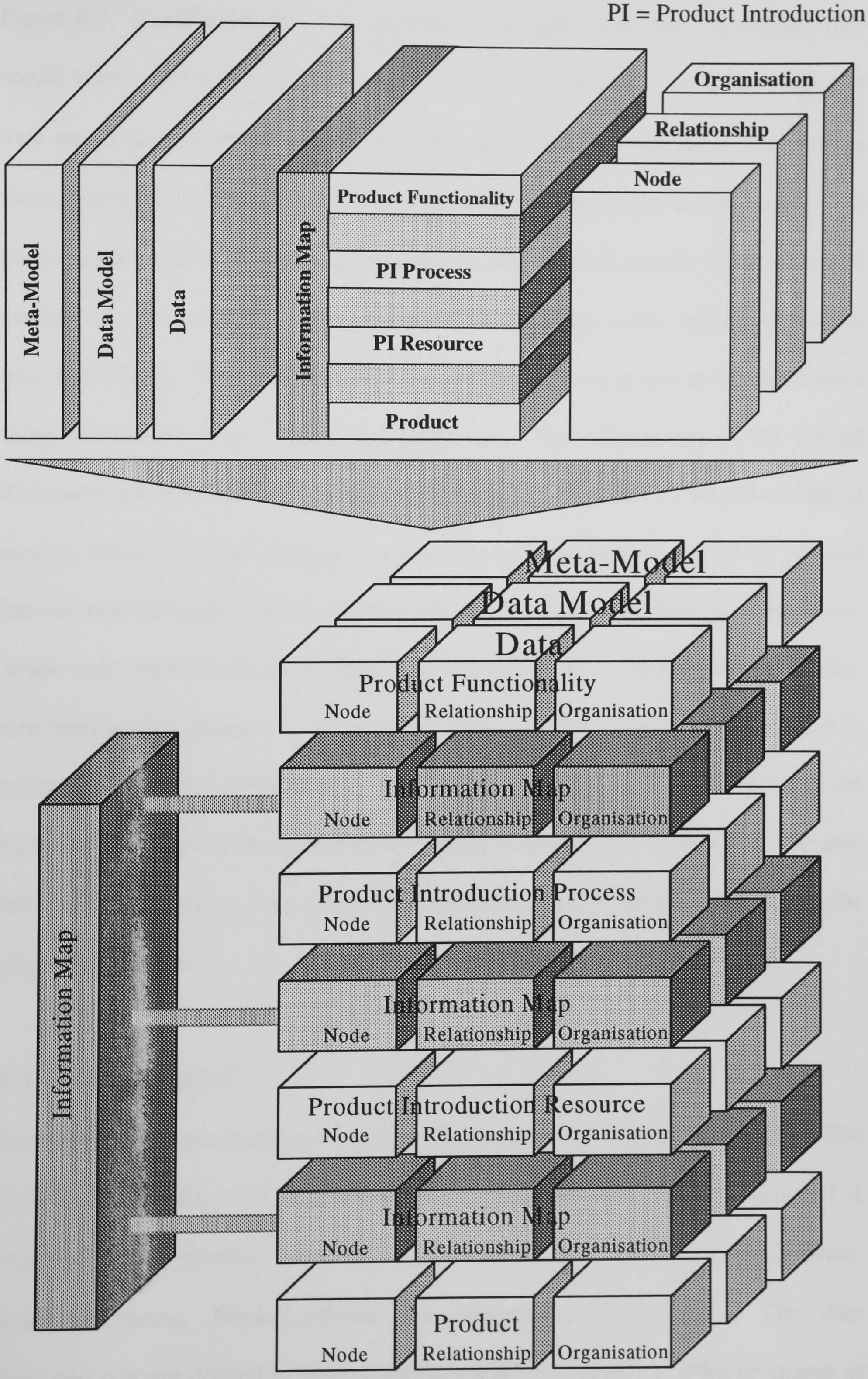


Figure 6.3 The building blocks of the product introduction information model

The building blocks of the information model for product introduction are shown in Figure 6.3. Considering the cross sections of the data model and the entities, they would correspond to the respective data models. For example, the cross section of data model and the entity 'product functionality' would correspond to the product functionality data model. Thus, the integrated information model would include:- ① product introduction process data model, ② product functionality data model, ③ product introduction resource data model, ④ product data model and ⑤ information map data model. The associations between different entities would be represented using information maps in the first dimension. The relationships in the second dimension would be used to represent the complex associations within the entities of product introduction information. As association is the main feature of the product introduction information, the description of most of the entities belonging to the blocks would make use of the above mentioned associations. Hence, the definition of the data structures in one schema would involve references to the data structures defined in another schema. The description of the data models are detailed in section 6.3, and the meta-model that would integrate the data models is described in section 6.4. The data layer would correspond to the instances or populated data of the data structures in the above data models.

6.3 DATA MODELS

The data model layer provides a frame for the product introduction information. The schema of the integrated information model (IIM) is named *IIM_Schema* and it includes the schemata - *Process_Schema, ProductFunctionality_Schema, Resource_Schema, Product_schema* and *InformationMap_Schema.* The data structures that are defined in these schemata would correspond to tables or classes of

corresponding databases on various computers. User groups, access rights, logging in identifications and passwords would need to be defined in order to provide the security over the data. As the objective of this research is to design an information model that supports the management of the PI process, these are not dealt in detail. The data structures necessary for the security of data are given in Appendix C.5.

6.3.1 Product introduction process data model

The schema corresponding to the product introduction process data model is named *Process_Schema* and it includes a data structure named *Process* that represents the PI process data (Table 6.1). Its definition makes use of the following associations - ① among the processes represented by a data structure *Process_Process* and ② between a process and an information map represented by a data structure *Process_InformationMap*. The column 'block' in Table 6.1 represents the name of the block in the architecture of the information model to which the attribute corresponds to.

Nodes in the product introduction process - The central notion in product introduction process structure would be an activity. The attributes of an activity can be derived based on various issues like identification, duration estimation, costing, associated information etc.; *name* and *description* of an activity are used for identification purposes, *duration* represents the time duration for a specific activity, and the attribute *cost* represents the cost of the activity. PI process is an aggregation of activities.

Table 6.1 Definition of the data structure for product introduction process

Schema name	IIM_Schema	
Sub-schema name	Process_Schema	
Data structure name	***Process***	
Super structure name	Nil	
Attribute	**Domain**	**Block**
name	string	Node
description	string	Node
why (or goal)	set_of *Process_Functionality*	Node
estimated_duration	numeric	Node
actual_duration	numeric	Node
estimated_cost	numeric	Node
actual_cost	numeric	Node
scheduled_start_date	date	Node
scheduled_due_date	date	Node
actual_start_date	date	Node
actual_due_date	date	Node
predecessor	set_of *Process_ Process*	Relationship
successor	set_of *Process_ Process*	Relationship
has_parts	set_of *Process_ Process*	Organisation
is_part_of	set_of *Process_ Process*	Organisation
top_forward	set_of *Process_ Process*	Relationship
top_backward	set_of *Process_ Process*	Relationship
input	set_of *Process_InformationMap*	Node
control	set_of *Process_InformationMap*	Node
estimated_resource	set_of *Process_Skill*	Node
actual_resource	set_of *Process_Resource*	Node
target_output	set_of *Process_InformationMap*	Node
achieved_output	set_of *Process_InformationMap*	Node

Relationships in the product introduction process - The predecessor and successor relationships among the activities (Figure 6.4) are represented using the attributes *predecessor* and *successor*. The data for these attributes would be filled in after analysing the dependencies among activities based on the information associated with the activities. The data structure to represent such relationships among processes is given in Table 6.2.

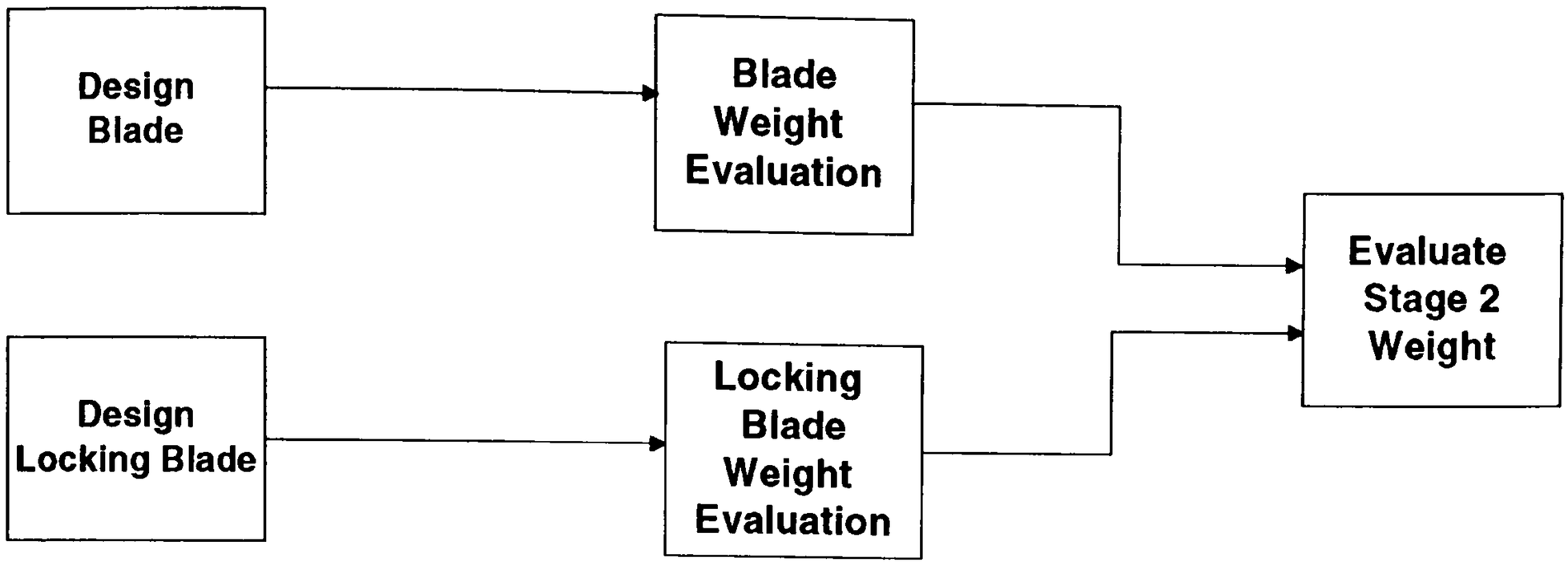


Figure 6.4 Activities with precedence relationships

Table 6.2 Definition of the data structure for association among the processes

Schema name	IIM_Schema
Sub-schema name	Process_Schema
Data structure name	***Process_Process***
Super structure name	Nil
Attribute	**Domain**
process1	Process
process2	Process

The data values of the necessary attributes and the *process_process* association are illustrated using data pointers (Tables 6.3 and 6.4). Pointers (a1, *a2, *a3, and *a4 in Table 6.3) under the column 'predecessor' would point to the associations among the processes represented by the instances of the data structure *Process_Process* (Table 6.4). In order to uniquely identify an instance of a class, identification codes would be assigned to instances; the column 'ID' in Table 6.3 represents these identification codes.

Table 6.3 A portion of process data *(Process)* showing predecessor data value

ID	name	description	predecessor
p1	Design_Blade	Design a blade for the required specification	{.....}
p2	Design_Locking_Blade	Design a locking blade for the required specification	{.....}
p3	Locking_blade_wt_evaluation	Compute the expected production weight from the CAD model	{*a1}
p4	Stage2_Weight_Evaluation	Compute the expected production weight from the individual part weights	{*a3, *a4}
p5	weight_evaluation	Compute the expected production weight from the CAD model	{*a2}

* denotes pointers

Table 6.4 Illustration of associations among processes *(Process_Process)*

ID	process1	process2
a1	*p3	*p2
a2	*p5	*p1
a3	*p4	*p3
a4	*p4	*p5

* denotes pointers

Associations between process and information map - The information maps assume various roles such as input, control and output depending on the view of the activity. Figure 6.5 shows the activity 'Blade weight evaluation', its associated information and the related activities. The data structure necessary for representing such associations between the process (or activity) and information maps is named *Process_InformationMap* and its definition is given in Table 6.5. An illustration of these associations using pointers is given in Tables 6.6 to 6.12.

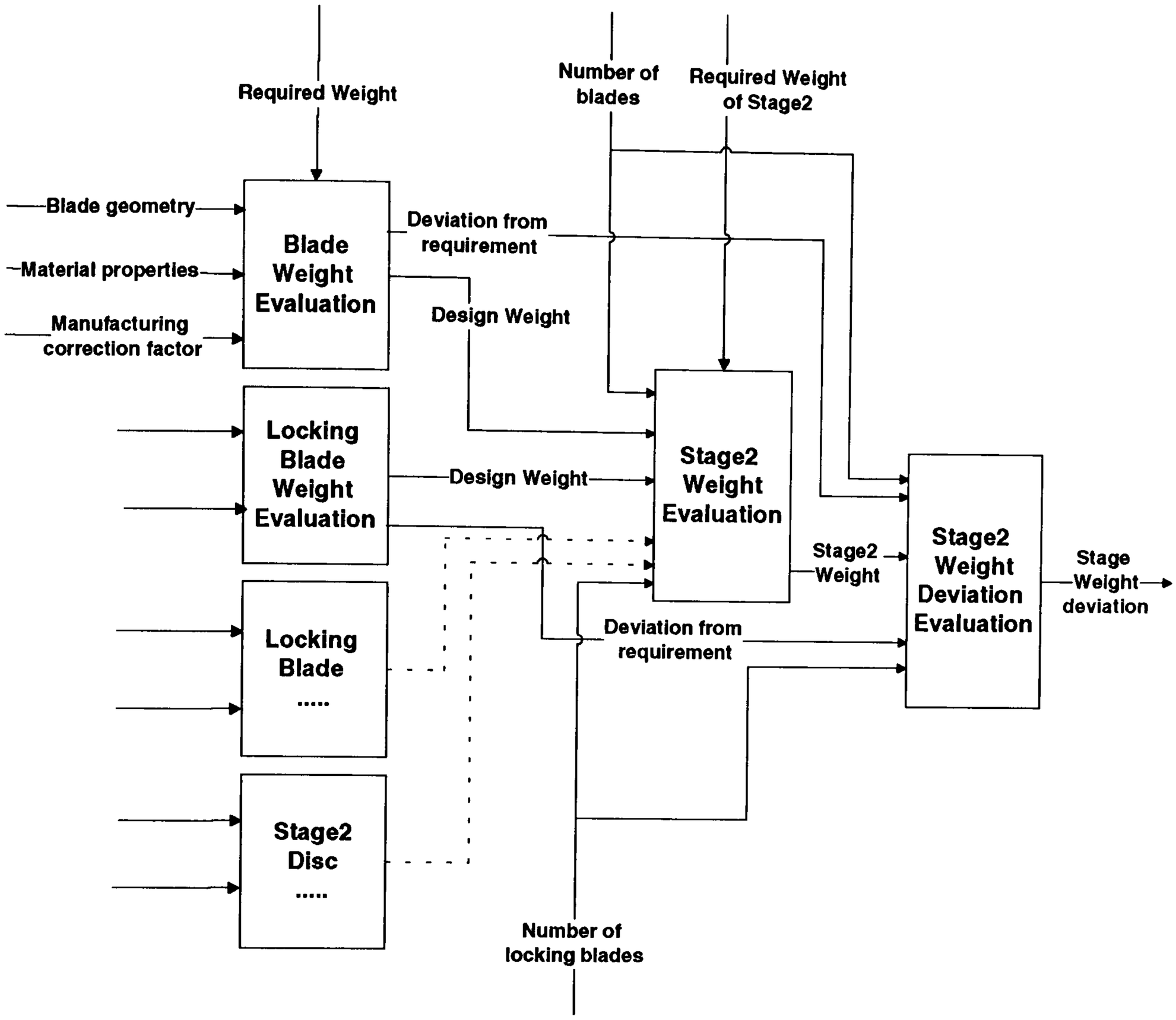


Figure 6.5 Activities and their input, control and output information

Table 6.5 Definition of the data structure for association among process and information map

Schema name	IIM_Schema
Sub-schema name	Process_Schema
Data structure name	***Process_InformationMap***
Super structure name	Nil
Attribute	**Domain**
process	Process
infmap	InformationMap

Table 6.6 A portion of process data (*Process*) showing input, control and output

ID	name	input	control	output
p5	weight_evaluation	{*q1, *q2,.....}	{*q4}	{*q5, *q6}

Table 6.7 Some instances of *Process_InformationMap*

ID	process	infmap
q1	*p5	*b2
q2	*p5	*m1
q4	*p5	*r1
q5	*p5	*d1
q6	*p5	*s1

Table 6.8 An example of an information map class - *Blade_geometry*

Instance ID	Length	Width
b1	5	2
b2	6	2.5

Table 6.9 An example of an information map class - *Material_property*

Instance ID	Name	Density	UTS	Proof_Stress
m1	Titanium	4000	1000	350

Table 6.10 An example of an information map class - *Required_weight*

Instance ID	weight
r1	24

Table 6.11 An example of an information map class - *Designed_weight*

Instance ID	weight
d1	25
d2	26

Table 6.12 An example of an information map class - *Status_weight_deviation*

Instance ID	percentage_deviation	absolute_deviation
s1	4	26
s2	4	27

The *Activity_InformationMap* is a data structure defined to store the associations between activity and the information maps associated with the activity. It would be a

special type of the association represented by *Process_InformationMap* where the element involved in the association should be an activity, not a process (Table 6.13).

Table 6.13 Definition of the data structure for association among activity and information

Schema name	IIM_Schema
Sub-schema name	Process_Schema
Data structure name	***Activity_InformationMap***
Super structure name	Process_InformationMap
Constraint	**Condition**
is_activity_informationmap	is_activity(SELF.process)

Organisation of the product introduction process - When the product introduction process is decomposed into sub-processes and when a sub-process is further decomposed into activities, then the dependency relationships - decomposition and aggregation - occur and such relationships could be represented by the attributes *has_parts* and *is_part_of* respectively (Figure 6.6). These dependency relationships

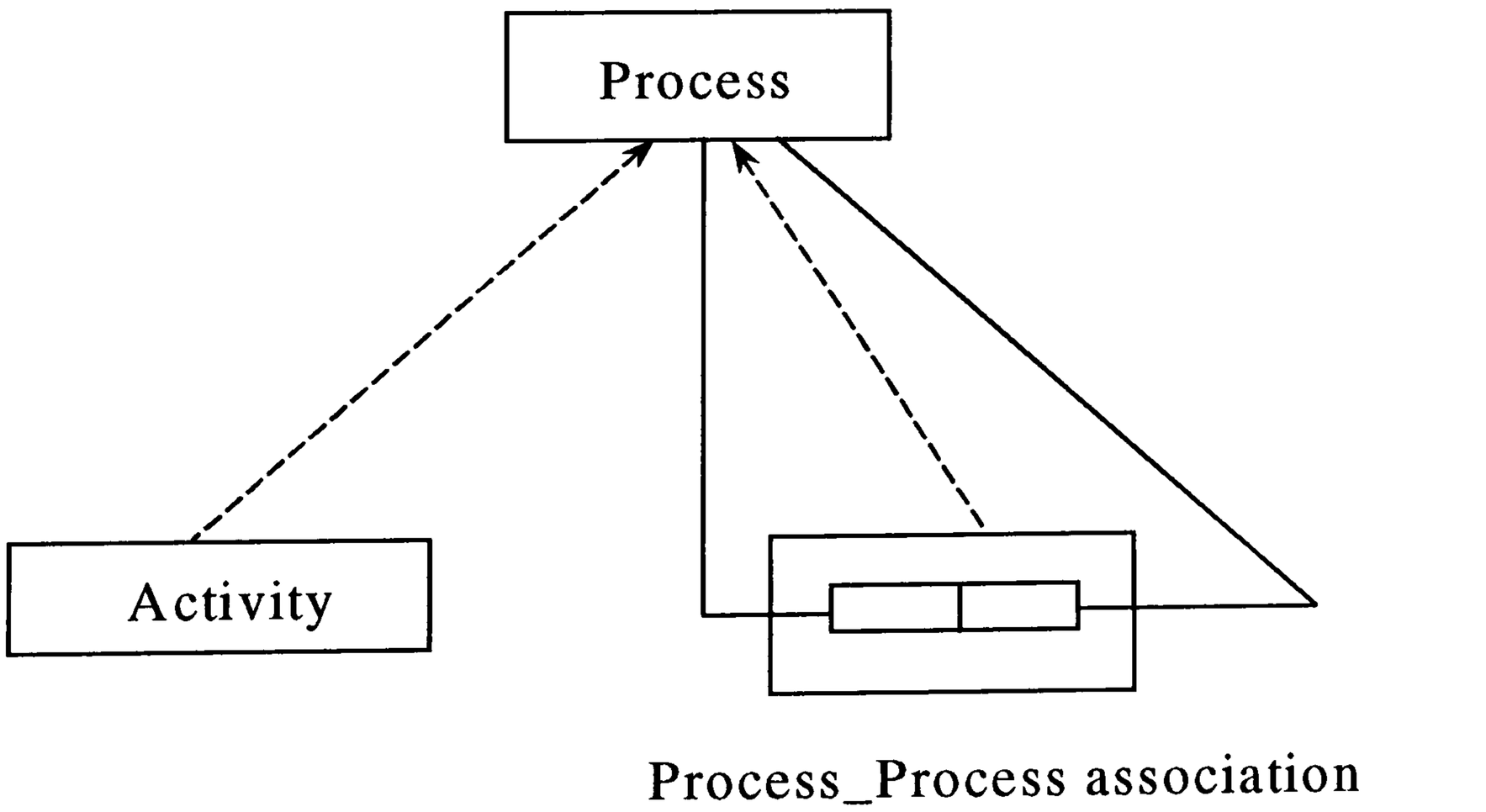


Figure 6.6 Generalisation of the product introduction process

would involve a collection of instances of the data structure *Process_Process* which is already defined in Table 6.2. (Note:- In the attribute names *has_parts* and *is_part_of*, the word 'part' has nothing to do with the physical parts of the product).

An activity is a decomposed unit of a process where no further decomposition can be done and, hence, the data structure *Activity* that would represent the data on activity would be a sub-type of *Process* where the attribute *has_parts* of an activity would be a null set (Table 6.14).

Table 6.14 Definition of the data structure for an activity

Schema name	IIM_Schema
Sub-schema name	Process_Schema
Data structure name	***Activity***
Super structure name	Process
Constraint name	**Condition**
is_activity	SELF.has_parts = Null set

Topological relationships of a process represents the other activities that are topologically linked to this process. The attribute *top_forward* represents the list of other activities topologically linked to this activity which are affected by changes to this activity, and *top_backward* represents the list of other activities topologically linked to this activity which can affect this activity. The data of these attributes would be filled in by analysing the dependencies among activities based on the information associated with the activities.

6.3.2 Product functionality data model

The functional structure of the product would be represented by the product functionality layer. As achieving a product that provides the required product functionality is the goal of the PI process and the goal has to be shared by the teams working on the PI project, product functionality is represented separately from the product information such as physical structure of the product. The schema corresponding to the product functionality data model is named *ProductFunctionality_Schema* and it includes a data structure named *Functionality* that would represent the product functionality data (Table 6.15). The definition of the data structure *Functionality* makes use of the associations - ① among the functionalitites represented by a data structure *Functionality_Functionality*, ② between a functionality and parameters represented by a data structure *Functionality_InformationMap*, ③ between a functionality and other product information represented by the data structure *Functionality_InformationMap* and ④ between a functionality and PI resources represented by a data structure *Functionality_Resource*.

As product functionality is one of the goals of the product introduction process, and as complex products may involve more than one functionality, it would be necessary to assign ownership for product functionalities. 'Ownership' would be a role played by a human resource with respect to a functionality of the product or functional subsystem of a product. The attribute *role* of the relationship between the functionality and the human resource (Table 6.16) would contain the value 'owner'.

Table 6.15 Definition of the data structure for product functionality

Schema name	IIM_Schema	
Sub-schema name	ProductFunctionality_Schema	
Data structure name	***Functionality***	
Super structure name	Nil	
Attribute	**Domain**	**Block**
name	string	Node
description	string	Node
how	set_of *Functionality_Functionality*	Relationship
why	set_of *Functionality_Functionality*	Relationship
has_parts	set_of *Functionality_Functionality*	Organisation
is_part_of	set_of *Functionality_Functionality*	Organisation
top_forward	set_of *Functionality_Functionality*	Relationship
top_backward	set_of *Functionality_Functionality*	Relationship
input	set_of *Functionality_InformationMap*	Relationship
noise_factor	set_of *Functionality_InformationMap*	Relationship
target_output	set_of *Functionality_InformationMap*	Relationship
achieved_output	set_of *Functionality_InformationMap*	Relationship
resource	set_of *Functionality_Resource*	Node

Table 6.16 Data structure for associations among functionality and resources

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Functionality_Resource***
Super structure name	Nil
Attribute	**Domain**
Functionality	*Functionality*
resource	*Resource*
role	string

Parameters play the role of input, noise factors, responses with respect to the product functionalities. The data structures that are necessary to represent the relationships between product functionality and parameters are shown in Appendix C.4. These parameters are the attributes of the products and they would evolve during the PI

process. The target values and the actual values of these parameters would also evolve.

Relationship between product functionality and the physical unit - Functional and physical integration of the product is very essential in order to have an integrated control over the diverse targets of the product introduction process. Against each product functionality, a list of product units that contribute to that functionality need to be generated. A product unit may play a role in many functionalities, and a single functionality may require many different product units. The *how* attribute of the functionality relates the *Functionality* class and *Product* class at certain stage of moving from abstract to concrete information in the problem solving system (Table 6.17). At that stage, the *why* attribute of the *Product* class would provide the inverse relationship. It is the attributes of the product that contribute to the functionality of the product. The parameters linked with the functionality are nothing but the attributes of the product represented in the product data model. The relationship between the functionality, parameters and attributes of product would be represented using information maps (Table 6.18). The data structure that represents the association among product fucntionalities is given in Table 6.19.

Table 6.17 Definition of the data structure for functionality unit

Schema name	IIM_Schema	
Sub-schema name	ProductFunctionality_Schema	
Data structure name	***FunctionalityUnit***	
Super structure name	*Functionality*	
Attribute	**Domain**	**Block**
how	set_of *Functionality_InformationMap*	Node

Table 6.18 Data structure for associations between product functionality and information map

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Functionality_InformationMap***
Super structure name	Nil
Attribute	**Domain**
functionality	*Functionality*
informationmap	*InformationMap*

Table 6.19 Data structure for associations among product functionality

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Functionality_Functionality***
Super structure name	Nil
Attribute	**Domain**
Functionality1	*Functionality*
Functionality2	*Functionality*

6.3.3 Product introduction resource data model

The structure of the resources used in the product introduction process especially the team structure is represented in this layer. The schema of the product introduction resource data model is named *Resource_Schema* and it includes a data structure named *Resource* that would represent the data on the resources (Table 6.20). The definition of the data structure *Resource* makes use of the associations ① among the resources represented by a data structure *Resource_Resource*, ② between a resource and the PI process represented by a data structure *Resource_InformationMap*, ③ between a resource and product functionality represented by a data structure *Resource_InformationMap* and ④ between a resource and product information represented by the data structure *Resource_InformationMap*. The classes for other types of resources such as physical resources (hardware and software) would need to be defined in a similar way.

Table 6.20 Definition of the data structure for PI resource

Schema name	IIM_Schema	
Sub-schema name	ProductResource_Schema	
Data structure name	***Resource***	
Super structure name	Nil	
Attribute	**Domain**	**Block**
name	string	Node
description	string	Node

The resources play various roles such as 'in-charge', 'owner' and 'performer' in the overall product introduction process; the relationship 'ownership' between a resource and project, resource and product, and resource and functionality could be captured in the attribute '*resource_of*' of the class *HumanResource.* The resources of an organisation are structured into an organisation structure; the information on 'how a team is structured' would be captured in the attributes '*has_parts*' and '*is_part_of*' of the data structure *HumanResource* of the resource data model (Tables 6.21 to 6.24). The reporting relationships among the human resources would be represented by the attributes *managed_by* and *manages* in the data structure *HumanResource.* The necessary data structures to represent the association between the resource and the skill possessed by a resource are given in Tables 6.25 and 6.26.

Table 6.21 Definition of the data structure for human resource

Schema name	IIM_Schema	
Sub-schema name	ProductResource_Schema	
Data structure name	***HumanResource***	
Super structure name	*Resource*	
Attribute	**Domain**	**Block**
managed_by	set_of *HumanResource_HumanResource*	Relationship
manages	set_of *HumanResource_HumanResource*	Relationship
has_parts	set_of *HumanResource_HumanResource*	Organisation
is_part_of	set_of *HumanResource_HumanResource*	Organisation
resource_of	set_of *HumanResource_InformationMap*	Node

Table 6.22 Definition of the data structure for person

Schema name	IIM_Schema	
Sub-schema name	ProductResource_Schema	
Data structure name	***Person***	
Super structure name	*HumanResource*	
Attribute	**Domain**	**Block**
designation	string	Node
department	string	Node
organisation	string	Node
office_address	*Address*	Node
email_address	string	Node
telephone_number	string	Node
fax_number	string	Node
skill	set_of *Person_Skill*	Node
Constraint name	**Condition**	
is_person	SELF.has_parts = Null set	

Table 6.23 Definition of the data structure for address

Schema name	IIM_Schema	
Sub-schema name	ProductResource_Schema	
Data structure name	***Address***	
Super structure name	Nil	
Attribute	**Domain**	**Block**
street	string	Node
city	string	Node
postcode	string	Node
country	string	Node

Table 6.24 Data structure for associations among human resources

Schema name	IIM_Schema
Sub-schema name	ProductResource_Schema
Data structure name	***HumanResource_HumanResource***
Super structure name	Nil
Attribute	**Domain**
Resource1	*HumanResource*
Resource2	*HumanResource*
role	string

Table 6.25 Data structure for association among person and skill

Schema name	IIM_Schema
Sub-schema name	ProductResource_Schema
Data structure name	***Person_Skill***
Super structure name	Nil
Attribute	**Domain**
person	*Person*
skill	*Skill*
level	string

Table 6.26 Data structure for skill

Schema name	IIM_Schema
Sub-schema name	ProductResource_Schema
Data structure name	***Skill***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string

6.3.4 Product data model

Product information such as information on the physical structure of the product, design drawings, production methods and materials that are generated and used by the product introduction process would be represented in the product layer. The schema of the product data model is named *Product_Schema* and it includes a data structure named *Product* that would represent the product data (Table 6.27). The definition of the data structure *Product* makes use of the associations ① between product and its functionality represented by a data structure *Product_Functionality*, ② between product and PI resources represented by a data structure *Product_Resource* and ③ among the product units (assembly, sub-assembly and part) represented by a data structure *Product_Product*. The column 'block' in Table 6.27 represents the name of the corresponding block in the architecture of the information model. As the scope of

the research is to design an information model that mainly supports the management of the PI process and captures the evolving information, the link between physical structure of the product and materials with that of production methods and drawings are defined but not dealt in detail.

Table 6.27 Definition of the data structure for product

Schema name	IIM_Schema	
Sub-schema name	Product_Schema	
Data structure name	***Product***	
Super structure name	Nil	
Attribute	**Domain**	**Block**
name	string	Node
description	string	Node
estimated_cost	numeric	Node
actual_cost	numeric	Node
why	set_of *Product_Functionality*	Relationship
has_parts	set_of *Product_Product*	Organisation
is_part_of	set_of *Product_Product*	Organisation
top_forward	set_of *Product_Product*	Relationship
top_backward	set_of *Product_Product*	Relationship
resource	set_of *Product_Resource*	Node
how	set_of *Product_ProductionMethod*	Relationship
drawing	set_of *Product_Drawing*	Relationship
material	set_of *Product_Material*	Relationship
technicaldatadefn	set_of *Product_TechnicalData*	Relationship

The actual cost of an assembly can be derived from the lower level product units that it contains; these lower level product units could be identified by following the link represented by the attribute *has_parts*. Thus, the attribute *actual_cost* that represents the actual cost of a product unit would be a derived attribute. The relationships between the product and the other elements - product functionality, PI resource, other technical information *(drawing, material, production_method)* of the product - would be represented by separate entities of the type *Product_Functionality, Product_Resource, Product_Drawing, Product_Material* and *Product_ProductionMethod* associations respectively.

1. **Relationship between product and its functionality** - product is created in order to realise the necessary functionalities. So, a product unit would be related to the product functionality by the relationship 'why the product unit exists? What is the purpose of the product unit?'. The attribute '*why*' of the data structure '*product*' links the product unit at various levels with the corresponding functionality units of the functional analysis structure.

2. **Relationship between product and PI resource -** The association 'ownership' between PI resource and product (or product units) would be represented by the attribute '*resource*' of the type *Product_Resource* in the data structure *Product.*

3. **Relationship between product and drawings** - PI process produces the detailed drawings for the product units and for the methods for producing the product units. Each assembly or part of the product may have one or more drawings associated with it; part drawing provides the part information and assembly drawing provides the assembly information. The definition of the data structures necessary to represent the data on drawing files is given in Appendix C.2.

4. **Relationship between product and production method** - The product units and the production methods are associated through the relationship 'how the product unit is produced' i.e. the production method. This association is represented by the attribute *how* of the data structure *Product;* further details on the representation of the association are given in Appendix C.1.

5. **Relationship between physical unit and material** - Material is transformed into a product by the manufacturing method. The definitions of data structures necessary to represent the relationship between product unit and the material are given in Appendix C.3.

Organisation of the product - the information on the physical structure (*has_parts, is_part_of)* of the product would be represented using the *Product_Product* association (Table 6.28). The attribute *quantity* would represent the number of units of product unit represented by *product2* that would be required in an assembly represented by the product unit *product1* (Table 6.28). An illustration of this association at the data level can be seen from Tables 6.29 and 6.30.

Table 6.28 Associations that lead to the physical structure of the product

Schema name	IIM_Schema
Sub-schema name	Product _Schema
Data structure name	***Product_Product***
Super structure name	Nil
Attribute	**Domain**
Product1	*Product*
Product2	*Product*
quantity	numeric

Table 6.29 A portion of product data

ID	name	description	has_parts	is_part_of
a1	Aeroengine	a description on Aeroengine	{*b1,.....}	Null Set
a2	Compressor	a description on Compressor	{*b2,.....}	{*b4}
a3	Stage-2	a description on Stage-2	{*b3,.....}	{*b5}
a4	Blade	a description on Turbine Blade	Null Set	{*b6}

* denotes a pointer

Table 6.30 Sample data of physical structure of the product

ID	product1	product2	quantity
b1	*a1	*a2	1
b2	*a2	*a3	1
b3	*a3	*a4	34
b4	*a2	*a1	
b5	*a3	*a2	
b6	*a4	*a3	

* denotes a pointer

An extension to the instantiation concept - In the product data model, it would be necessary to provide a data structure that would represent the technical information about product units. For example, for a 'blade' in an aeroengine, the technical information such as weight, stress, life cycle of the blade would need to be represented. The attribute called *technicaldatadefn* in the *Product* class would represent a link to the data structure that would represent the technical information on the corresponding product unit. The data structure necessary to represent such link is given in Table 6.31. The data structures that represent the technical information would be sub data structures of the data structure *TechnicalData* in the product data model (Table 6.32), and they would be created dynamically through meta-modelling technique. An illustration of how the link would be provided is given in Tables 6.33 to 6.36. It can be seen that the entity 'blade' occurs at data level as an attribute value under the attribute *name* in the data structure *Product* (Table 6.34), and at schema level as a data structure *Blade* (Table 6.36) within a single framework.

Table 6.31 Associations between product unit and its technical data

Schema name	IIM_Schema
Sub-schema name	Product _Schema
Data structure name	***Product_TechnicalData***
Super structure name	Nil
Attribute	**Domain**
Product	*Product*
technicaldata	*TechnicalData*

Table 6.32 Definition of the data structure *TechnicalData*

Schema name	IIM_Schema
Sub-schema name	Product _Schema
Data structure name	***TechnicalData***
Super structure name	Nil

Table 6.33 Definition of the data structure *Blade*

Schema name	IIM_Schema
Sub-schema name	Product _Schema
Data structure name	***Blade***
Super structure name	TechnicalData
Attribute	**Domain**
variant_ID	string
variant_name	string
weight	numeric
life_cycle	numeric

Table 6.34 A portion of product data represented by the class *Product*

ID	name	description	technicaldatadefn
a1	Aeroengine	a description on Aeroengine	{.....}
a2	Compressor	a description on Compressor	{.....}
a3	Stage-2	a description on Stage-2	{.....}
a4	Turbine Blade	a description on Turbine Blade	{*t1}

Table 6.35 Association between product units and their technical data *Product_Technicaldata*

ID	Product	technicaldata
t1	*a4	*x1

Table 6.36 A portion of blade data represented by the class *Blade*

Instance ID	variant_ID	variant_name	weight	life cycle
x1	1	Turbine Blade	25	4000

The other solution for representing the same entity at data level and at schema level would be to have the name of the data structure that represent the technical information same as the name of the product unit represented in the class *Product.* This would completely eliminate the attribute *technicaldatadefn* in the class *Product.* For example, the product unit 'Blade' would be represented as an instance of the data structure *Product* in the product data model with an instance id 'a4' (Table 6.37). If

the technical attributes such as weight, stress, life cycle of the blade would be represented in a separate data structure named *Blade* (Table 6.38) then it can be seen that 'Blade' can occur at data level (Table 6.37) and at schema level (Table 6.38). This method of having the same name for the product unit and the data structure would be useful only when the technical information of a product unit could be represented using only a single data structure, not a collection of different data structures.

Table 6.37 Sample data of product represented by the data structure *Product*

ID	name	description
a1	Aeroengine	a description of Aeroengine
a2	Compressor	a description of Compressor
a3	Stage-2	a description of Stage-2
a4	**Blade**	a description of Turbine Blade

Table 6.38 A portion of blade data represented by the data structure *Blade*

Instance ID	Variant ID	Variant name	weight	life cycle
x1	1	Turbine Blade	25	4000

6.3.5 Information map data model

PI process that generates and uses the product information is related to the product information through the semantic associations between them, and these semantic associations would be represented by the information maps in the information map layer. The schema corresponding to the information map data model is named *InformationMap_Schema* and it includes a data structure named *InformationMap* that would represent the information map data (Table 6.39). The definition of the data structure is based on how the relationship between the PI process and product

information is viewed. As the product is evolving out of the PI process, the information map classes would need to be defined dynamically.

Table 6.39 Definition of the data structure for information map

Schema name	IIM_Schema
Sub-schema name	InformationMap_Schema
Data structure name	***InformationMap***
Super structure name	Nil

Nodes in the information map - The nodes in the information map layer would be mappings that represent the semantic relationships between activity of the PI process and other entities - product, product functionality and PI resource. In order to dynamically create the data structures that represent the information maps, the definitions of the data structures of the information maps would need to be stored first in the meta-model as instances of *MetaClass*. An example of a query that would retrieve the information on the definitions of information map classes represented in the meta-model is shown in Table 6.40. The information map classes would be generated dynamically in the data model (*InformationMap_Schema*) based on these definitions (Tables 6.41 and 6.42). Thus, the information map classes that connect the product introduction process and the product information would be defined using their properties (attributes). The instances of these dynamically created information map classes are shown in Tables 6.43 and 6.44.

Table 6.40 Information represented in the meta-model

Instance ID	Database name	Class name	Attributes
1	IMapDB	Blade_geometry	{length, width}
2	IMapDB	Material_property	{name, density, uts, proof_stress}

Table 6.41 Definition of the information map class '*Blade_geometry*'

Database name	IMapDB	
Schema name	IIM_Schema	
Sub-schema name	InformationMap_Schema	
Data structure name	***Blade_geometry***	
Super structure name	InformationMap	
Attribute	**Domain**	**Block**
length	numeric	Node
width	numeric	Node

Table 6.42 Definition of the information map class '*Material_property*'

Database name	IMapDB	
Schema name	IIM_Schema	
Sub-schema name	InformationMap_Schema	
Data structure name	***Material_property***	
Super structure name	InformationMap	
Attribute	**Domain**	**Block**
name	string	Node
density	numeric	Node
uts	numeric	Node
proof_stress	numeric	Node

Table 6.43 An example of an information map class - Blade_geometry

Instance ID	Length	Width
b1	5	2
b2	6	2.5

Table 6.44 An example of an information map class - Material_property

Instance ID	Name	Density	UTS	Proof_Stress
m1	Titanium	4000	1000	350

Relationships among the information maps - The information map classes would be instantiated with values from the output of an activity or from the values of the product information base. The relationship between the attribute of the information map class and the attributes from which it is derived would be represented in the meta-model and used during the instantiation of the information map classes. As this

relationship would be among attributes, an attribute mapping technique is proposed to represent such derivation relationships. Information on attribute maps would be represented in the meta-model and they would be used during instantiation of the data model. Hence the technique is detailed in the discussion on data i.e. instances under Section 6.5.

Organisation of the information maps - An information map may be made of other information maps. For example, an output (information map) of a higher level activity could be a report aggregating the outputs (information maps) of the lower level activities. By defining the type of the attribute of an information map as an information map, the instances of the information map would become nested and form a structure. A technical report about blade that gives the details of the materials and geometry of the blade would be represented as an information map. The data structure that represents this technical report would contain relationships to the information map classes that represent material properties and the geometry of the blade i.e. to the data structures *Material_property* and *Blade_geometry* (Table 6.45). As the information maps are dynamically defined, the relationships among them would also need to be defined dynamically.

Table 6.45 Definition of the information map class 'Technical_Report_on_Blade'

Schema name	IIM_Schema	
Sub-schema name	InformationMap_Schema	
Data structure name	***Technical_Report_on_Blade***	
Super structure name	InformationMap	
Attribute	**Domain**	**Block**
title	string	Node
part_description	string	Node
material_details	*Material_property*	Relationship
blade_geometry_details	*Blade_geometry*	Relationship

6.4 META-MODEL

Meta-model integrates the data models - product functionality data model, PI process data model, PI resource data model, product data model and information map data model. The entities (Figure 6.7) necessary for integrating these data models would be represented in the meta-model layer (Figure 6.8). The schema of the meta-model is named *MetaModel_Schema* and it includes the data structures - *MetaDatabase* (Table 6.46), *MetaClass* (Table 6.47), *MetaAttribute* (Table 6.48), *MetaAttributetype* (Table 6.49), and *MetaClass_MetaAttribute* (Table 6.50) that would represent the information on databases that are to be integrated, classes of these databases, attributes of the classes, type of the attributes and the association between a class and its attributes respectively. The instances of these data structures would represent the information on the definition of the classes in order to dynamically create and update their structures in the data models.

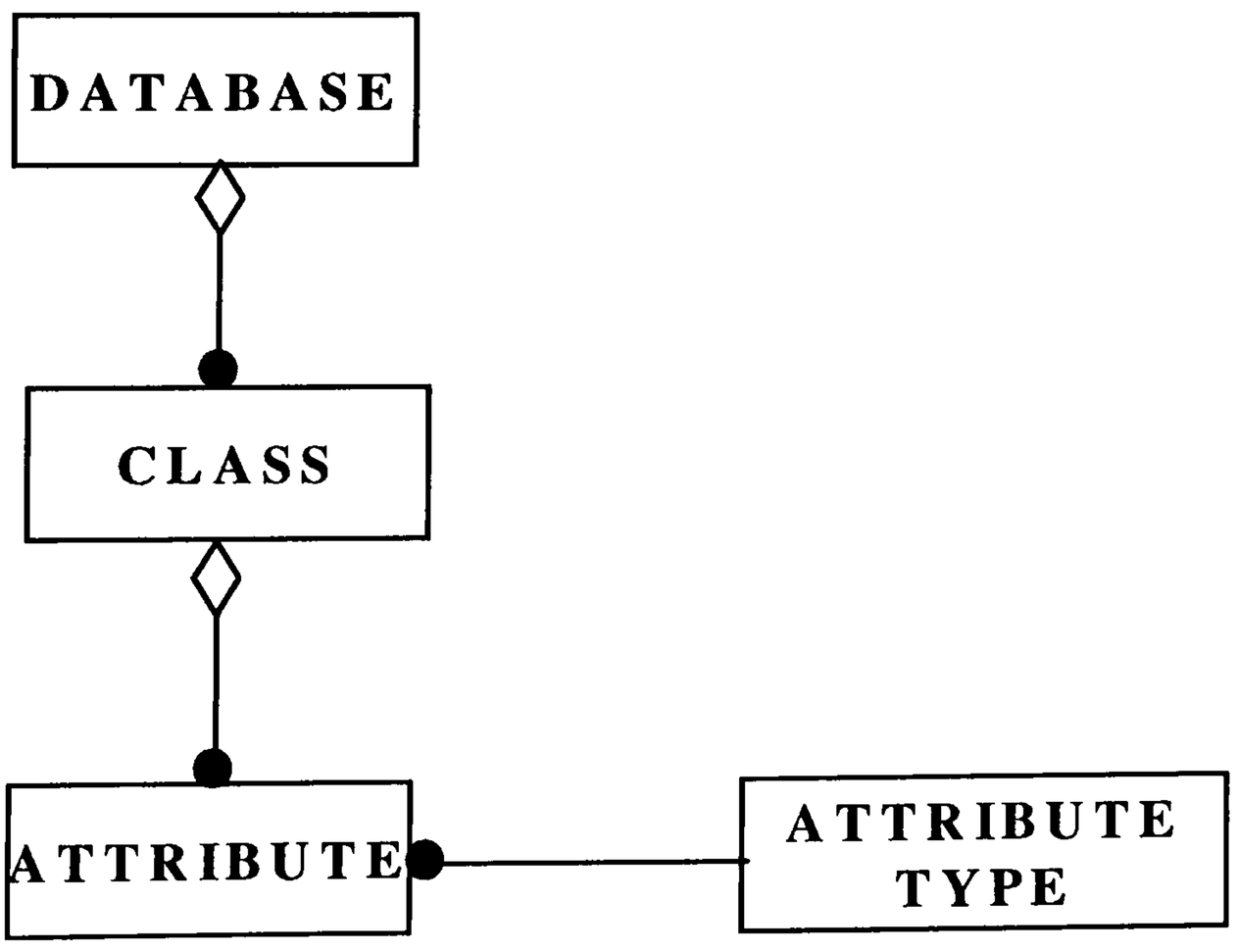


Figure 6.7 Object model of the elements in a database

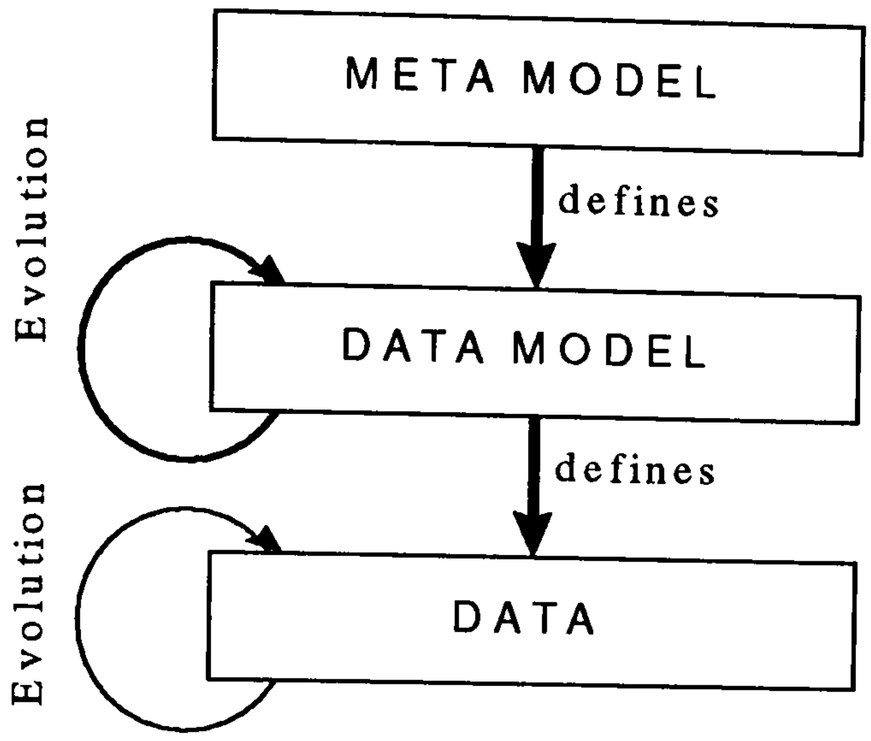


Figure 6.8 Evolution

Table 6.46 Definition of the data structure *MetaDatabase*

Schema name	IIM_Schema
Sub-schema name	MetaModel_Schema
Data structure name	***MetaDatabase***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string
path	string
classes	set_of *MetaClass*

Table 6.47 Definition of the data structure *MetaClass*

Schema name	IIM_Schema
Sub-schema name	MetaModel_Schema
Data structure name	***MetaClass***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string
superclass	set_of MetaClass
has_attributes	set_of *MetaClass_MetaAttribute*

Table 6.48 Definition of the data structure *MetaAttribute*

Schema name	IIM_Schema
Sub-schema name	MetaModel_Schema
Data structure name	***MetaAttribute***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string
type	*MetaAttributetype*

Table 6.49 Definition of the data structure *MetaAttributetype*

Schema name	IIM_Schema
Sub-schema name	MetaModel_Schema
Data structure name	***MetaAttributetype***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string

Table 6.50 Definition of the data structure *MetaClass_MetaAttribute*

Schema name	IIM_Schema
Sub-schema name	MetaModel_Schema
Data structure name	***MetaClass_MetaAttribute***
Super structure name	Nil
Attribute	**Domain**
mclass	*MetaClass*
mattribute	*MetaAttribute*

An attribute of a data structure would be either of a simple data type such as numeric, text and boolean, or of an association type. The association would be defined by the attribute type, and represented by a 'data structure' to which it is associated. The content of an attribute (in other words, the attribute value) would be either a single value or set of values. Instances of the data structure *MetaClass* would represent the information about the data structure i.e. they provide the definition of the data

structure that need to be created dynamically in the data model layer. *MetaDatabase* would provide the information on the data model in which the class has to be created.

6.5 DATA

The data layer would correspond to the instances of the data structures in the data model. In the earliest stages of product introduction process, the model may contain few, high-level objects; more details (additional, lower-level objects) would be added as the information becomes available. Completion of an activity would indicate the availability of information and, using the information maps specified at the output attribute of the *Process* class, other information classes would be instantiated. Instantiation can be done in one of the following ways - ① an attribute may be directly assigned a value, ② an attribute may be derived from the attributes of its own object using a method or ③ an attribute may be derived from one or more attributes elsewhere in the model. In case of derived attributes, it would be necessary to store the method for derivation; the derivation method may be a formula or rule. The relationships between the attribute that is derived and the attributes from which it is derived would be represented using attribute maps. An *AttributeMap* establishes an one-to-many relationship from an attribute of a data structure to the attributes in any of the data structures within the whole information model. An attribute of a data structure which is mapped in more than one information repository creates a link between the repositories that it is mapped into. In case of information maps that would assume the role of outputs of activities, the mapping would be from an information map class - attribute pair to a set of information class - attributes. It would contain the name of the class and the attribute in the information map, the name

of the class(es) and attribute(s) in the information schema, and the derivation rule. In case of information maps that would assume the role of inputs of activities, the mapping would be from an information class - attribute pair to a set of information map class - attributes. The attribute maps would be specified in the meta-model layer, thus integrating the data models.

6.6 DISCUSSION

The product introduction information has been modelled from three related but different viewpoints, each capturing important aspects of the product introduction information, but all required for a complete description. The first dimension or view point deals with the handling of the project management (PI project, PI resources) and technology knowledge (product functionalities, product), form in which the technical information are required by the activities of the project, form in which the output information is generated by the activities (information map), and the form in which they are structured (product model) for further usage. Information maps are used as the "glue" in the first dimension, providing semantic access paths between the entities of the PI process.

The overall model includes sub-models and each of these sub-model describes one aspect of the product introduction information model but contains references to the other models. The highlights of the architecture of the information model are ① information mapping technique, ② an extension to instantiation concept that helps in representing the management and technical information under a single framework and ③ meta-model that helps to represent the evolving information. Information maps

assemble sets of related information that are traditionally available at the same time but not in a consolidated form, thus facilitating easy access and use. The structure of the information map would contain pointers to the related objects that are stored in different databases, and other properties of the relationship among the related objects. It would guarantee easy retrieval of information as a group and enable information flow among the activities of the PI project by pointing to the information necessary to start an activity.

Integration is achieved through relationships; there are two types of integration in the proposed architecture: ① Inter-layer integration and ② intra-layer integration. Information maps are used to achieve inter-layer integration, and the 'organisation' and 'relationship' that represent the internal associations such as structural relationships help in achieving intra-layer integration. The proposed information model can be developed as a unified relational and object-oriented layer that sits on top of one or more different data models called local data models. A local data model is a data model that is built using one of the relational or object oriented data modelling techniques. The system that would be built based on the proposed architecture would provide users with a consolidated view of the data model that is stored in the local data models. As a result, data in all of the local databases can be accessed as if it belonged to a single database.

CHAPTER 7

7. PROTOTYPE DESIGN

7.1 INTRODUCTION

The previous chapter described how an integrated information model for product introduction (PI) process could be defined. Since the integrated information model described in this thesis is a new approach for representing the product introduction information, inherently there is no existing software tool that can be directly used to conduct a realistic test of its usefulness. To write such software tools from scratch is beyond the scope of this thesis. However, it was considered useful and practicable to develop a prototype that will show the primary features of the integrated information model. The information on the objects of the product introduction process:- product functionality, project, resources and product information correspond to the local databases (Note:- these databases are local with respect to the integrated database); these databases contain the central management and technology knowledge. This chapter gives an overview on the design and development of the local data managers. Section 7.2 gives an overview of the prototype; section 7.3 details the local data managers that manage the managerial and technical information of the product introduction process.

7.2 PROTOTYPE

The prototype is designed using a top-down approach and developed using a bottom-up procedure; it consists of three main modules :- meta-model manager, data model manager and data manager (Figure 7.1). The meta-model manager allows the

maintenance of the information about the structure, data model manager is for generating and maintaining the structure, and data manager for maintaining the instances/data. In the information model, a meta-model is used to represent the data

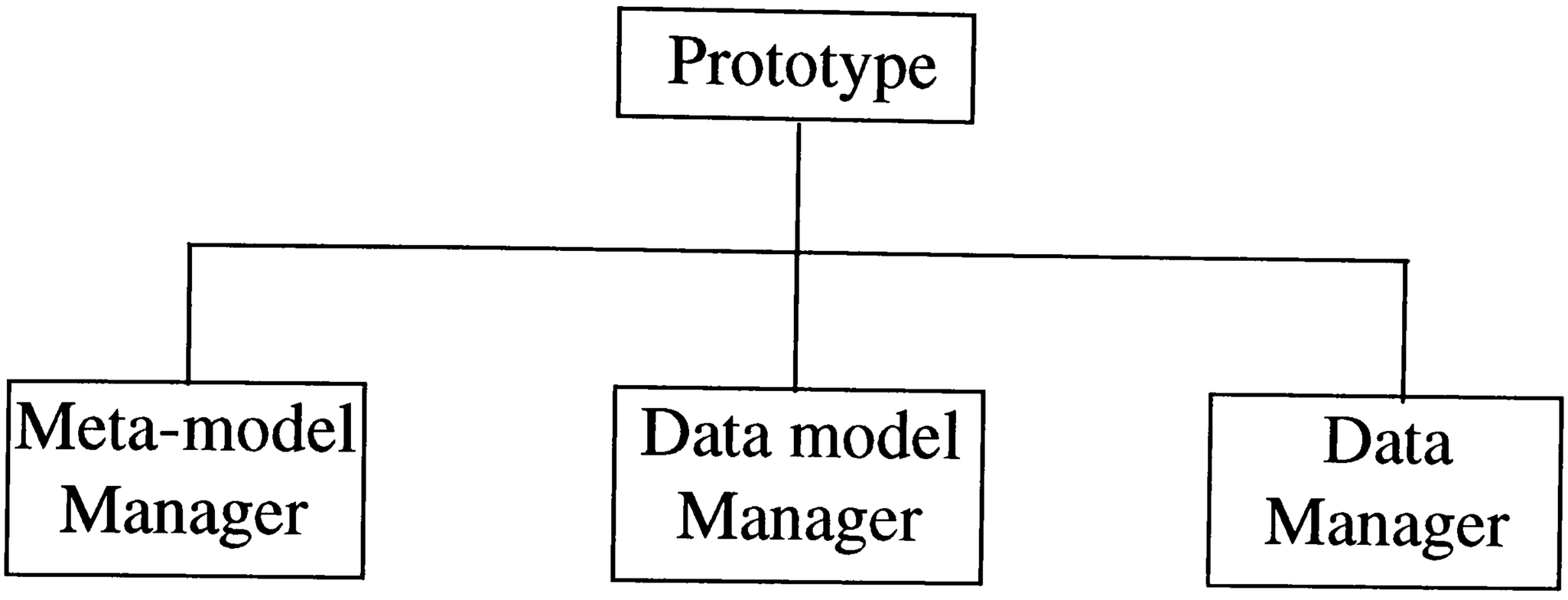


Figure 7.1 Prototype modules (Level 1)

definition of the classes and grouping of these classes into various databases; using these database and class definitions stored in the meta database, classes (and/or tables) that represent the data model are interactively generated in different databases. The classes (and/or tables) generated are instantiated, and the instances form the data layer of the product introduction information model. The relationships among the main modules of the prototype and of the information model are shown in Figure 7.2. Information on product funtionalities, product introduction project, resources and product are stored respectively in the local databases:- functionality database, project database, resources database and product database. Using the meta-model, these local databases can be integrated under a global database.

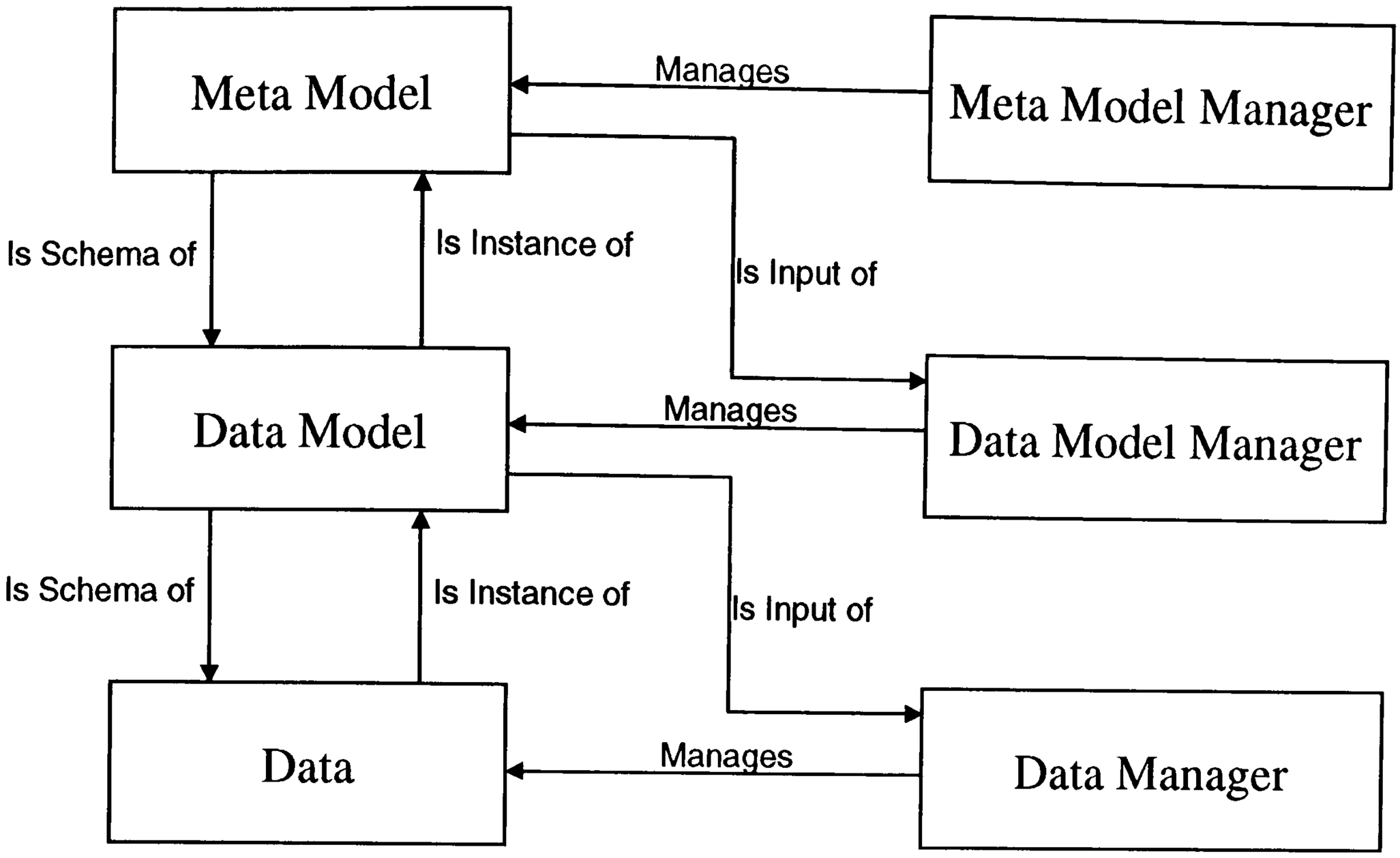


Figure 7.2 Relationships between information model modules and prototype modules

7.3 DATA MANAGERS

The data manager allows the interactive instantiation of the tables/classes generated by the data model manager. Each local database is managed by the respective data manger:- product functionality data manager, PI project data manager, PI resource data manager, and product data manager. Information maps are used to connect the information in the local databases and are stored in a database called information map database; the information map data manager manages this database. With proper user logins and passwords, users are allowed to add, modify and delete data in the local databases. The product functionality data manager allows the set up of the structures of the product functionality. Using the project data manager, one can set up the work breakdown structure and project network structure based on relative starting date; the organisational structure of the company, resource structure of the PI resources can be described using the resource data manager and the product breakdown structure can

be set up using the product data manager. The information map data manager is used for integrating the individual databases through the relationships such as 'which information is the output of which activity', 'which information is the input of which activity' and 'who is the person responsible'. The set of schemata defined in the previous chapter has been successfully used to represent all the information in the product introduction process; the relevant branches of the schema instance trees are shown using EXPRESS-G, a graphical version of the EXPRESS data specification language. A brief description of EXPRESS-G is given in APPENDIX F.

7.3.1 Product functionality data manager

The product functionality data manager maintains the data relevant to the product functionality. A class that stores the product functionality information is named *functionality*; the product functionality data manager maintains the following types of information:

1. Information on the identification of function
2. Design intent
3. Information on functional analysis
4. Information on function diagram
5. Information on information maps associated with the product functionality

Information on identification of function - Each function is described by a verb-noun phrase as this helps to sharpen the thinking on what the product is intended to achieve. Pull down lists of verbs and nouns can be provided to the user, from which he/she can frame the name of the function.

Design intent - The product functionality data manager maintains the design intent of the proposed new product - in the form of a list of critical parameters, each with its nominal value and range (Figure 7.3). These target values would be used while testing the functions and are useful in monitoring the progress of the design throughout the product introduction process.

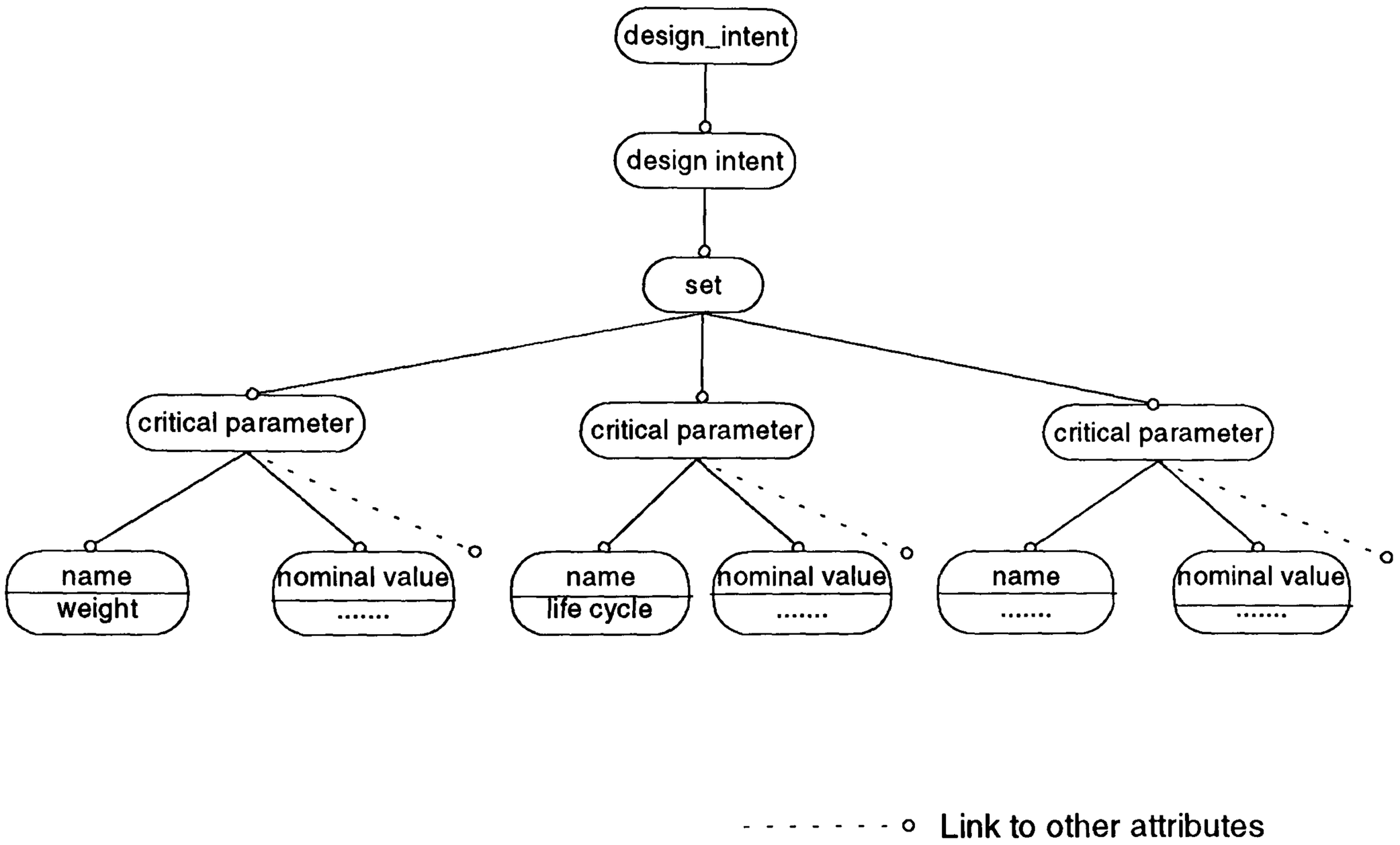


Figure 7.3 A representation of the design intent

Information on functional analysis - the functional analysis brings out the interrelationships and dependencies of all the functions. The functional relationships between all the functions of a system form a network. The relationships 'why does the function exist' and 'how does the function occur' of a function (Figure 7.4) are captured in the attributes '*why*' and '*how*' of the class '*functionality*' (Figure 7.5). When the

functional dependencies of the functions of a product is represented, functions at a higher level represent the product's functionalities, while those at the lower level, usually prefixed '*Provide* ....' represent the parts (Figure 7.5).

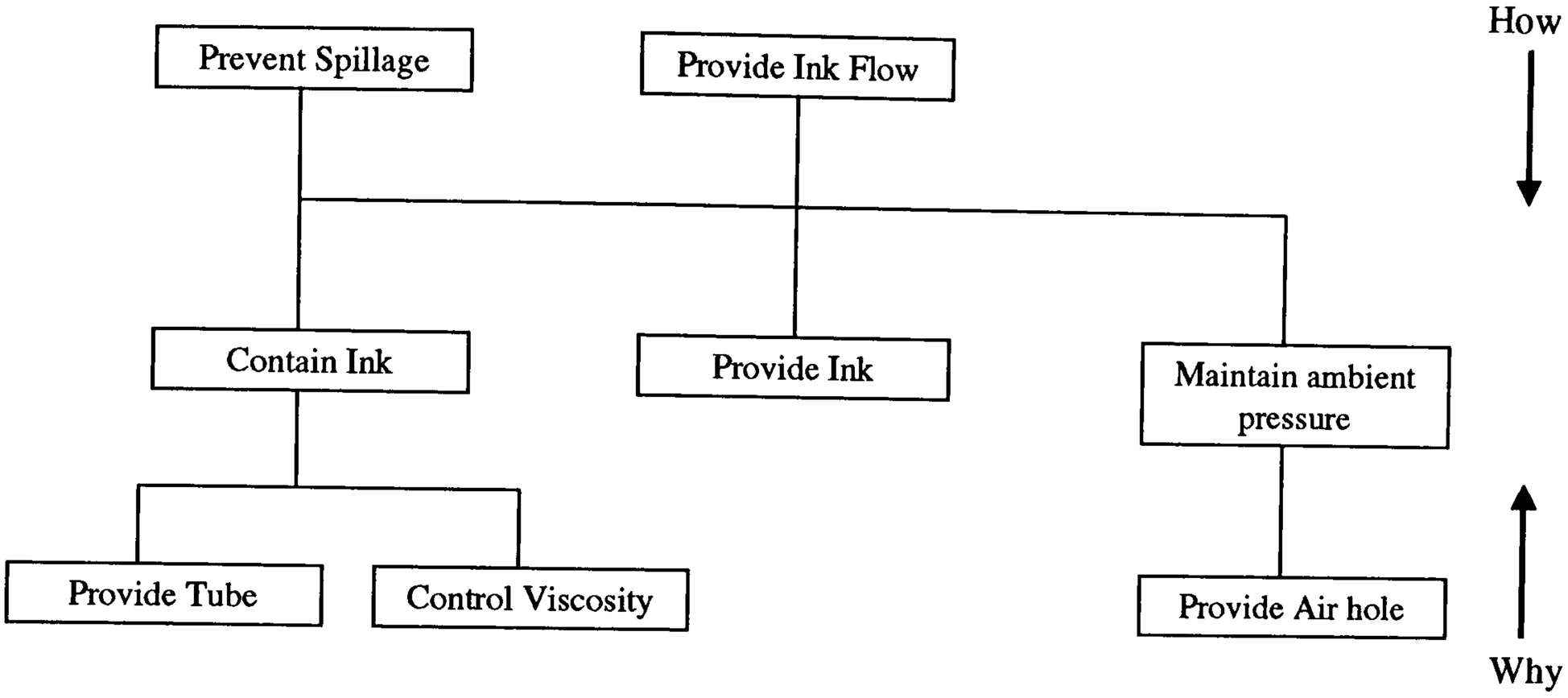


Figure 7.4 A part of the functional analysis diagram (Fox 1993)

Information on function diagram - In order to evaluate a function, it is necessary to know more about what it is that makes a function operate effectively and reliably. To understand this, it is necessary to look at what goes in to a function to make it occur and what is its output when it operates effectively (Figure 7.6). Function diagram ties together the relationships between the parameters that control the function, the function itself and the outputs from the function and their associated latitude and failure modes. Information on function diagram includes information on critical parameters (such as name, unit of measure, nominal values, range of tolerance), noise factors, responses, latitude of the response and failure modes.

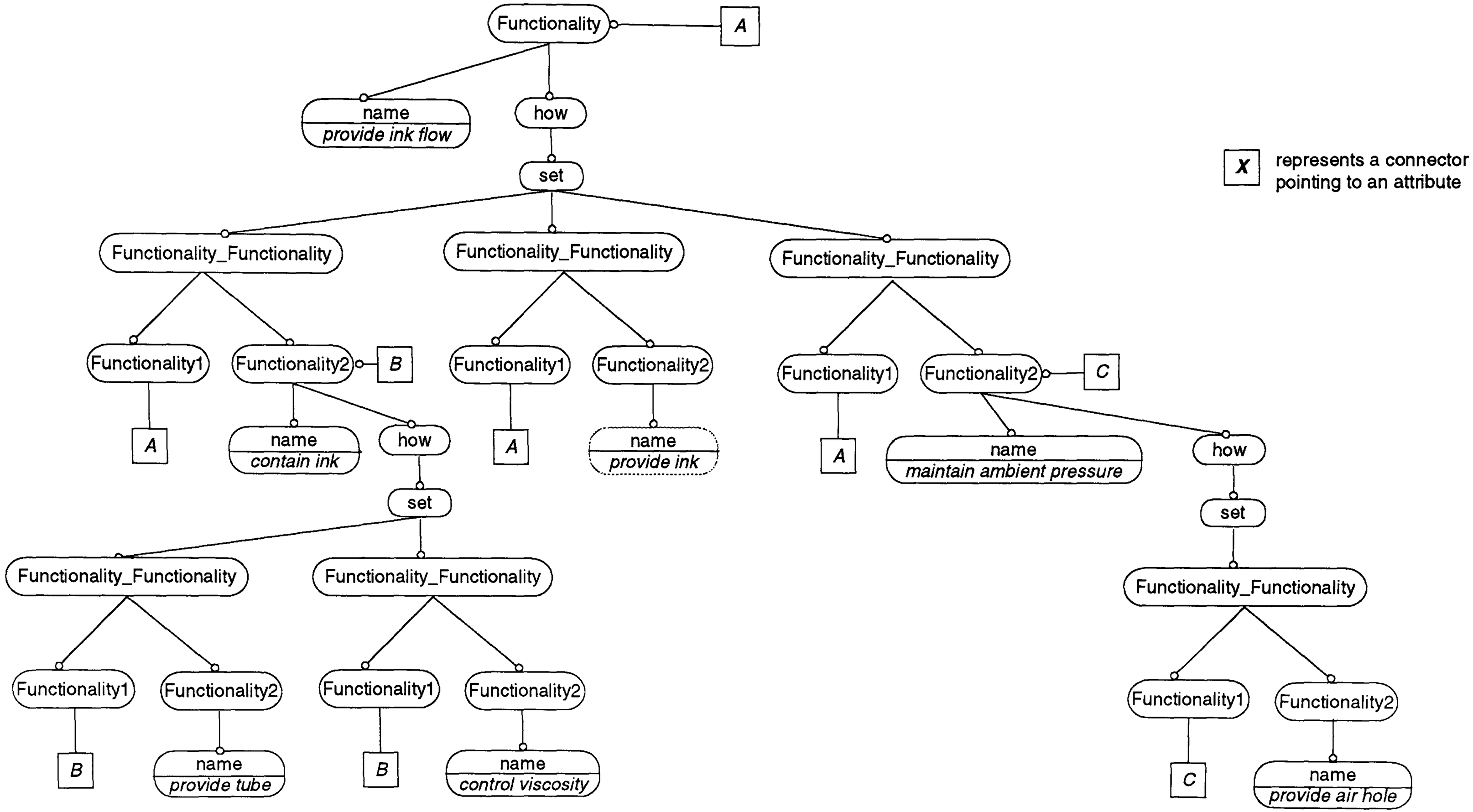


Figure 7.5 A representation of functional dependencies shown in figure 7.4

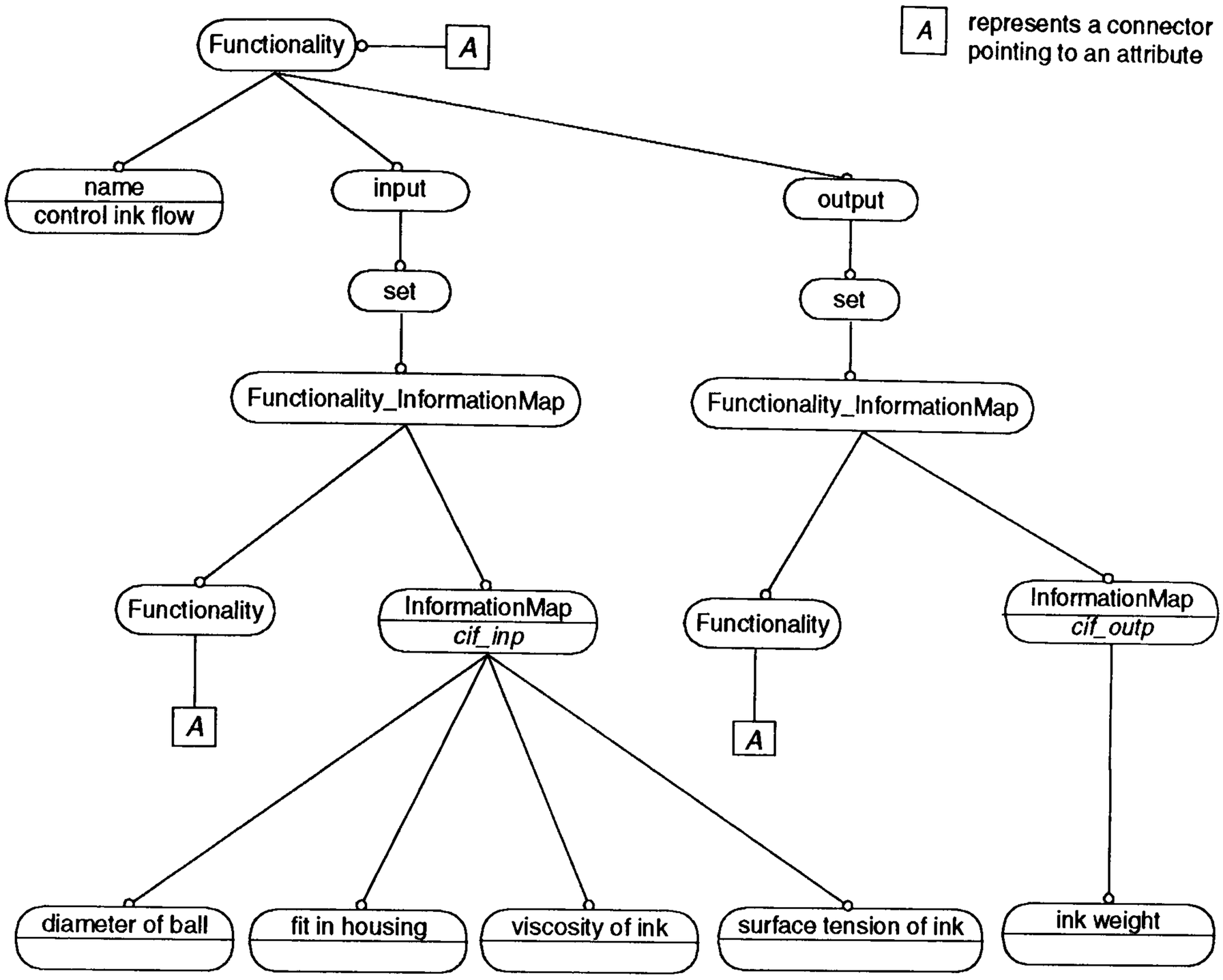


Figure 7.6 A representation of the information of the function diagram

Product functionality and information maps - Information maps are used to represent the relationships of the product functionality with project and product.

Link between product functionality and project - To achieve a product functionality in an efficient manner, the necessary design activities have to be planned and executed using project management techniques. The actions involved in a new product introduction are:

1. identification of the functions that are most prone to failure
2. linking the failure modes with the critical parameters that drive them; the list of the critical parameters provides the basis for deciding where the design effort should be

applied to enable the maximum gain to be made in terms of customer satisfaction of the product.

3. reviewing each of the critical parameters in the list and evaluating the following:

   3.1 how much more latitude can be gained and failure modes eliminated by enhancing the control on the critical parameter

   3.2 how much of this work has already been done

   3.3 how much effort and resource will need to be expended to achieve the gains that result from this study. This must be weighed against the needs of the new product - how much can be afforded, when it is required, etc..

The relationships between the product functionality and project would be captured in the attributes *target_output* and *achieved_output* of the class *project* (Figure 7.7); the information maps that would be used to represent this relationship would include the corresponding performance parameters.

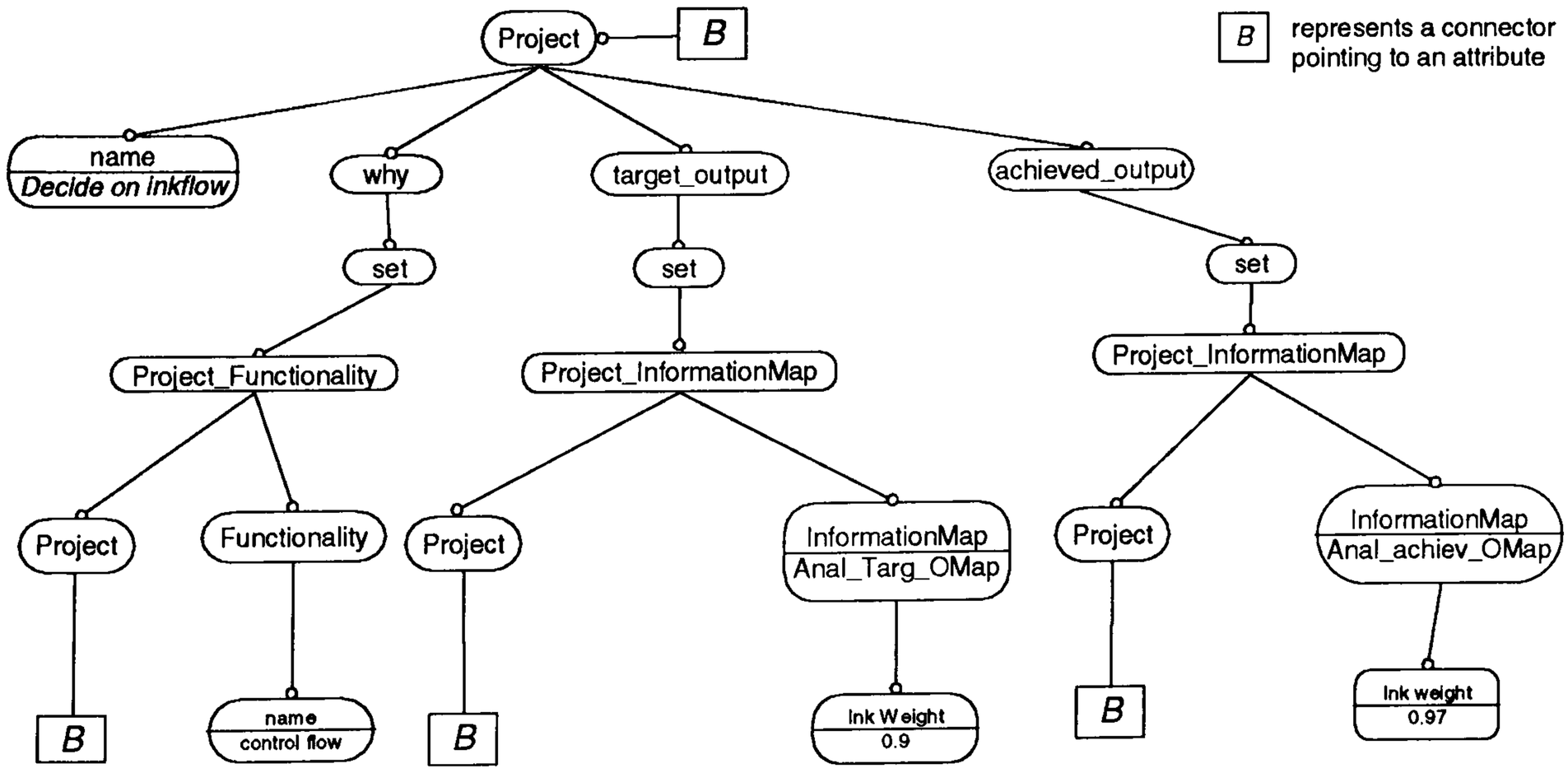


Figure 7.7 A representation of the relationship between project and product functionality

Link between product functionality and product - Information about the performer or the enabler of the product functionality is stored in the product database. The attribute *'why'* in the class *product* provides a link between the physical structure of the product and the functional structure of the product; and this link is detailed in section 7.3.4 under product data manager.

7.3.2 Product introduction project data manager

PI project data manager maintains the data relevant to the PI project. The following types of information are modelled within the project framework:

1. Information on project structure
2. Information on sequencing of activities
3. Information on scheduling
4. Information on budgeting
5. Information on resource assignment
6. Information on information maps associated with the activity
7. Information on information exchange between activities

Information on project structure - The hierarchical, dependency relationships in the work breakdown structure of the PI project are captured in the attributes *'has_parts'*, *'is_part_of'* of the class *'project'* (here the word 'part' has nothing to do with the physical parts of the product) (Figure 7.8). The attributes *'has_parts'*, *'is_part_of'* account for decomposition and aggregation respectively; an activity is a special type of project for which *'has_parts'* attribute points to a null set.

Information on sequencing of activities - The central notion in product introduction project structure is the notion of an activity. From the view point of the relative starting date among activities (and/or subprojects), the product introduction project is represented by a collection of network structures. This type of relationship can be called "A connected-to B"; sequencing relationships among the activities are captured in the attributes '*predecessor*' and '*successor*' of the *project* class. These attributes would be instantiated by analysing the information associated with the activity and their semantics.

Information on scheduling - The attributes *scheduled_start_date, actual_start_date, estimated_duration* and *actual_dura*tion of the class *project* are used in storing the scheduling information of activities.

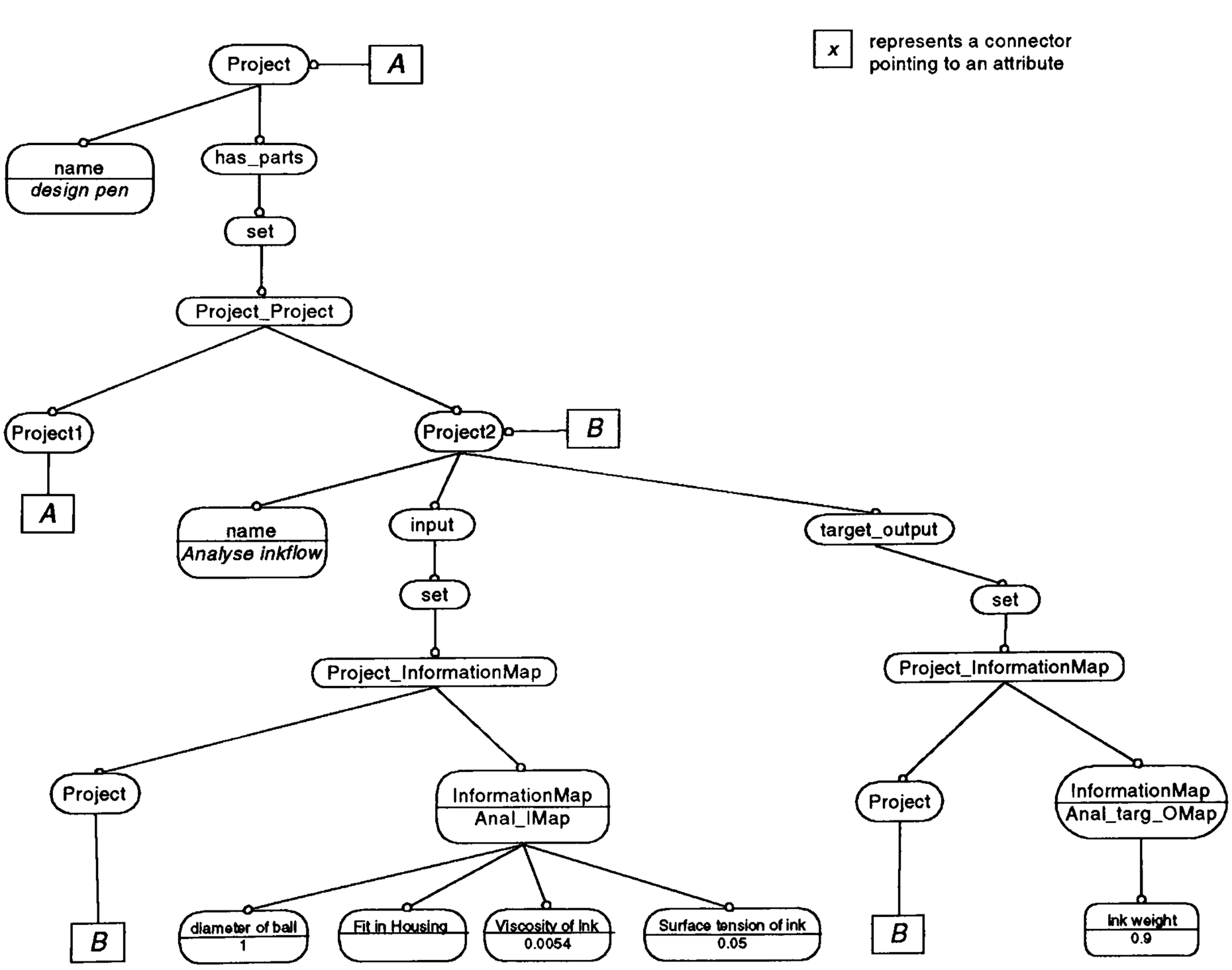


Figure 7.8 A representation of the project information

Information on resource assignment - The scheduling of activities depends on the resources that will be committed; the relationships between project information and the resource information are captured in the attributes *'estimated_resource'* (Figure 7.9) and *'actual_resource'* (Figure 7.10) of the class *project.* The attribute *'actual_resource'* contains the answer to the query 'Which organisational units perform the individual activities?'. The attribute '*role*' defined in the *Project_Resource* class would represent information such as ownership when owners would be assigned to the projects.

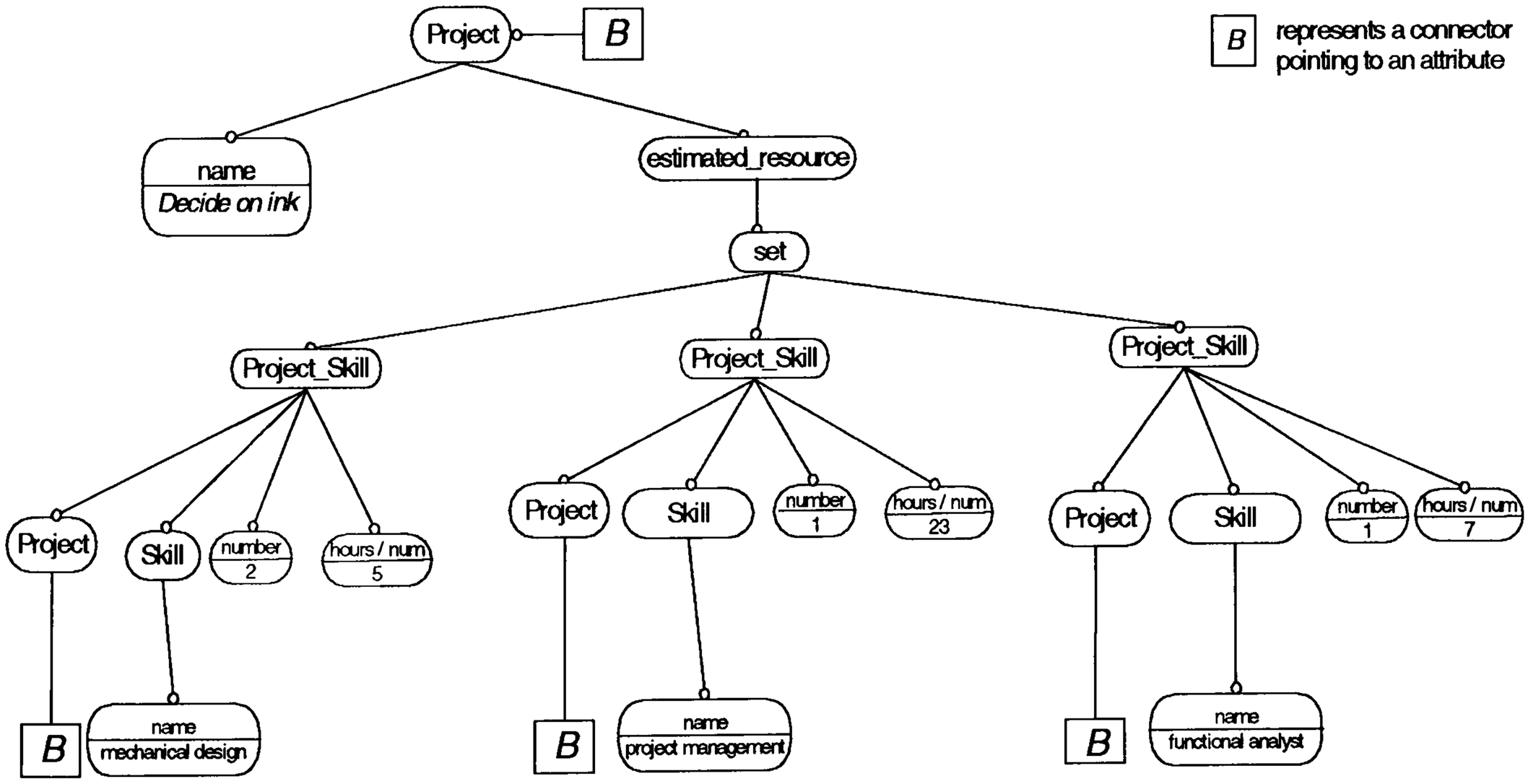


Figure 7.9 A representation of the information on estimated resources of a project

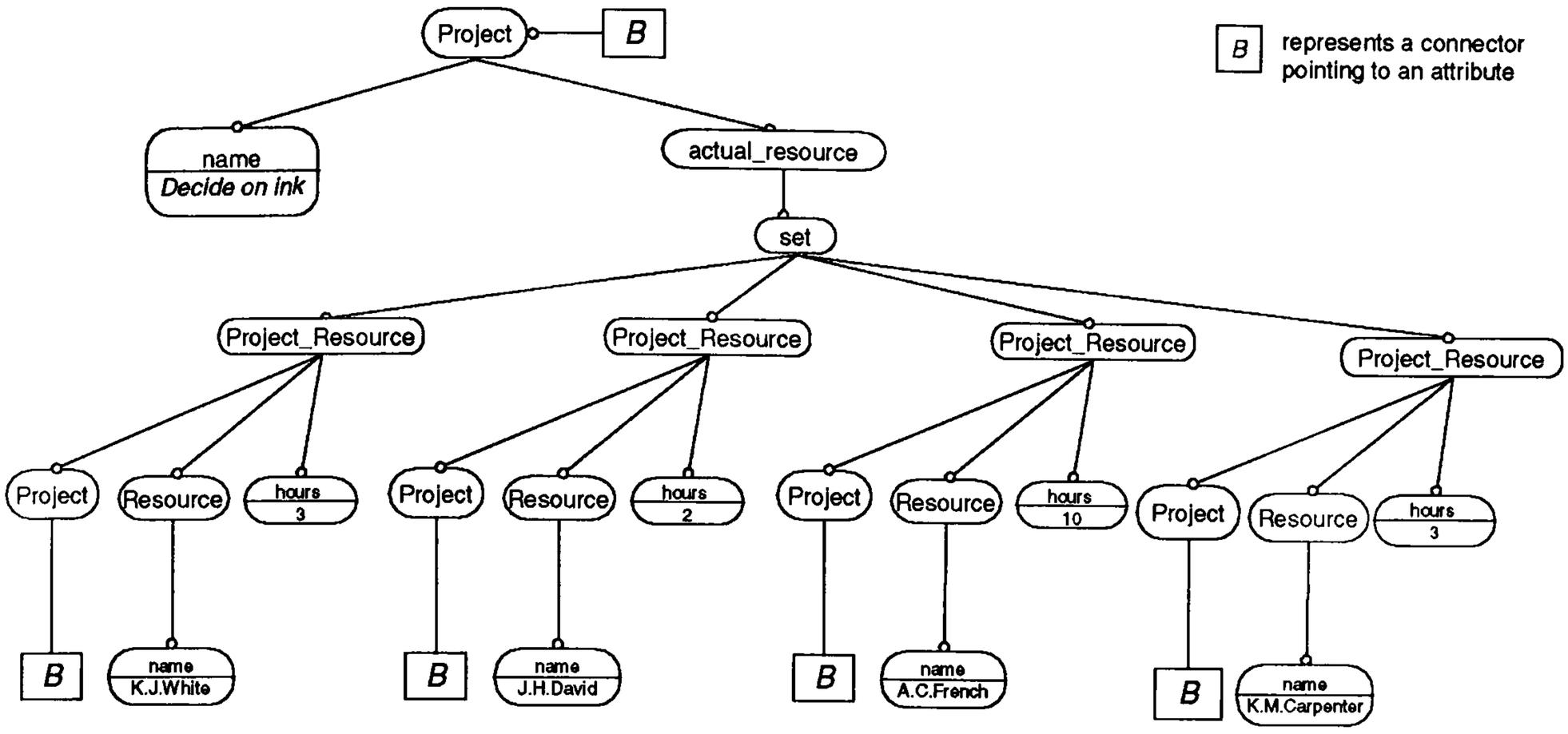


Figure 7.10 A representation of the information on actual resources of a project

Information on budgeting - Specific budgets for each activity can be created based on the resources that will be dedicated to the activity and specific time at which it is scheduled. The attributes *estimated_cost* and *actual_cost* are used to store the budgeting information of the activity.

Information on information maps associated with the activity - Problem-solving depends on combining different knowledge sets which may be widely distributed across the organisation. The product introduction project is related to product functionality structure, resource structure of product introduction, product structure through the attributes *input, resource* and *output (target_output* and *achieved_output)*; these attributes are of the type *set of information map* (Figure 7.8). Information maps that specify the input, output, resources and control information for each of the activities can be maintained using the PI project data manager.

Information on information exchange between activities - Apart from the hierarchical links (through the attributes *has_parts* and *is_part_of*) and the information links (through the attributes *input, output* of the type *set of information map*) of the activities of the PI project, communication accelerator that alerts the team about the readiness of the appropriate information as soon as the information becomes available has been designed which can be implemented using the information maps and attribute maps. As the attribute maps form a part of the meta-model, they are dealt in detail under meta-model manager in chapter 8.

7.3.3 Product introduction resource data manager

The product introduction resource data manager allows for entering, modifying, deleting, viewing and querying the information about the resources of the product introduction process. The resources of the PI process include human resources that form a team structure and computational agents consisting of hardware and software. Information about the human resources which supports or which does the overall product introduction process is represented using the class *HumanResource* with the attributes *name and description. Name* represents the name of the team or sub-team depending on the level of decomposition in the resource structure. The elementary notion in the human resource structure is *Person* with the attributes: *designation, department, organisation, office_address, email_address, telephone_number, fax_number* and *skill.* Two types of information are modelled within the resource framework:

1. Information on resource structure
2. Information on the role of resources

Information on resource structure - The resources of an organisation are structured into an organisation structure; the information on 'how a team is structured' is captured in the attributes *'has_parts'* and *'is_part_of'* of the class *HumanResource* (Figure 7.11).

Figure 7.11 A representation of resource dependencies (an instance of the *Resource* class)

Information on the role of resources - The resources play various roles such as 'incharge', 'owner' and 'performer' in the overall product introduction process. In

complex products, product is broken down into subsystems and a particular individual is made responsible for preserving the design of the subsystems of the product. Thus there exists a relationship between the human resource and the project; it is captured in the attribute '*resource_of*' of the class *HumanResource.* A resource plan is required to accomplish the work breakdown structure; the scheduling of the activities depends on the resources that will be committed. The relationship such as 'owner' and 'manager' between the human resource and the project is also captured in the same attribute (Figure 7.12).

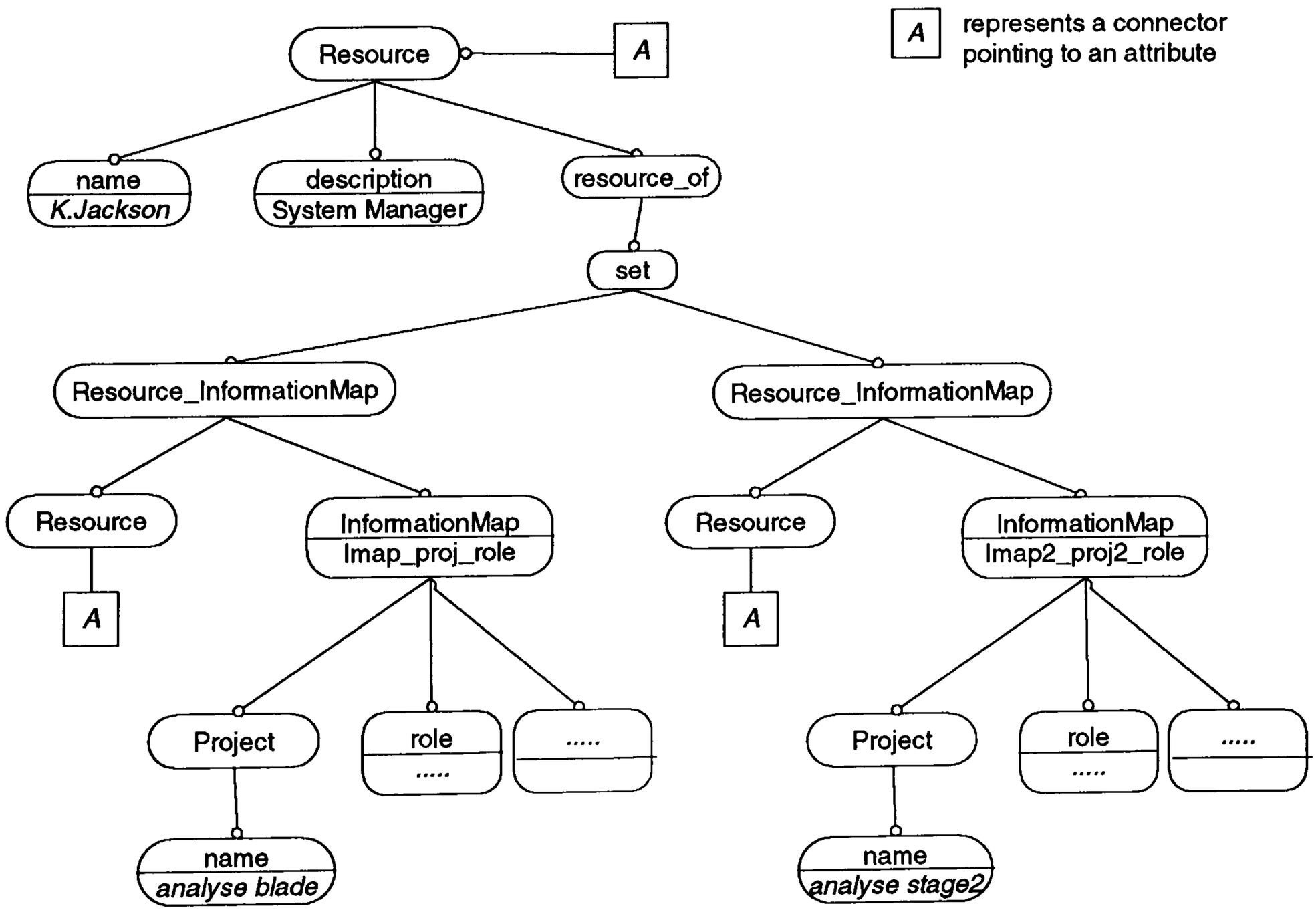


Figure 7.12 Representation of the roles played by a resource

7.3.4 Product data manager

The product data manager allows the maintenance of the data relevant to the product such as physical structure of the product and technical details of the product (design - drawings in computer aided design files and production method). The product data is represented using a class *product* with the attributes *name* and *description* for the identification purpose, and *cost* to store the cost of the product; the elementary node in the product structure is a *part*. Product data manager captures five types of product information:

1. Information on the physical structure of the product
2. Functionality of the product
3. Technical Information of the product
4. Information on the source of the product information
5. Information on product cost

Information on physical structure of the product - The dependency relationships 'aggregation' and 'decomposition' in the physical structure of the product (i.e. Bill-of-Materials) are captured in the attributes *is_part_of* and *has_parts* of the class *product* respectively (Figure 7.13).

Functionality of the product - The information on the functionality of the product (or part) is stored in the product functionality database. The attribute '*why*' of the class *product* provides a link from the product database to the functionality database (Figures 7.14 and 7.15).

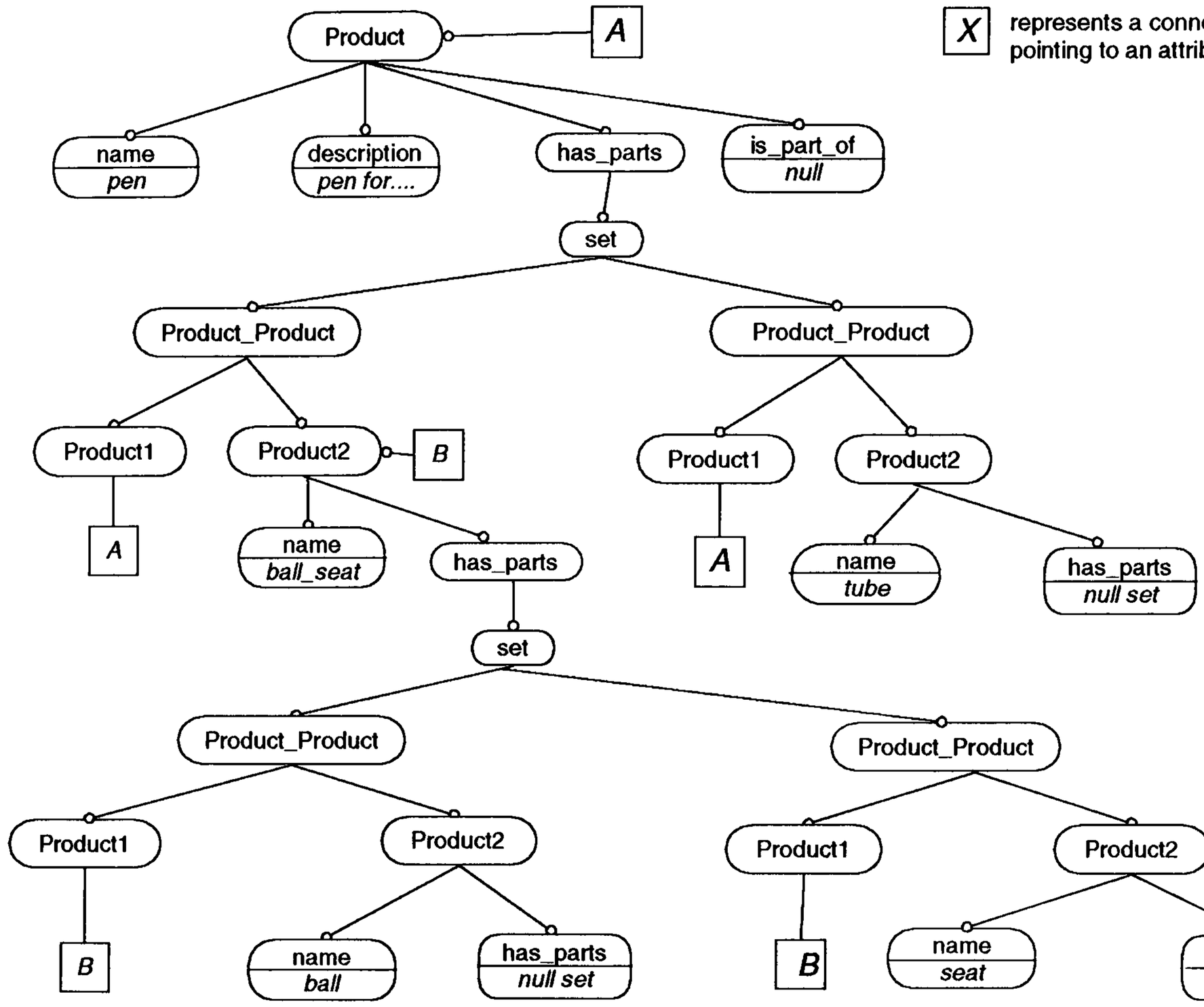


Figure 7.13 A representation of the physical structure of the product (Instance of the *product* class showing the *has_parts* attribute)

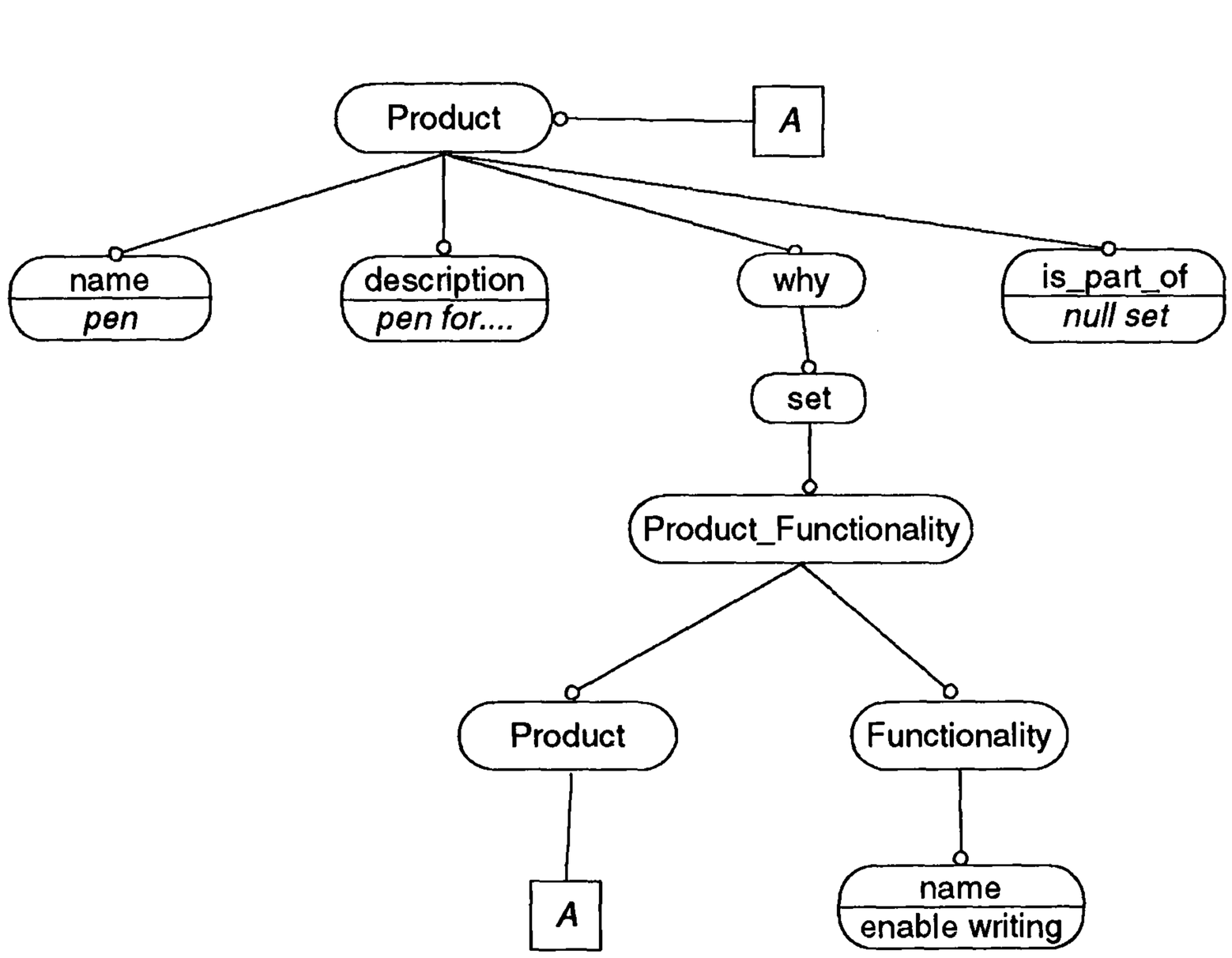


Figure 7.14 A representation of the relationship between product and its functionality

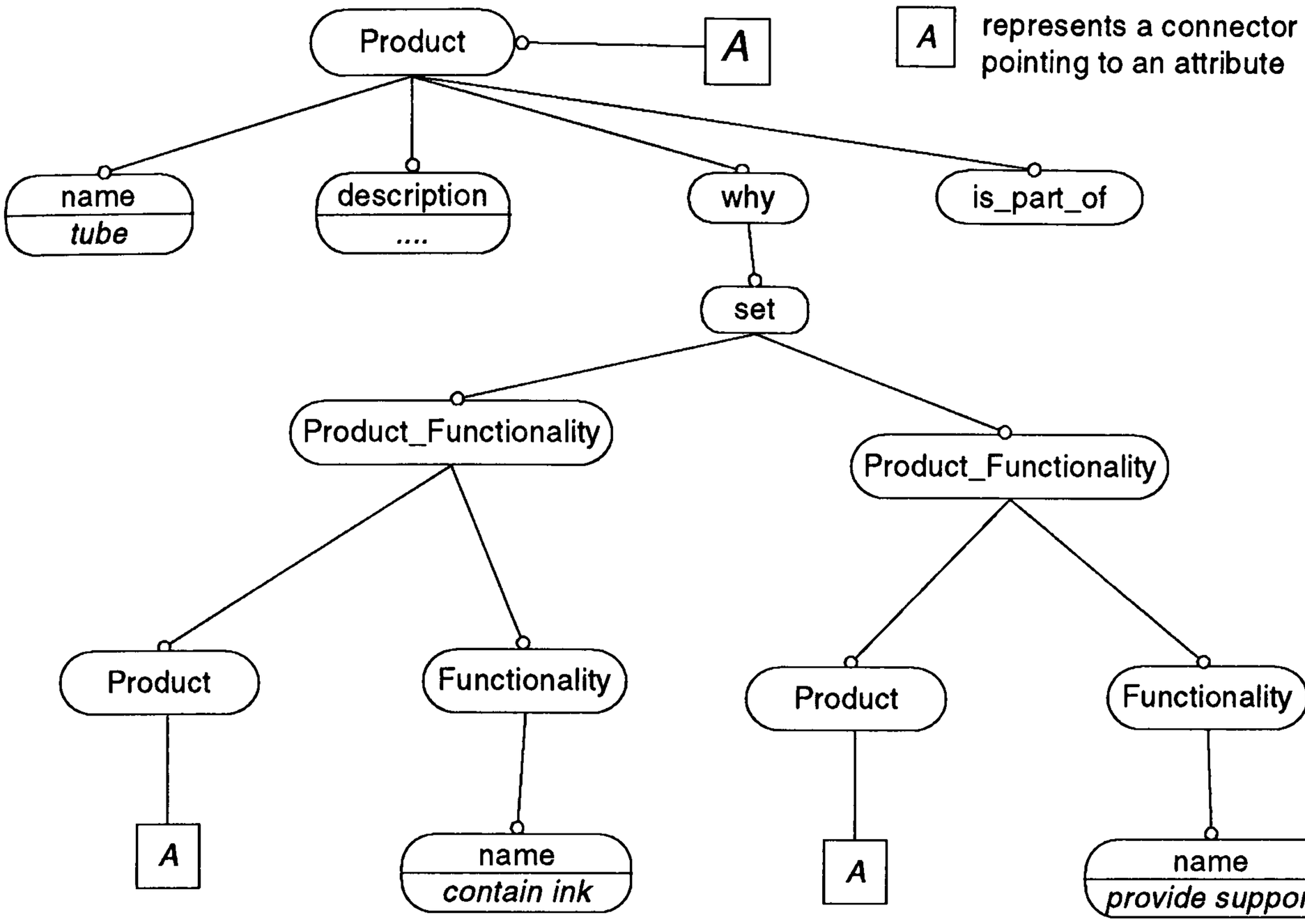


Figure 7.15 A representation of a product providing more than one functionality

Technical information about the product - The attribute '*drawing*' of the class *product* represents the information about the computer-aided design files for the product units as well as the methods for producing them. Each instance of the *product* class has an associated class with the name given by *product.name*; this associated class stores the parameters of the product (Figure E.1 in Appendix E). The attribute '*material*' of the class *product* represents the relationship between the product unit and the material used to produce the product unit. The attribute *'how'* of the class *product* represents the relationship between the product unit and the method for producing it.

Information on product cost - The attributes *estimated_cost* and *actual_cost* are used to store the cost of the product. Material cost and production cost (assembly

cost) could be applied to each assembly, assembly costs aggregated to provide an estimate of the product cost (Figure 7.16).

7.3.5 Information map data manager

The relationships among product functionality, PI process, PI resources and product are represented using information maps which assemble sets of related information that are traditionally available at the same time but not in a consolidated form facilitating easy access and use. The structure of the information map contains pointers to the related objects that are stored in different databases, and other properties of the relationship among the related objects. Information map (Figure 7.8 in section 7.3.2) guarantees good and easy retrieval of information as a group and enables information flow (which information are necessary for performing the activity?) among the activities of the PI project. The dependencies among the activities can be traced easily by following the information maps and attribute maps.

7.4 DISCUSSION

The prototype captures the product introduction process information at a number of levels:- managerial level - project database, resources database; technical level - product functionality database, product database; and relationship level between managerial level and technical level - information map database. Individual data managers that allow the instantiation of the data structures with the evolving data have been developed to manage these databases. For example, project data manager allows the instantiation of the output information map classes using the *output* module of the *activity-information map* manager. The required product functionality values can be

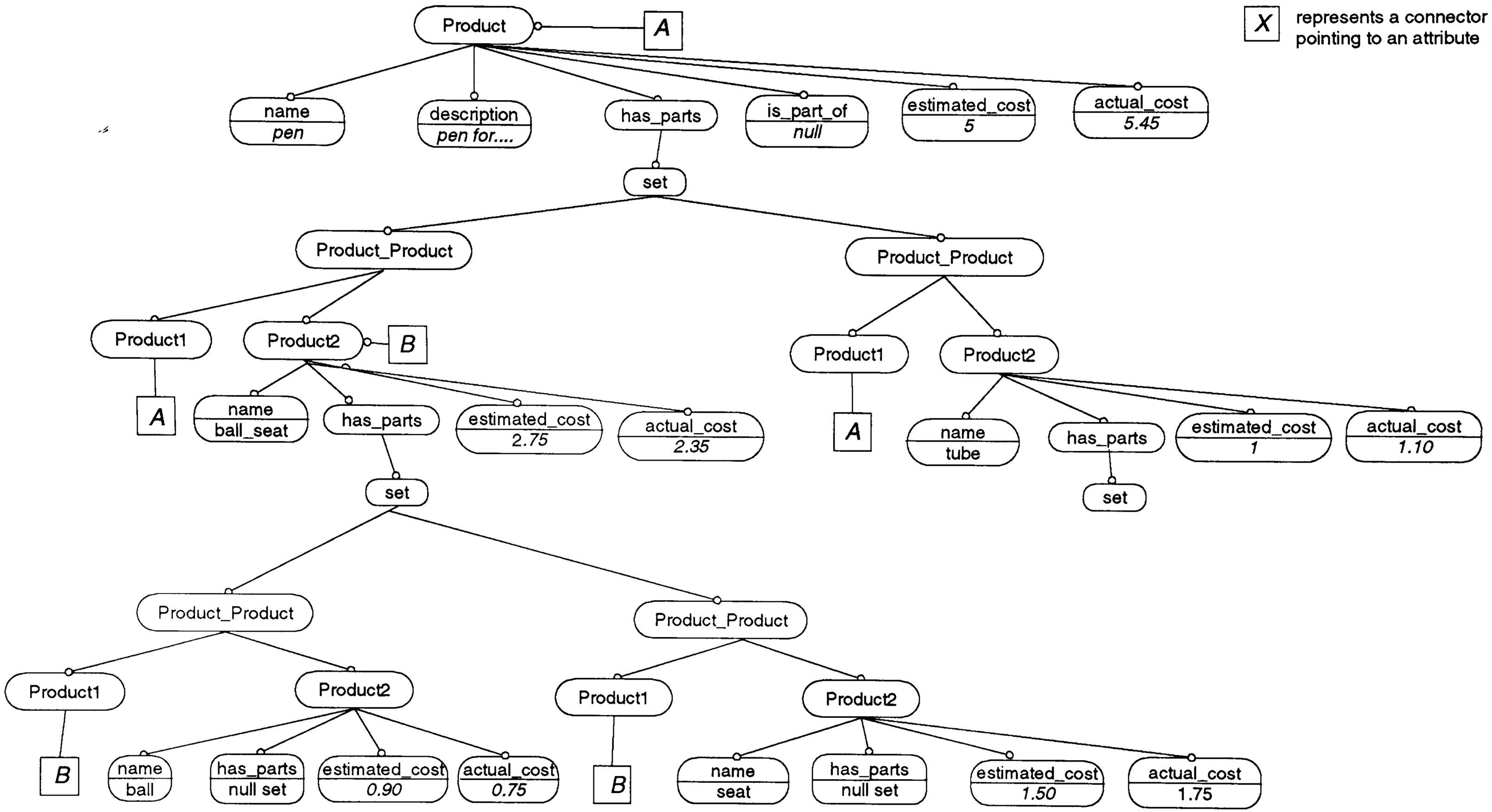


Figure 7.16 Part of the instance of the *product* class showing the cost details

stored as parameter values in the product functionality database, and linked with the activity as information maps using the *control* module of the *activity-information map* manager. These data values would be used to test whether the product design has met the required functionality. Based on the test results, decisions can be taken either to proceed forward or iterate backward in the network of activities. An important feature of the information model is that it represents the hierarchical relationships (*has_parts, is_part_of*) among the project and activities, as well as the information links (*input, output, control, resource*) among the activities, and the sequencing relationships (*successor, predecessor*) among activities. Using the information maps, information flow among the activities would be enhanced and the concurrency among the activities of the PI process can be analysed. The information model represents the necessary information to control the PI process and attaches them to the activity using information maps. The information on the status of the project in terms of target project time against the actual project time, target project cost against the actual project cost; and the status of the output of the project in terms of target product functionality against the achieved product functionality, and target product cost against the actual product cost that are necessary for the management of the PI process can be drawn from the information model. Hence, it can support the management of the PI process.

The data models of the above mentioned databases are integrated into a single global model with the use of the meta-model. The method of integration and the usage of the information model for effective information flow are described in the next chapter.

CHAPTER 8

8. DEVELOPMENT OF THE GLOBAL DATABASE

8.1 INTRODUCTION

In order to manage the product introduction process, the local databases that contain the central management and technology information need to be integrated into a global database. A meta-database that stores the information about the schema of the local databases has been developed towards this integration. A prototype that shows the primary features of the integrated information model has been developed, which include (1) data representations, (2) dynamic data modelling, (3) linking heterogeneous local databases and (4) linking managerial and technical information. In order to demonstrate the use of the integrated information model, the following two tools have been developed: (1) activity dependency analysis tool and (2) communication accelerator; the tools also show how to retrieve the information from the distributed, autonomous, heterogeneous databases.

This chapter describes how the global database representing the integrated information model can be developed and how the primary features of the information model have been achieved. Section 8.2 illustrates the data representation techniques; section 8.3 explains the development of meta-model manager and data model manager that provide dynamic data modelling capabilities. Section 8.4 explains the use of a global database that integrates the distributed and heterogeneous local databases; section 8.5 describes how the managerial and technical information are brought under one roof in

the integrated information model. Finally, Sections 8.6 and 8.7 explain the tools that bring out the use of the integrated information model.

8.2 DATA REPRESENTATIONS

The structural aspects of the various information involved in the product introduction process, and the complex associations among them demands a uniform framework for the treatment of arbitrary user-defined data types such as *'set of Activities'*, *'Activity_InformationMap'* association. Representation mechanisms of such data becomes necessary for the development of the information model. Object-Relational data model is chosen to represent the product introduction information; the reasons for the selection of object-relational database management system (ORDBMS) are described here with an example.

Relational systems are designed to manage only limited types of data, such as integer, floating-point number, string, boolean, data, time and monetary, and they are not designed to manage arbitrary user-defined data types. Relational data models manage a limited set of very important but simple data elements excellently. A central notion of an object-oriented data model is the uniform treatment of arbitrary data types and the facility to add new data types (UniSQL/X 1996). The notions of encapsulation and inheritance in the object-oriented paradigm can reduce the difficulties of design and evolving database. At the same time, object-oriented database management systems are not recommended for transaction-intensive environments such as the introduction of a complex product like an aeroengine.

A simple example - Here is an illustration of how object-relational data models support the migration of a relational infrastructure towards object orientation. Consider the example of an organisation using a matrix style management system. Each employee belongs to one department, but may participate in many product introduction project teams whose membership overlaps departmental boundaries.

A conventional relational model of this situation is likely to use four distinct tables: R_Employee, R_Department, R_Project and R_Proj_Team (Code Sample 1A); and there is no information in the schema to indicate how tables and columns are interrelated. For applications or users to navigate between tables, they must join tables based on foreign key relationships (Code Sample 2A). For example, to obtain a report listing employees and their managers, the R_Employee table can be joined with the R_Department table based on the foreign key relationships, as shown in code sample 2A, making possible a common SQL query.

By virtue of UniSQL[#] 's hybrid data model, the conventional relational structures (Code samples 1A and 2A) can be easily moved from a pure relational database, to the object-relational database, without any technology dislocation (UniSQL 1995). A class-based model makes it much easier to represent the situation inside the database using classes (Code Sample 1B). The advantages of this revised schema are:

1. Relational operations are still valid on this class-based schema. A user or application can still create arbitrary views of the data by using joins. For example, to obtain a list of Employees who are both department managers and project leaders, a common select statement as given in code sample 2B works fine.

UniSQL is an object-relational database of UniSQL, Inc.

Code Sample 1

A. Conventional Relational Schema

```
CREATE TABLE R_Employee
(       emp_id  integer,
        emp_name        string,
        dept_id         integer,
        emp_sal monetary        );

CREATE TABLE R_Department
(       dept_id         integer,
        dept_name       string,
        dept_budget     monetary,
        dept_manager    integer   );

CREATE TABLE R_Project
(       Proj_id         integer,
        proj_name       string,
        proj_lead       integer   );


CREATE TABLE R_Proj_Team
(       Proj_id         integer,
        emp_id          integer   );
```

B. Class-based Schema

```
CREATE  CLASS  Employee;
CREATE  CLASS  Departmnet;
CREATE  CLASS  Project;

ALTER CLASS  Employee  ADD ATTRIBUTE
(       emp_name        string,
        emp_sal         monetary,
       emp_dept         Department,
        emp_projs       set_of (Project)  );

ALTER CLASS  Department ADD ATTRIBUTE
(       dept_name       string,
        dept_budget     monetary,
        dept_manager    Employee,
        dept_staff      set_of (Employee)  );

ALTER CLASS  Project   ADD ATTRIBUTE
(       proj_name       string,
        proj_leader     Employee,
        proj_team       set_of (Employee)  );
```

2. Joins are, however, very performance intensive operations that the object-relational model allows users or applications to skip. The domain of the attribute emp_dept of the class Employee is "Department". The value actually stored for emp_dept is an Object Identifier (OID) of an instance (a record) of the class Department. The OID uniquely identifies an instance of a class, providing both its type and physical location. Thus OIDs can be considered a kind of pointer. Navigating from one object to its component parts is, therefore, simply a matter of following these pointers. Thus, "pointer chasing" can be used to navigate complex relationships among objects, which is faster than creating a relational join. Thus, the revised SELECT statement (Code Sample 2C) is shorter, simpler and avoids a join.

Code Sample 2

A. Conventional Relational Join and Query

R_Employee.dept_id = R_Department.dept_id and
R.Department.dept_manager = R_Employee.emp_id

Hence the SQL Query:

```
SELECT  e.emp_name,  m.emp_name
FROM  R_Employee e,  R_Employee m,  R_Department d
WHERE  e.dept_id = d.dept_id  AND  d.dept_manager = m.emp_id;
```

B. Simplified Relational Join and Query

```
SELECT  DISTINCT  emp_name
FROM  Employee e,  DEPARTMENT d,  PROJECT p
WHERE  e = d.dept_manager  and  e = p.proj_leader;
```

C. Optimised Class-based Query With No Join

```
SELECT emp_name,  emp_dept.dept_manager.emp_name  FROM Employee;
```

3. As the OID designates the type of an instance, the database will automatically disallow invalid instance types, which helps to maintain the integrity of the database.
4. Sets are a very powerful construct in object-based data management, but are not supported in pure relational systems. Reoccuring lists or groups must be given their own table in a relational model, which is why there was a fourth table, the Employee-Project table R_Proj_Team, in the first schema (Code Set 1A). The set-value capability of the object model eliminated the need for the table in the class-based schema, and application performance will be substantially improved because it will be unnecessary to join that table to access the information.

According to UniSQL (UniSQL/X 1996), the relational data model could be fully supported as a subset or special case within the object data model. Object-Relational Data Base Management System (ORDBMS) allows complex nested data, such as a bill of materials, to be represented in a natural hierarchy. Data can be retrieved via navigation (pointer chasing), rather than joining tuples in multiple relations. The relational database constructs can be fully supported as a special case within an object-relational database.

Product introduction process has deeply nested data elements with complex relationships. Complex data types such as sets and associations could be represented in the information model as shown in Figures 8.1 and 8.2. In the aeroengine development project, a branch of the product tree that represents the product structure of aeroengine is shown in Figure 8.1; the *set* value is used to store the information of the 'many' components that are involved in an assembly, and the *product_product* associations are used as the domain to represent the relationship between an assembly and its components. Thus the *has_parts* attribute of the *product* class is of the type *set_of product_product* associations. Similarly, '*design stage 2*' (Figure 8.2) is a sub-project; the attribute '*has_parts*' of the *project* class is of the type *set_of project_project* associations; and the *input* and the *control* of the activity are *set_of activity_informationmap* associations.

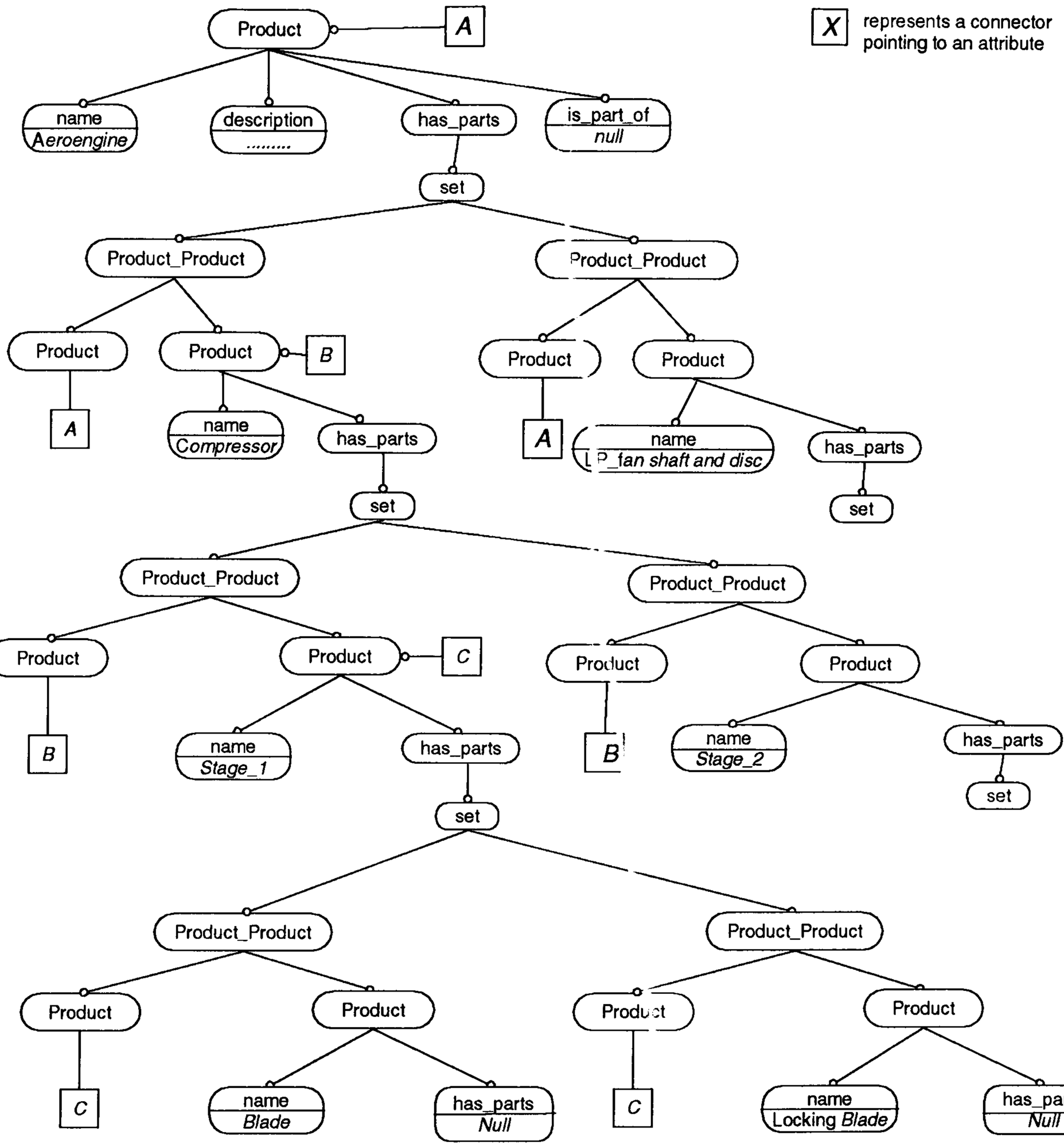


Figure 8.1 A representation of the hierarchical relationship in product structure

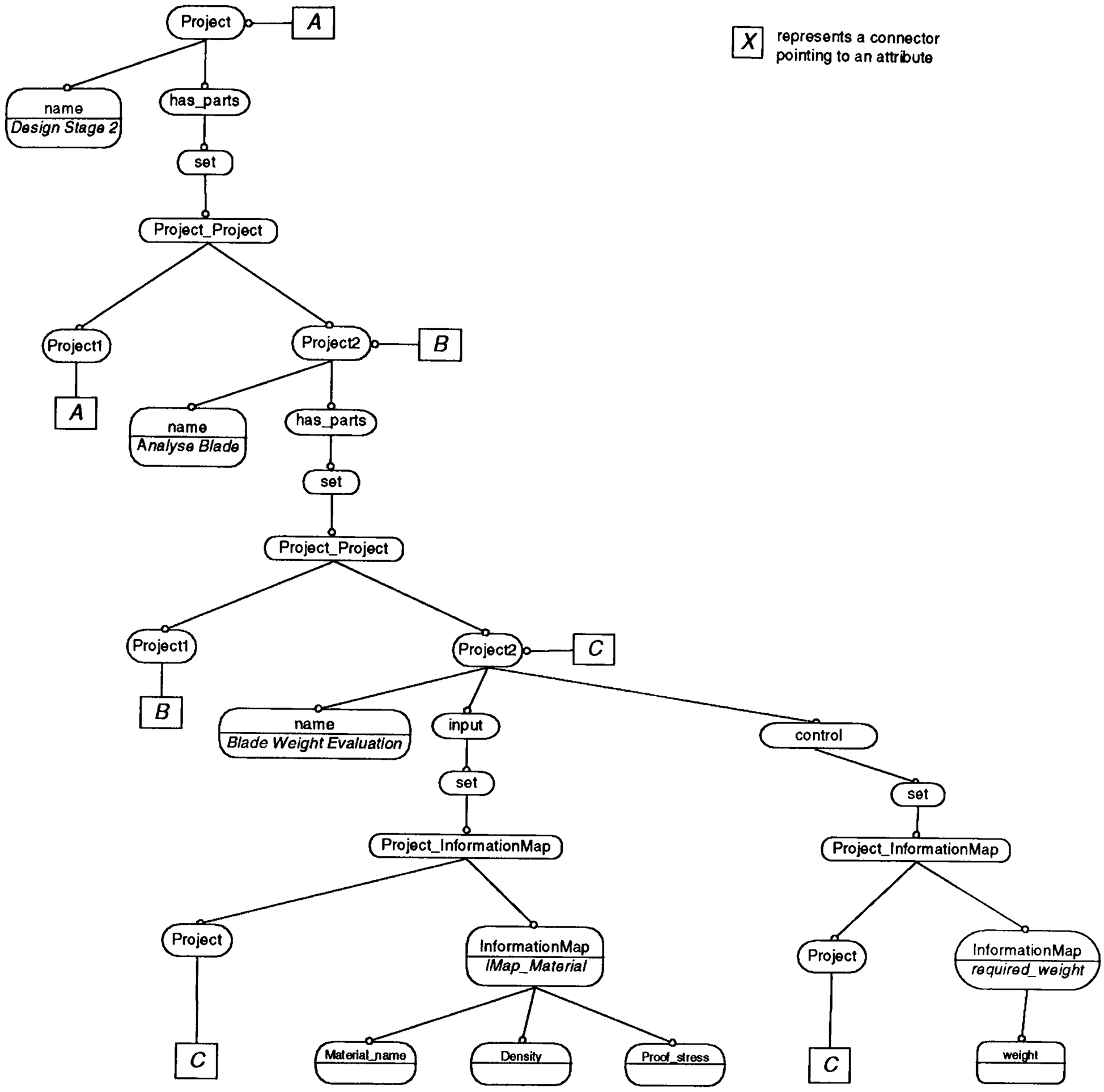


Figure 8.2 A branch of the project information representation showing complex attributes

8.3 DYNAMIC DATA MODELLING

During the product introduction process, the information model may be refined in several ways: by addition of design objects and by modifications to the object class definitions of existing objects (i.e. adding increasing detail). To accommodate this refinement, in other words, to enable dynamic data modelling, the classes need to be created in the relevant databases. Thus, it becomes necessary to store the information

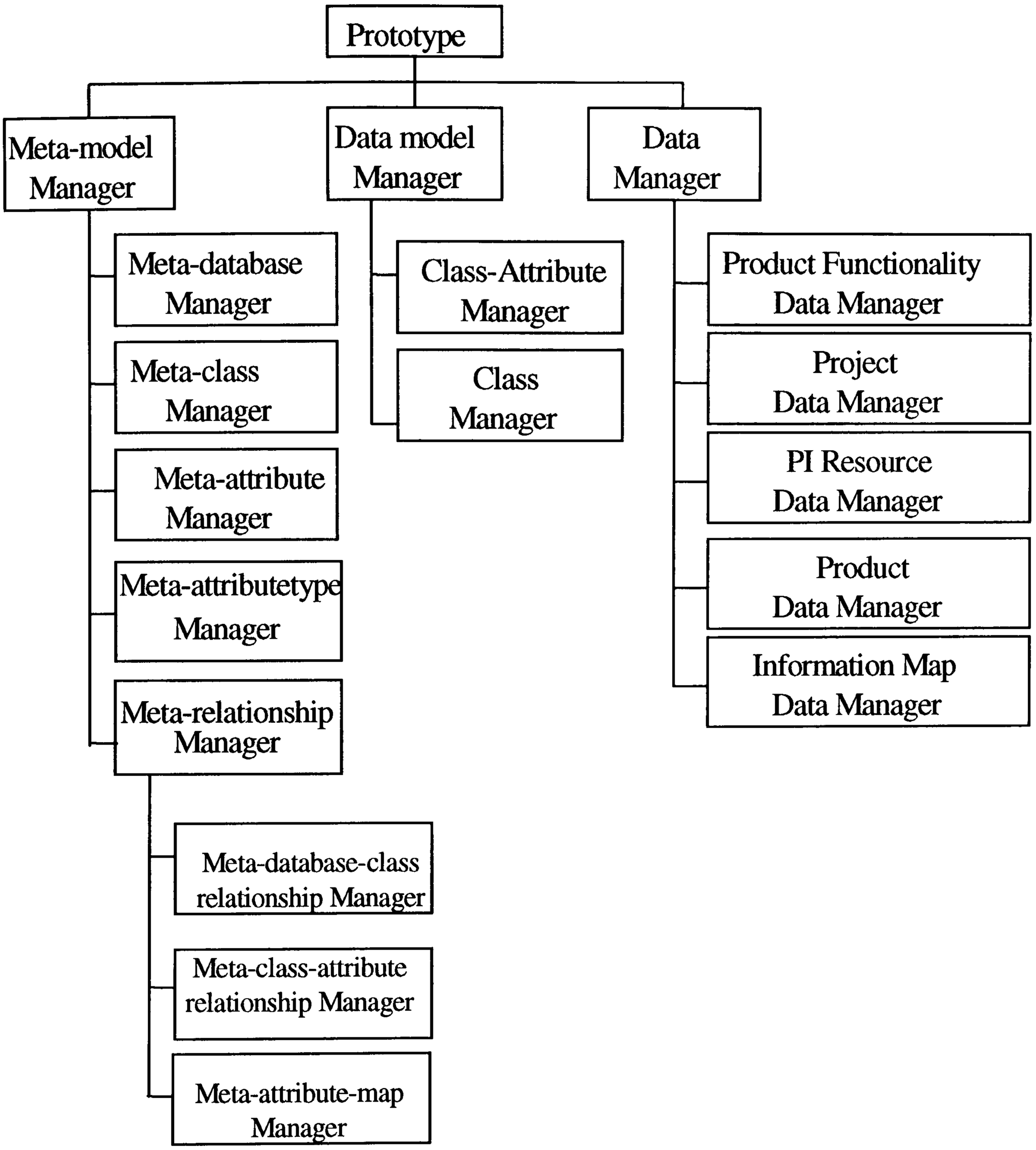


Figure 8.3 Prototype modules (Level 2)

about the data definition (schema) of the databases - product introduction project database, product functionality database, product introduction resource database, product database and information map database, and the relationships between a class and the database to which it belongs; meta-database is created for this purpose. Meta-model manager and data model manager are the two software modules (Figure 8.3) developed for implementing dynamic data modelling facility.

8.3.1 Meta-model manager

The meta-model manager is a tool which can be invoked by a database administrator or an authorised user for interactively adding, modifying, deleting, querying and viewing the information about the data definition of the databases, tables/classes, attributes, attribute types and their relationships that comprise the data model. These information about the data definitions are stored in a database named 'meta-database'; classes (or data structures) that are defined to store the data definition information are: *mdb, MetaClass, MetaAttribute, MetaAttributeType, mdbcl* and *MetaInformation*; the attributes of these classes and operations on them are shown in Table 8.1. The meta-model of product functionality database, product introduction project database, product introduction resource database, product database and information map database would be maintained using the meta-model manager (Thirupathi and Roy 1998).

Table 8.1 Properties and operations of meta-model classes

Data definition of	Class Name in Meta-model	Properties / Attributes	Operations
Database	Mdb	name, description, path, tag	New, Open, Close, DBClasses
Class	Metaclass	name, description, superclass, properties	New, Open, Delete, Superclass,
Attribute	Metaattribute	name, description, type, constraint	New, Existing, Delete
Attribute type	Simpledatatype	name	New, Existing, Delete
Relationship between database and class	Mdbcl	database name, class name	Add a class, Drop a class, View
Relationship between class and attribute	MetaInformation	Class name, attribute name	Add an attribute, Drop an attribute, Set primary key, View

Integration of the local databases necessitates registration of local databases under a global database. The name and location of each of the local databases, along with other information required for accessing the entities in these databases need to be stored in the meta-database and this information will be used when the local database is registered under the global database. In the earliest stages of the PI process, a simple model may be used (consisting of few, high level objects). When more detail (additional, lower-level objects) need to be added to the model as the information becomes available, the following actions would be carried out:

1. meta-model manager would be used to input the information about the structure; this information would be stored in the meta-database and
2. using the information stored in meta-database, data model manager can generate the structure in the relevant database.

Figure 8.4 shows how the data flows when the meta-database is populated. The sub-modules of the meta-model manager that are used for populating the meta-database are:

1 meta-class maintenance module

2 meta-attribute maintenance module

3 meta-attribute type maintenance module

4 meta-relationship maintenance module

4.1 meta-database_class relationship maintenance module

4.2 meta-class_attributes relationship maintenance module

5 meta-attribute map maintenance module.

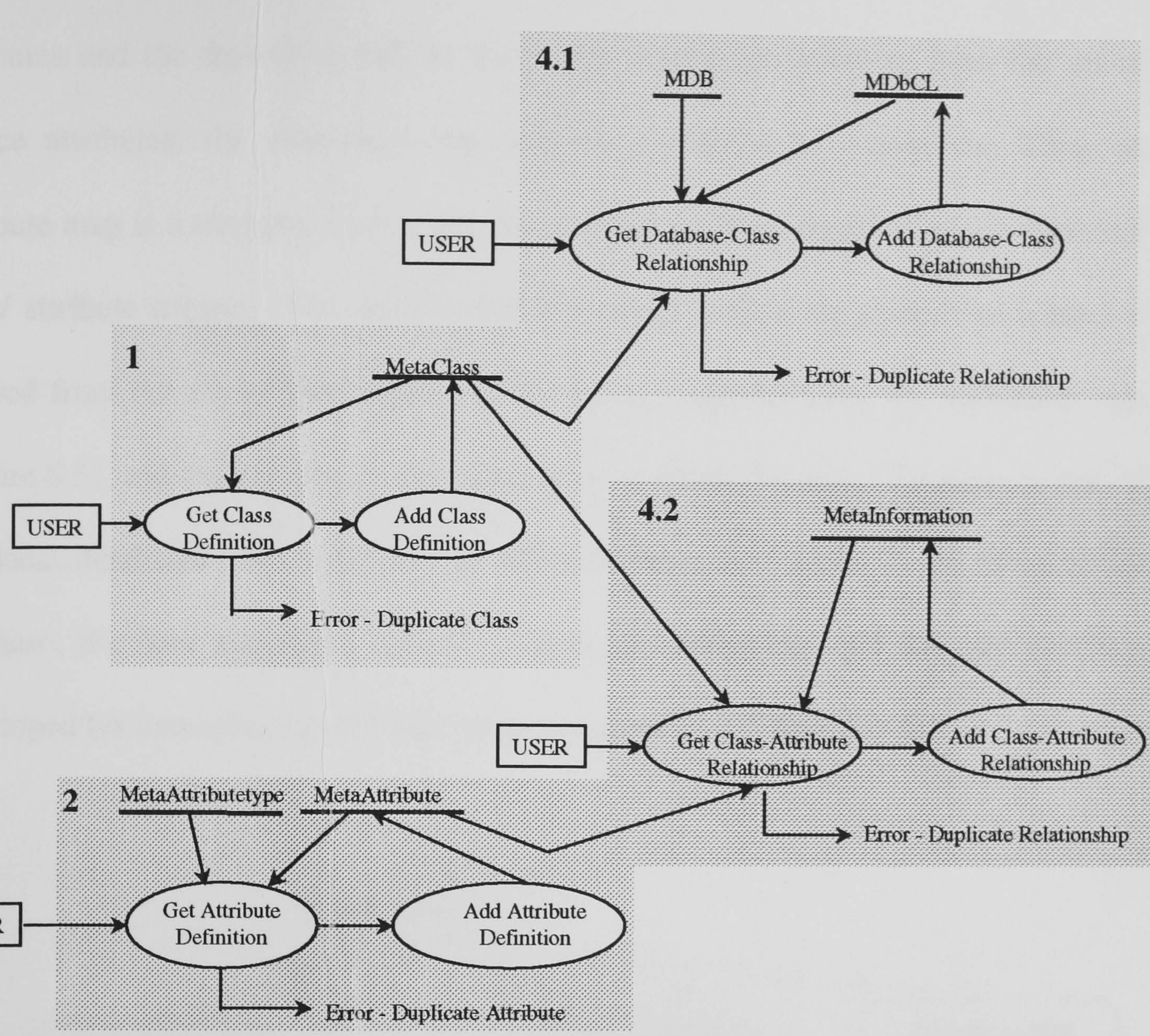


Figure 8.4 Data flow diagram of the meta-model manager

Attribute map manager - is a module of the meta-model manager that performs a set of services related to the information that pertains to the attribute maps. In the information model there are three possible derivation routes for an attribute value. It may be directly assigned a value, it may be derived from the attributes of its own object using a method or it may be derived from another attribute value (or) set of attribute values elsewhere in the model. In case of derived attributes, the relationship between the derived attribute and the attributes from which it is derived is represented using an attribute map; the derived attribute is termed as destination attribute in further discussions. An attribute map consists of a destination attribute, a set of source

attributes and the derivation rule to derive the destination attribute from the set of source attributes; the derivation rule may be a function or expression. Thus, an attribute map is a mapping from a database/ class/ attribute triplet to many database/ class/ attribute triplets. The value of the first triplet (called the destination triplet) is derived from the set of later triplets (called source triplets) using the derivation rule (Figure 8.5), and the database in the triplet may be either the global database or one of the local databases. The definition of attribute maps are stored in the meta-model database; the data structures defined to store the definitions and the user-interface developed for managing the attribute maps are given in APPENDIX D.

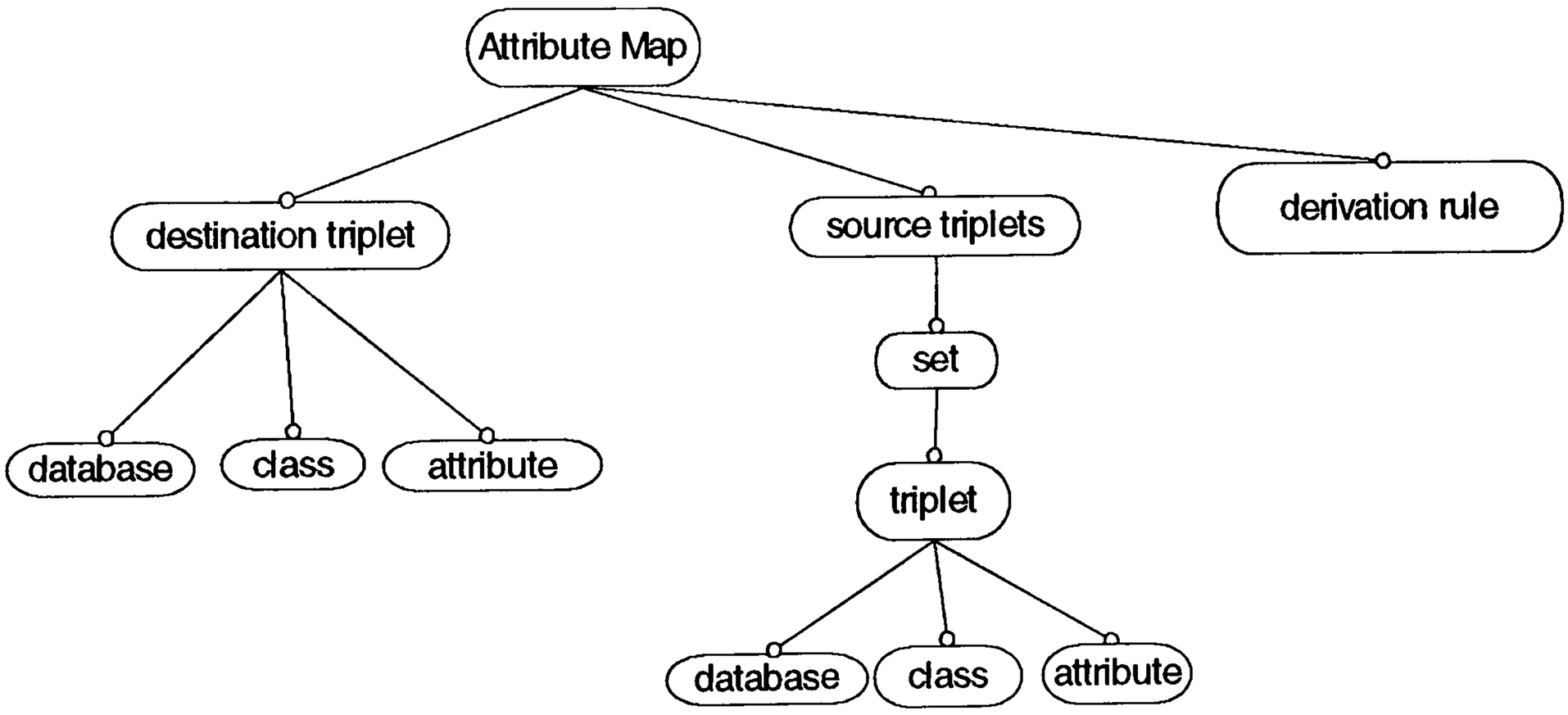


Figure 8.5 Structure of an attribute map

Table 8.2 shows three examples for attribute maps; the first two are of the simple type and the third one is of complex type. Figures 8.6 and 8.7 show how the attribute maps 1 and 3 of Table 8.2 would be represented in the information model.

Table 8.2 Sample attribute maps

No	Attribute Map
1	Imapdb.mdb.Input_Stage_Weight.Blade_weight =(Proddb.mdb.Designed_Weight.Weight)
2	Imapdb.mdb.Input_Stage_Weight.Locking_Blade_weight =(Proddb.mdb.Designed_Weight.Weight)
3	Imapdb.mdb.Output_stage_weight.Weight =(Imapdb.mdb.Input_Stage_Weight.Blade_number * Imapdb.mdb.Input_Stage_Weight.Blade_weight + Imapdb.mdb.Input_Stage_Weight.Locking_Blade_number * Imapdb.mdb.Input_Stage_Weight.Locking_Blade_weight)

The attribute map is a guide that the attribute map manager uses to locate information in the various repositories. The map is used by the project data (communication) manager at the completion of an activity and at the instantiation of an attribute to trace the dependent activities in order to alert the resources of the dependent activities for the availability of a piece of information, thus enabling an early start in activities that are critical to a successful product release.

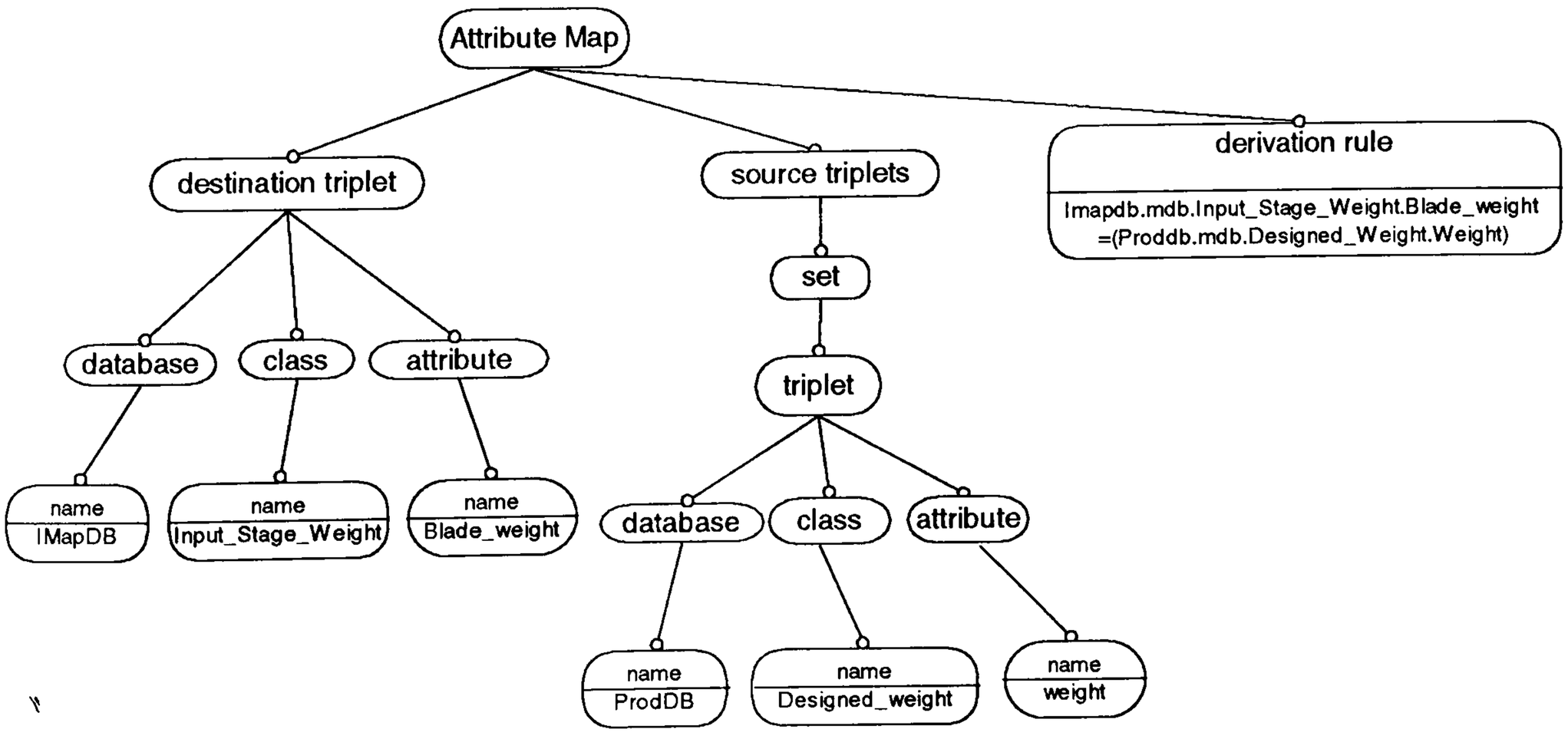
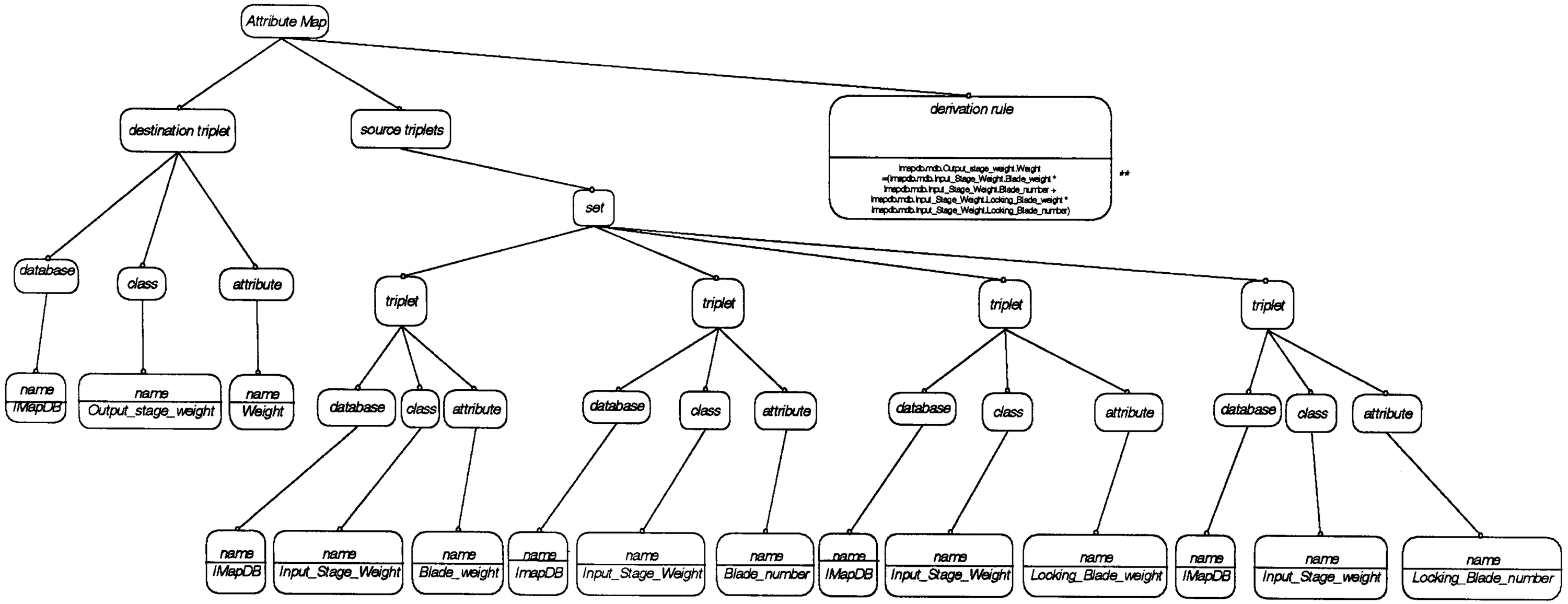


Figure 8.6 An illustration of the representation of a simple attribute map



** Imapdb.mdb.Output_stage_weight.Weight =(Imapdb.mdb.Input_Stage_Weight.Blade_weight * Imapdb.mdb.Input_Stage_Weight.Blade_number + Imapdb.mdb.Input_Stage_Weight.Locking_Blade_weight * Imapdb.mdb.Input_Stage_Weight.Locking_Blade_number)

Figure 8.7 An illustration for the representation of a complex attribute map

8.3.2 Data model manager

The data model manager is used for interactively creating, modifying, deleting and viewing the structure of tables/classes of the various local databases. It uses the information stored in the meta-database to perform the operations *create* class, *drop* class, *add an attribute* to an existing class, *delete an attribute* from a class, and *set a primary key* for a class in databases. The steps for creation of a class are given in Figure 8.8 and the corresponding data flow diagram is shown in Figure 8.9.

1. Select database
2. Read the class definition of the class to be generated
3. For the class, read the definition of related attributes
4. Frame 'create class' query
5. Set the access scope as the required database
6. Perform the query
7. If not Error then class is created

Figure 8.8 Algorithm for dynamic class creation

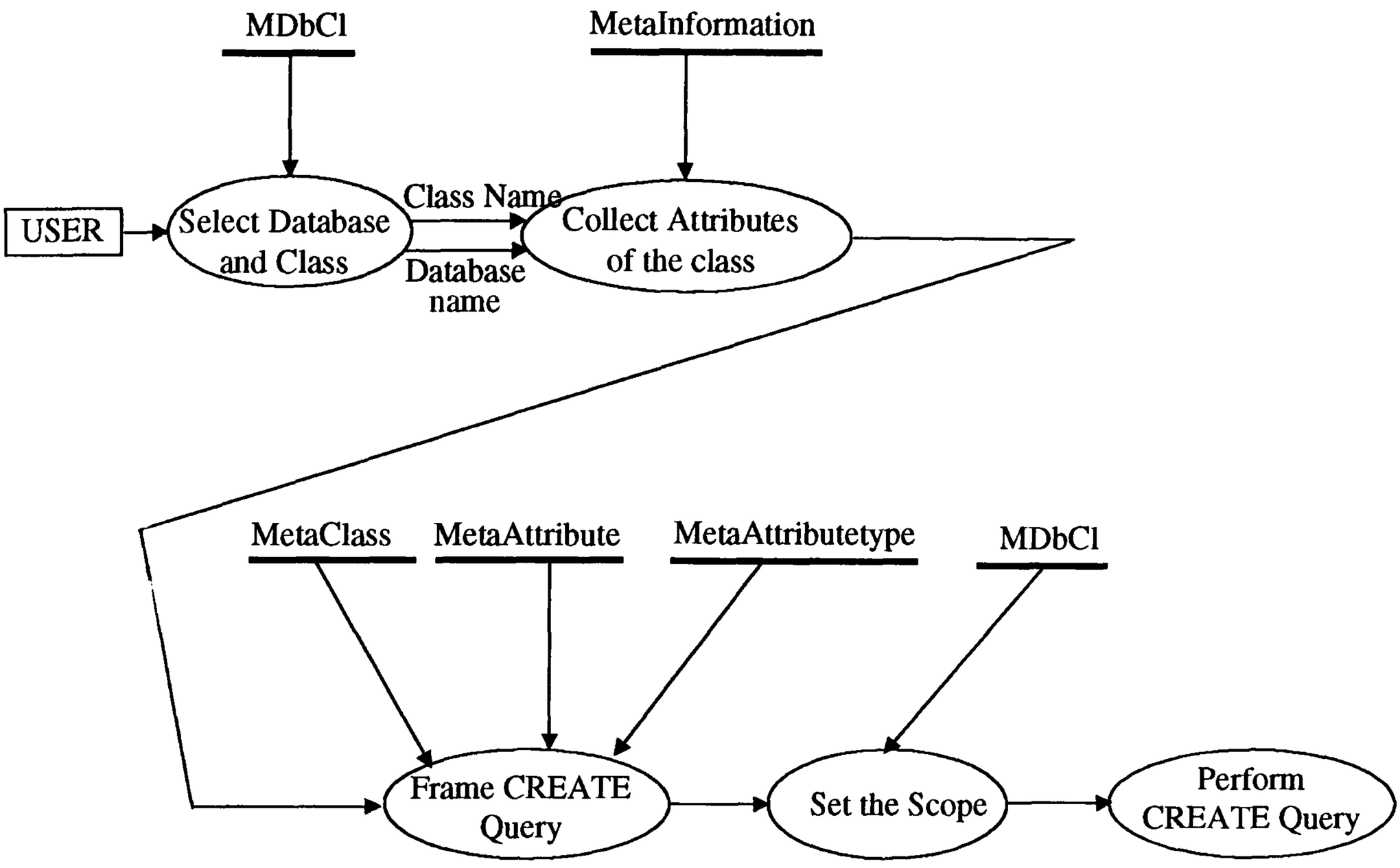


Figure 8.9 Data flow diagram for the creation of a class

A class in the local database can be created either from the corresponding local database or from the global database, i.e. integrated database. From the local database, a stored procedure that performs any data definition query is fired to execute the CREATE query. In the second case, EVALUATE ON statement that points to the local database is included before the CREATE query and then the query (Figure 8.10) is executed from the global database to create the class in the specified local database.

```
EVALUATE ON LDB <local database>

'CREATE .....'
```

Figure 8.10 CREATE QUERY from the global database

Apart from generating and maintaining the structures of tables/classes of the various databases, it would be necessary to allocate, modify, delete access rights on data for data security. Data structures have been defined to store the information on users and access rights, but user interface to manage these information is yet to be developed.

The meta model manager and data model manager have been implemented in a prototype using MS-ACCESS and Visual Basic; the forms designed and libraries developed are given in APPENDIX D.

8.4 LINKING HETEROGENEOUS DATABASES

The local databases involved in the product introduction process would be heterogeneous (Table 8.3). These heterogeneous local databases can be linked using a concept called 'integration through unified schema'. Information specific to the local databases, such as column and table names in the individual schema represented in meta-schema and stored in meta-database, is needed only by the database designer or administrator or an authorised user who creates the unified schema for the global database. The meta-database provides the information for creating the unified schema that integrates the local databases that are distributed, autonomous and heterogeneous. Distributed means that the storage and processing for local databases may be located on different host computers that are linked by a network. Autonomous means that each local database system is an independent database system, and has its own administrator, policies and user community. Heterogeneous means that each local database may be managed by a distinct database management system. SQL features that are available in the local database serve as the basis for accessing the data in the

local database. Global database can be developed as a unified relational and object-oriented layer that sits on top of the five different local databases (Table 8.3) and links and unifies the heterogeneous databases.

Table 8.3 List of databases

No	Name of the database	Type	DB Name
1	Product Introduction Project Database	Relational	ProcDB
2	Product Functionality Database	Object-Relational	PfuDB
3	Product Introduction Resource Database	Relational	RsrcDB
4	Product Database	Object-Relational	ProdDB
5	Information Map Database	Object-Relational	IMapDB

Figure 8.11 provides an overview of the database architecture. The global database provides users with a consolidated view of the information that is stored in the individual local databases. As a result, data in all of the local databases can be accessed as if it belonged to a single database. In other words, the global database provides users with a single, seamless application view of multiple heterogeneous databases. Users will have access to standard data management facilities, such as querying, database views and access authorisation, just as if they were using a single database management system, and queries and updates made from the global database would be automatically applied to data in the local databases. The procedure for creating the global database involves the following steps:

1. Registering local databases
2. Creating proxies
3. Creating virtual classes

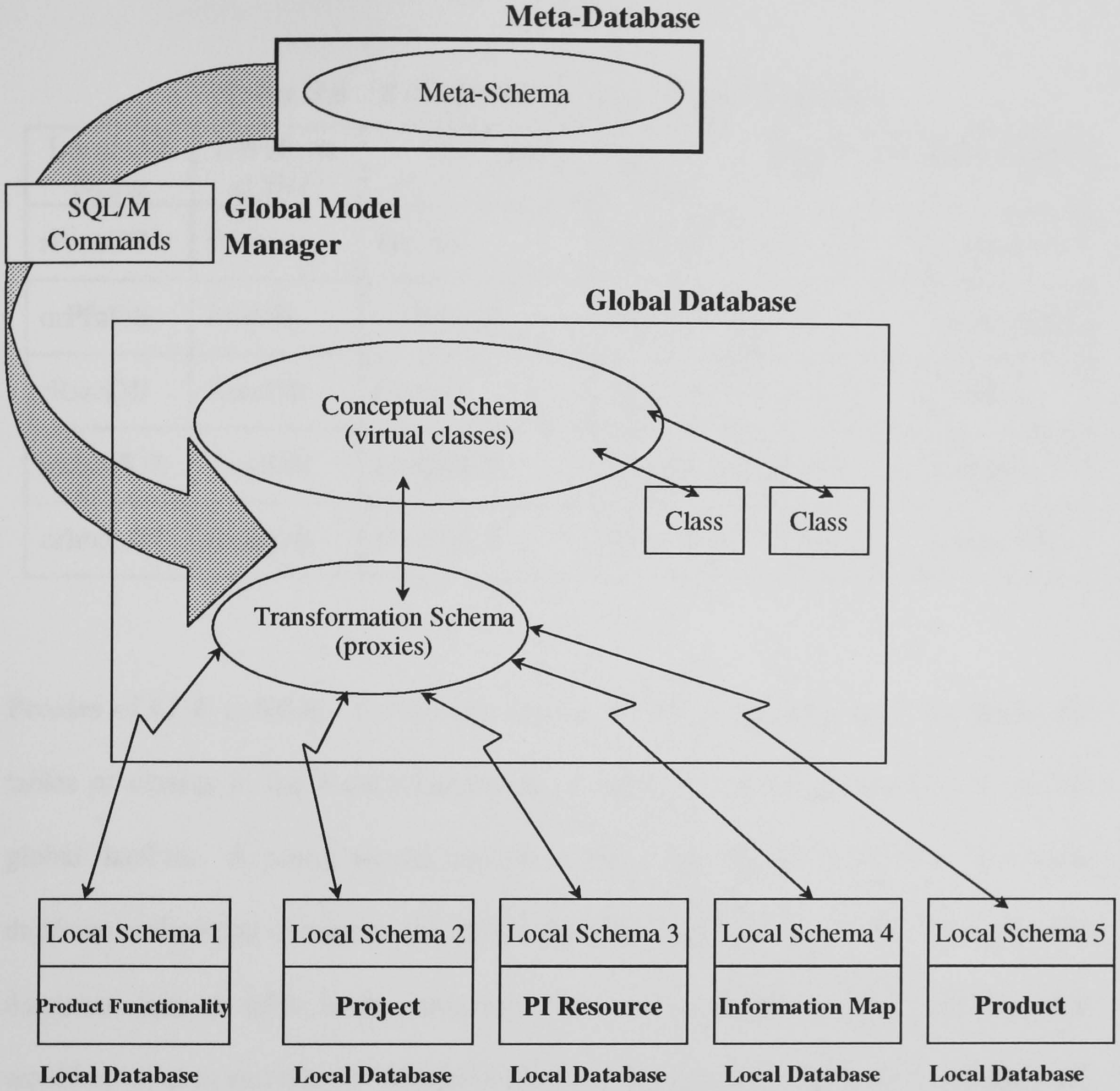


Figure 8.11 Overview of the database architecture

Registering local databases - Information stored in local database entities (classes and/or tables) can be accessed after the local database is registered. Registration of the local database identifies its name and location (Table 8.4), along with other information needed for user access from the global database (Figure G.1 in APPENDIX G).

Table 8.4 Registration details of local databases

Local DB Name	DB Name at Host	DB Type	Host Name	User Name	Password
rProcDB	ProcDB	Oracle	Fha219	Alan_L	Mount*(
orPfuDB	PfuDB	UniSQLX	Fha212	Carris_N	HouseSM
rRsrcDB	RsrcDB	Oracle	Fha219	Rowse_C	Villa77
orProdDB	ProdDB	UniSQLX	Fha212	Olive_J	Flag97
orImapDB	ImapDB	UniSQLX	Fha212	Taylor_P	Aum*Mu

Proxies of LDB entities - Proxies are created in the global database to transform the tables or classes in the local databases to a form that is readily understood by the global database. A proxy would provide a direct link between the data in a local database and virtual classes in the global database, and can be created for each local database class or table to be accessed. While creating the proxies, meta-database would be used to provide the details of the class or table and the attributes of the local database. A proxy would be created with two distinct parts – the schema definition and the query specification. The schema definition contains an ordered list of attribute names and domains. A single query specification is defined to retrieve data from the local database entity that the proxy represents, corresponding to the ordered list of attributes in the schema definition (Figure 8.12).

```
create proxy project_px on ldb rProcDB
(
    id              integer,
    name            char(40),
    description     char(250),
    esd             date,
    asd             date,
    cost            double,
    ..........
)
as
    select num, name, description, exp_st_date, act_st_date,
            duration, cost,.....  from project;

create proxy product_px on ldb orProdDB
(
    part_id         char(10),
    description     char(50),
    cost            double
)
as
    select part_id, description, cost from product;
```

Figure 8.12 Illustrations for creating proxies

Virtual classes – The relationships between entities in the local databases are captured by unifying the data retrieved by proxies into a single entity called a virtual class. In addition, a virtual class may reference data from other virtual classes or from classes in the global database. Figure 8.13 gives an illustration for creating a virtual class. The definition of a virtual class serves to unify the information that is accessed through proxies, such that data from several different local database entities in one or more local databases can be retrieved with a single query. Virtual classes define the conceptual schema of a global database. A virtual class is composed of a schema definition, but more than one query specification is allowed. Data is viewed from a virtual class via queries on other entities, which may be located in the local database or the global database. Proxies and virtual classes of the global database can be queried

and updated, which makes it simple for users to employ the single, unified schema of the global database and maintain data that is stored in multiple local databases.

```
Create vclass delayed_activities on ldb rProcDB
(
        activity_id                 integer,
        name                        char(40),
        description                 char(250),
        expected_start_date         date,
        actual_start_date           date,
        delay_in_days               integer
)
        object_id (activity_id)
as
        select id, name, description, esd, asd, asd-esd
        from project_px  where asd > esd;
```

Figure 8.13 An illustration of creating a virtual class

Since a query or update on a virtual class may involve access to a number of other entities in the global database, the appropriate authorisation privileges must be granted to users and owners of the virtual class. The same privileges must also be authorised on entities in the local databases that are accessed by proxies on behalf of the virtual class. Privileges can be granted and revoked by owners of the proxies and virtual classes, or by the global database administrator. The special issues in authorisation on global database entities are detailed in APPENDIX G (under section G.4).

8.4.1 Global model manager

As the schemata of the product database and resource database are of evolving nature, it becomes necessary to create proxies, virtual classes dynamically and also set the access permissions for the users dynamically to the newly created elements of the database. To provide this facility, it is necessary to store the information on proxies,

virtual classes, users and their access permissions, and the relationships among proxies, attributes, virtual classes and users. Similar to the meta-model of the local databases, classes such as *meta-proxies, meta-vclass, meta-users, meta-permissions* need to be defined to store this information. The existing *meta-attribute* class of the meta-model can be used for storing the definition of the attributes of the proxies and virtual classes also. Thus, the meta-model manager would include a meta-global-model manager that will perform the functions of capturing the information on the structure of proxies and virtual classes. This module of the meta-model manager would have the following modules: meta-proxies manager, meta-vclass manager, meta-user manager and meta-relationships manager. Using the information stored in the meta-classes, the global model manager (Figure 8.14) would create the proxies, virtual classes of the global database that integrates the local databases and set the access rights for the users to access the data. Thus, meta-database stores the structure representing the schema of the local databases and the global database, and the global model manager is the system integrator (Figure 8.14) that integrates all the local databases using the information stored in the meta-database.

8.5 DATA MANAGER

The data managers 'product functionality data manager, project data manager, resource data manager, product data manager and information map data manager' to manage the respective databases have been developed. An user interface for the overall information model manager has been developed and it makes use of these data managers. As a part of the project data manager, tools to view the project structure data and the information content of the activities in tree views have been developed.

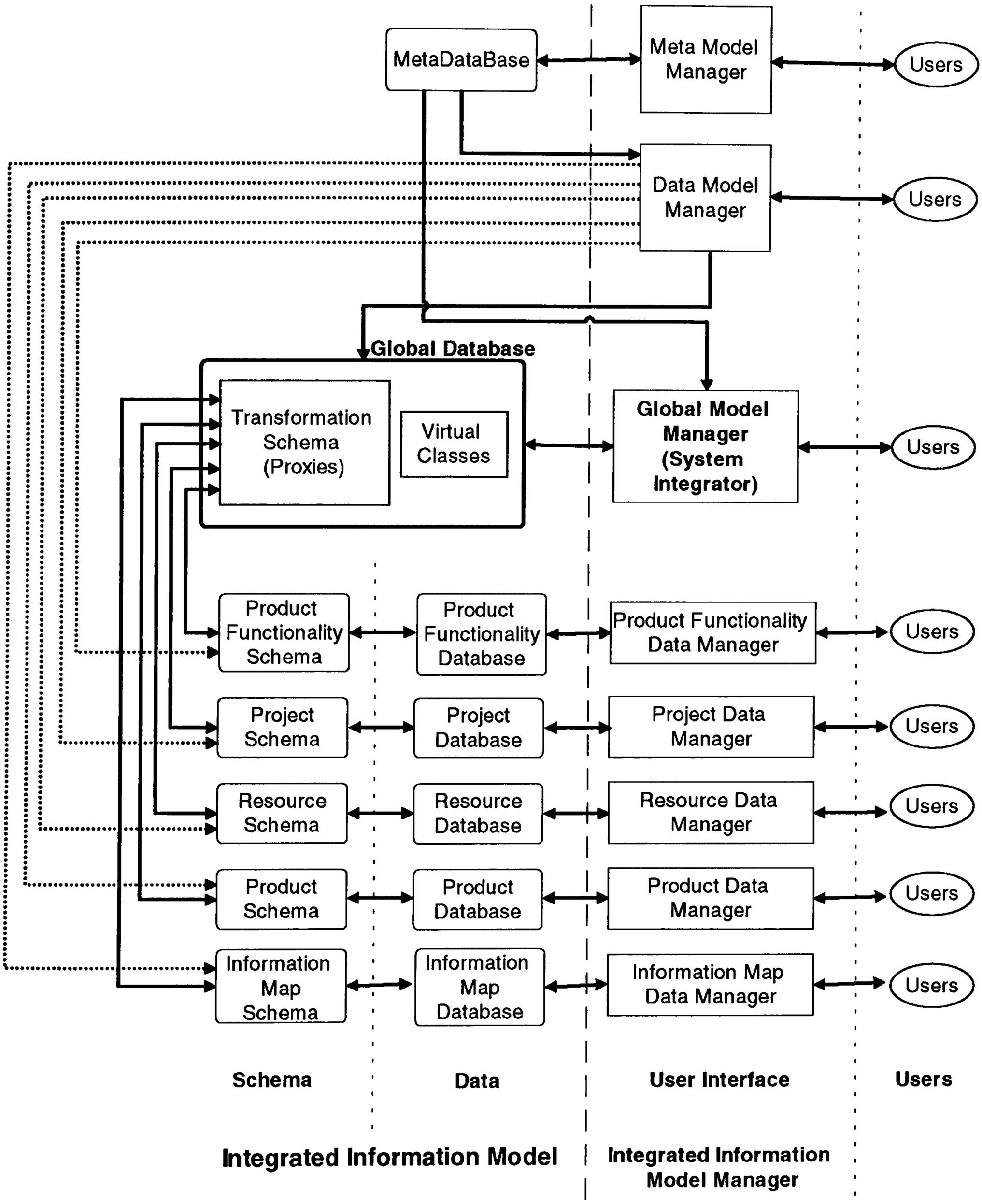


Figure 8.14 Relationships between the information model and the information model manager (Detail level)

Sample Screens that show the user interface of the prototype are included in APPENDIX D.

Managerial information such as information on schedules, duration, resources, cost are represented in the *activity* class as attribute values. Technical information such as product functionality and product information associated with an activity are represented using information maps, as attributes of the *activity* class depending on the role that the information play with respect to the activity. The information assumes roles such as input, control and output with respect to an activity. In order to provide a link between the managerial and technical information, structure of the information maps and the derivation routes of attributes have been represented in the meta-model, and information maps are associated with the activity based on the role they play with respect to the activity. Thus, the detailed link between activity and product information follows the following chain: 'activity → information map classes → information map class structure from meta-model → attribute maps from meta-model → product information classes → product information'. The link between activity and product functionality, and activity and resource also follow similar chains. The link between managerial and technical information is essential in the product introduction process as it is the output of the activities i.e. the technical information that controls the decisions in managing the project. Figures D.11 to D.13 in APPENDIX D show how the information map structures and instances in one user-interface form can be handled through the activity-information map data manager (of the project data manager).

8.6 COMMUNICATION ACCELERATION

Data is created by an activity to be used by someone else. Once created, it should be moved on, it should flow to the activity that is going to use it (Stark 1992). Providing appropriate product design information to the project groups as soon as it is available will allow them to get an early start in activities that are critical to a successful product release. The pursuit of reduced product development cycle time is likely to be sufficiently important to make communication acceleration an important information processing function (Rosenthal 1992).

An algorithm has been developed and implemented to answer the question of who should be alerted when the output data of an activity becomes available. In order to understand the algorithm, it is necessary to know about the types of dependencies among activities; there are two types of dependencies among activities defined in this research work:- direct dependency and indirect dependency.

Direct dependency:- Activity Y is directly dependent on activity X when there is at least one class or attribute that is an output from X and is also an input to Y (Figure 8.15).

Indirect dependency:- Activity Y is indirectly dependent on activity X, when there is at least one information class 'C' or attribute 'A' that is an output from X, and there is an attribute of the input of Y that can be derived from 'C' or 'A' (Figure 8.16).

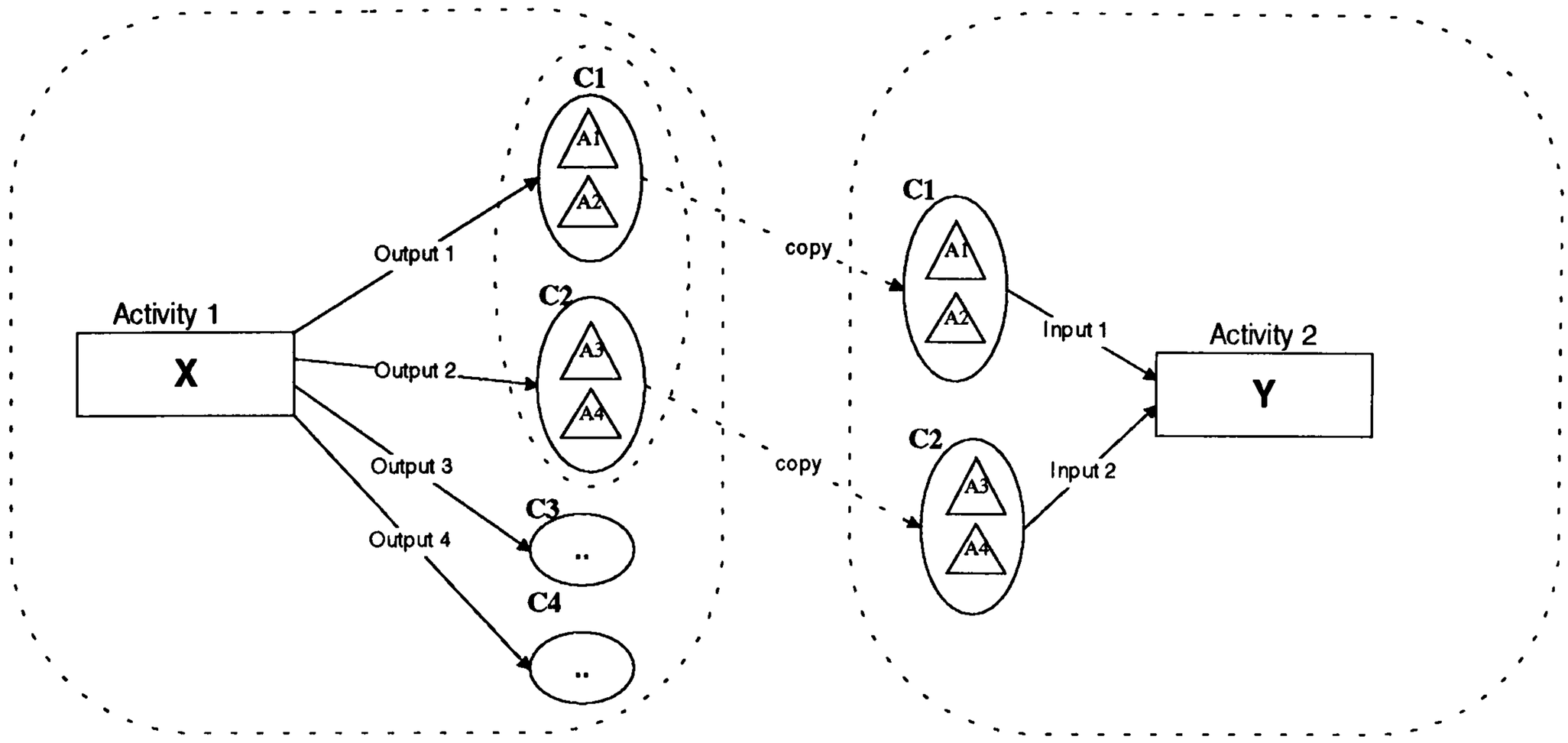


Figure 8.15 Direct dependency between activities (based on class)

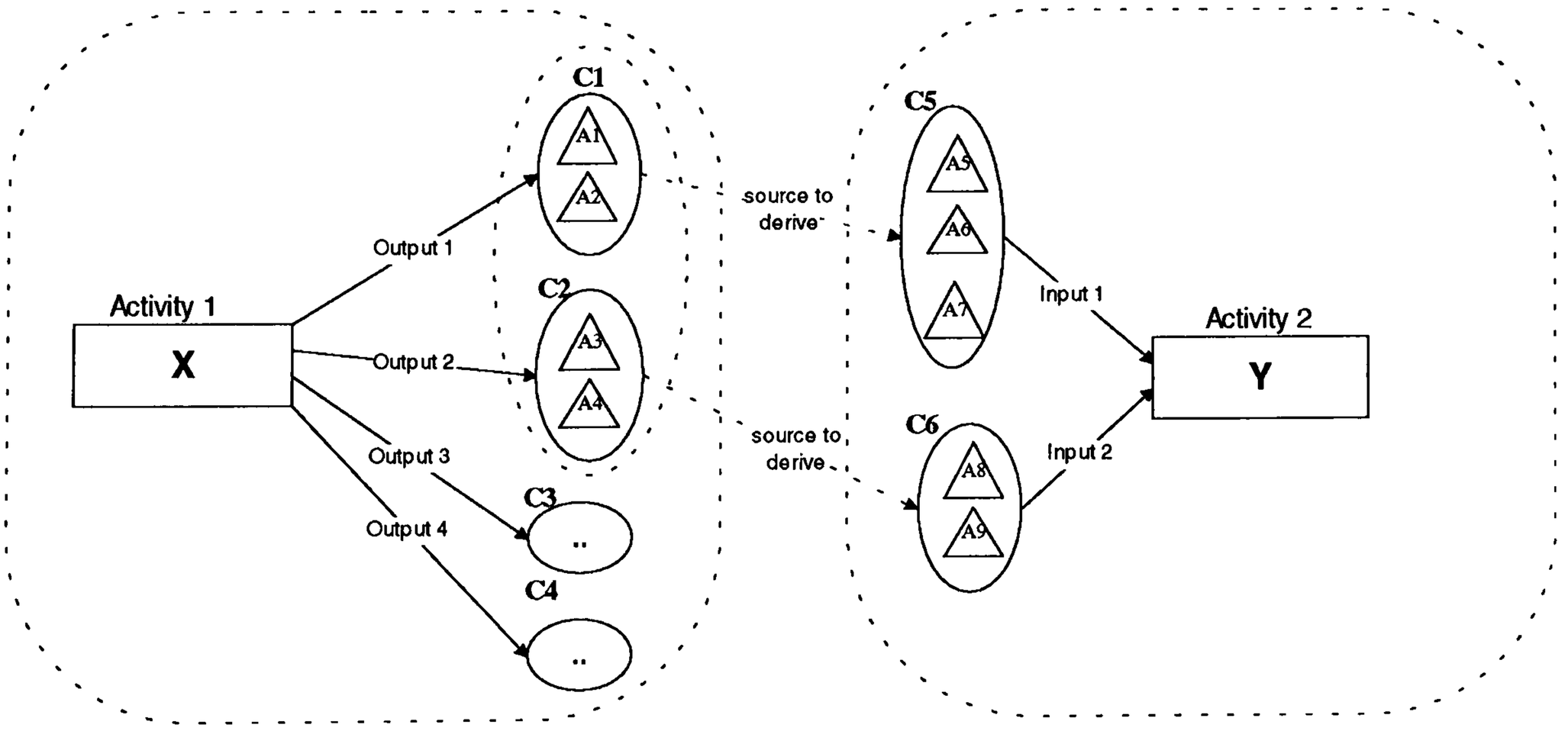


Figure 8.16 Indirect dependency between activities

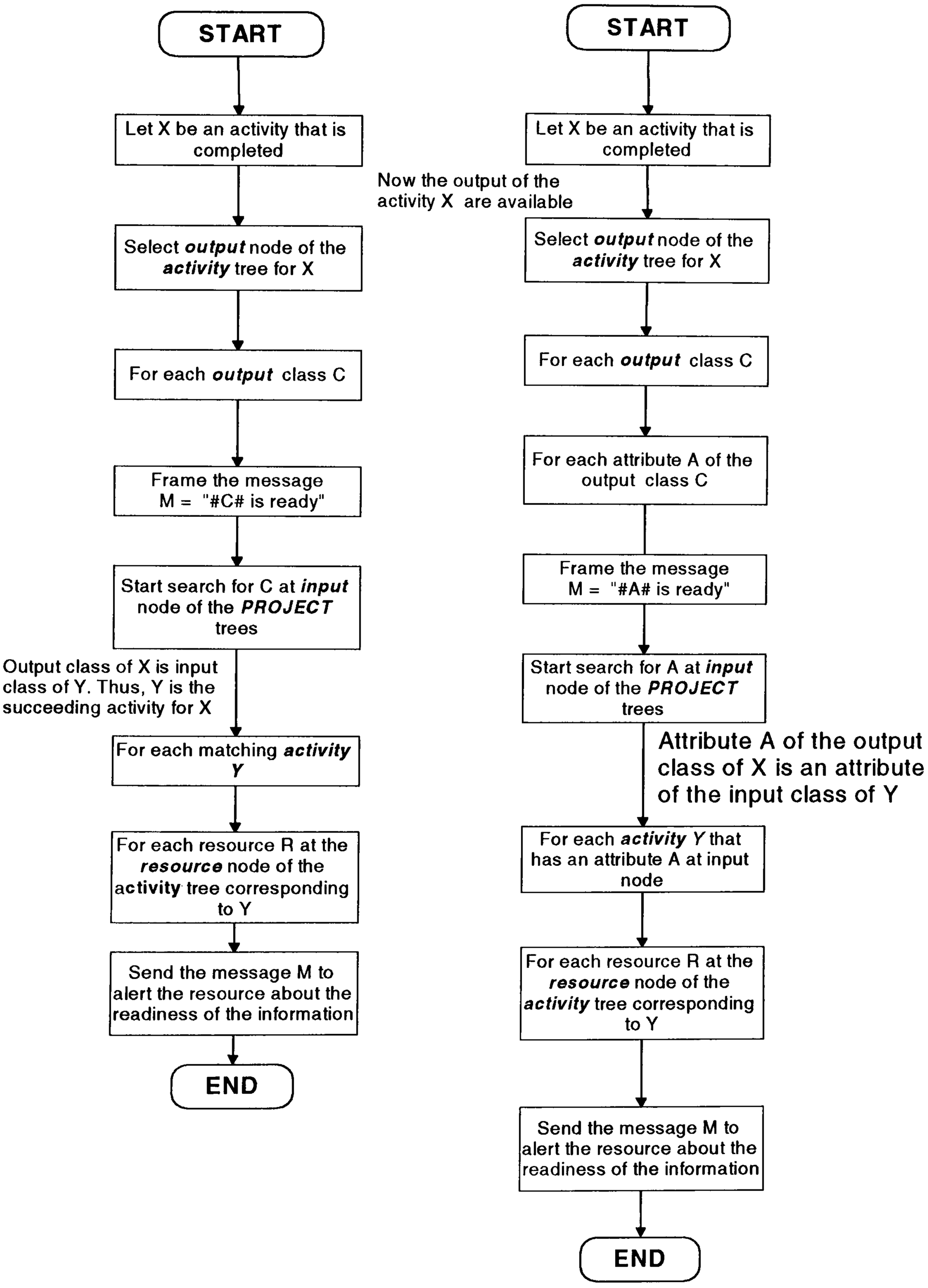
In case of direct dependent activities, the algorithm for communication acceleration is straightforward as shown in Figure 8.17; whereas, in case of indirect dependent activities, it is necessary to use the attribute maps defined in the meta-model to arrive at the dependent activities. The algorithm designed for the communication acceleration for indirectly dependent activities has four passes (Figures 8.18, 8.19), and it makes use of the information from project database, meta-database and resource database (Table 8.5).

As a part of the implementation of the communication accelerator, a tool that analyses the dependencies among activities (detailed under Section 8.7) has been designed and implemented in the prototype. The communication accelerator can be implemented using triggers, which can be defined to provide appropriate information to the teams as soon as the value of a source attribute becomes available. Such triggers facilitate the coordination of project work beyond individual departments and plants and guarantee the integrity of the results.

8.7 ACTIVITY-DEPENDENCY BASED ON INFORMATION CONTENT

Using the information maps attached to the activities, the role of the information (input / output / constraint / resource) with respect to an activity, and the attribute maps stored in the meta-database, the dependencies among activities can be automatically analysed. As a result of the analysis, direct and indirect dependent activities in the forward and backward directions of sequencing can be identified. The idea behind the analysis is to find, given an activity A, the activities that use A's output directly; the activities that use the information derived from A's output; the activities that

generate/output A's input; the activities that generate the output that can be used to derive A's input.



(a) Dependency Through Classes **(b) Dependency Through Attributes**

Figure 8.17 Communication acceleration algorithm for direct dependent activities

For a given activity A,

Pass 1: Traverse the tree of activity A, top-down, to find the attributes of the output information map classes; these attributes, at the leaves of the output branch of the activity tree, form the set of source attributes.

Pass 2: For each source attribute, traverse the trees of the attribute map forest in the meta model, bottom-up, to find the dependent attributes i.e. attributes that can be derived from the source attribute.

Pass 3: For each dependent attribute, traverse the trees in the project forest, bottom-up, to search for the activities for which the input information map classes include the dependent attribute. Frame the message "*#source attribute#* is ready to derive the *#dependent attribute#* of *#information map class#*".

Pass 4: For the activities found in pass 3, traverse the trees in the project forest, top-down, to find the resources which should be alerted with the message.

Note :

In this algorithm , the variables given between #s would be substituted with the actual names that they represent. Example of Message: "ProdDB.Designed_weight.weight is ready to derive the ImapDB.Input_Stage_Weight.Blade_Weight".

Figure 8.18 Communication acceleration algorithm for indirectly dependent activities

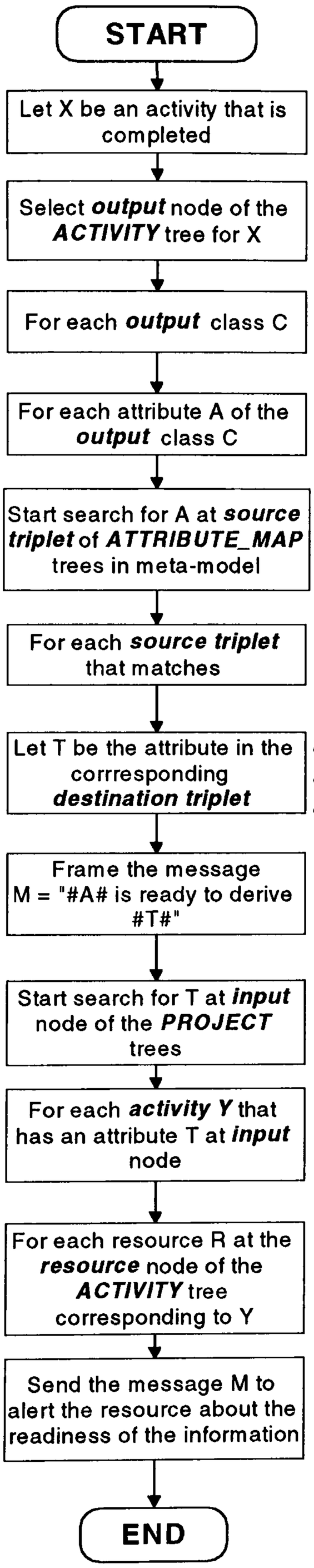


Figure 8.19 Communication acceleration algorithm for indirect dependent activities

Table 8.5 Elements and procedures involved in communication acceleration of indirect dependent activities

Pass	Given	Tool	Branch	Traversal	Search	Result	Location of result
1	Project	Project tree	Output	Top-down	Attributes of output classes	Set of source attributes	Leaves
2	Source attribute	Attribute map forest	Destination triplet	Bottom-up, and then top-down	Destination attribute	Set of dependent attributes	Leaves
3	Dependent attribute	Project forest	Input	Bottom-up	Input classes	Set of projects	Root
4	Project	Project forest	Resource	Top-down	Resources	Set of resources	Leaves

The activity dependency analysis tool identifies the following direct and indirect dependent activities of A:-

Direct Successor - identifies all the activities for which the input information includes at least one output of A. The algorithm behind this analysis would use the following sequence of data :

'Activity A → Output Class → Input Class → Activity B'.

Indirect Successor - identifies all the activities for which the input information can be derived from output of A. The algorithm behind this analysis would use the following sequence of data :

'Activity A → Output Class → Attribute Map → Input Class → Activity B'.

Thus, it makes use of the meta-model that contains the attribute map definitions.

Direct Predecessor - identifies the activities for which the output information includes at least one input of A. The algorithm behind this analysis would use the following sequence of data :

'Activity A → Input Class → Output Class → Activity B'.

Indirect Predecessor - identifies the activities having the output information that are used to derive the input of A. The algorithm behind this analysis would use the following sequence of data :

'Activity A → Input Class → Attribute Map → Output Class → Activity B'.

Thus, it makes use of the meta-model that contains the attribute map definitions.

Activity dependency analysis (based on information content) tool has been designed and implemented in the prototype. The results are shown in Appendix D.

8.8 PROTOTYPE TESTING

The prototype was tested with data from a manufacturer of aeroengines. The information used in the product introduction process within the company is scattered across various software applications: project information on 'MS-Project', design project specification on 'MS-WORD', resources information on 'Access' database, technical accounts on 'MS-EXCEL', financial analysis and budgeting on 'SAP 5', design information on CAD packages, assembly operation information on 'Oracle Relational Database', material information on OPTEGRA, analysis information on various analysis softwares (aerodynamic analysis, steady state stressing (SCO3), vibration analysis including modal analysis and thermal analysis), and personnel

information on 'PROFS'. The information required by the PI process is not directly compatible with the form in which the information currently tends to be available.

A design engineer from the aerospace company used the prototype to populate the databases. In the basic process hierarchy (Figure 8.20), activities 'blade weight evaluation' (identified by 4.2 in Figure 8.20) , 'locking blade weight evaluation' (identified by 4.4 in Figure 8.20) and 'stage 2 weight evaluation' (identified by 3.2 in Figure 8.20), and their associated information (Figure 8.21) were represented. The product functionality considered was the required weight. The required weight of the aeroengine and its hierarchical relationships to the sub-systems (compressor, turbine, stage 2, ...) down to the component (blade) were considered. The prototype modules that were tested are given in Table 8.6 and the representation of the information are shown in Figures 8.22 and 8.23. The user's guide for the prototype is given in APPENDIX D.6.

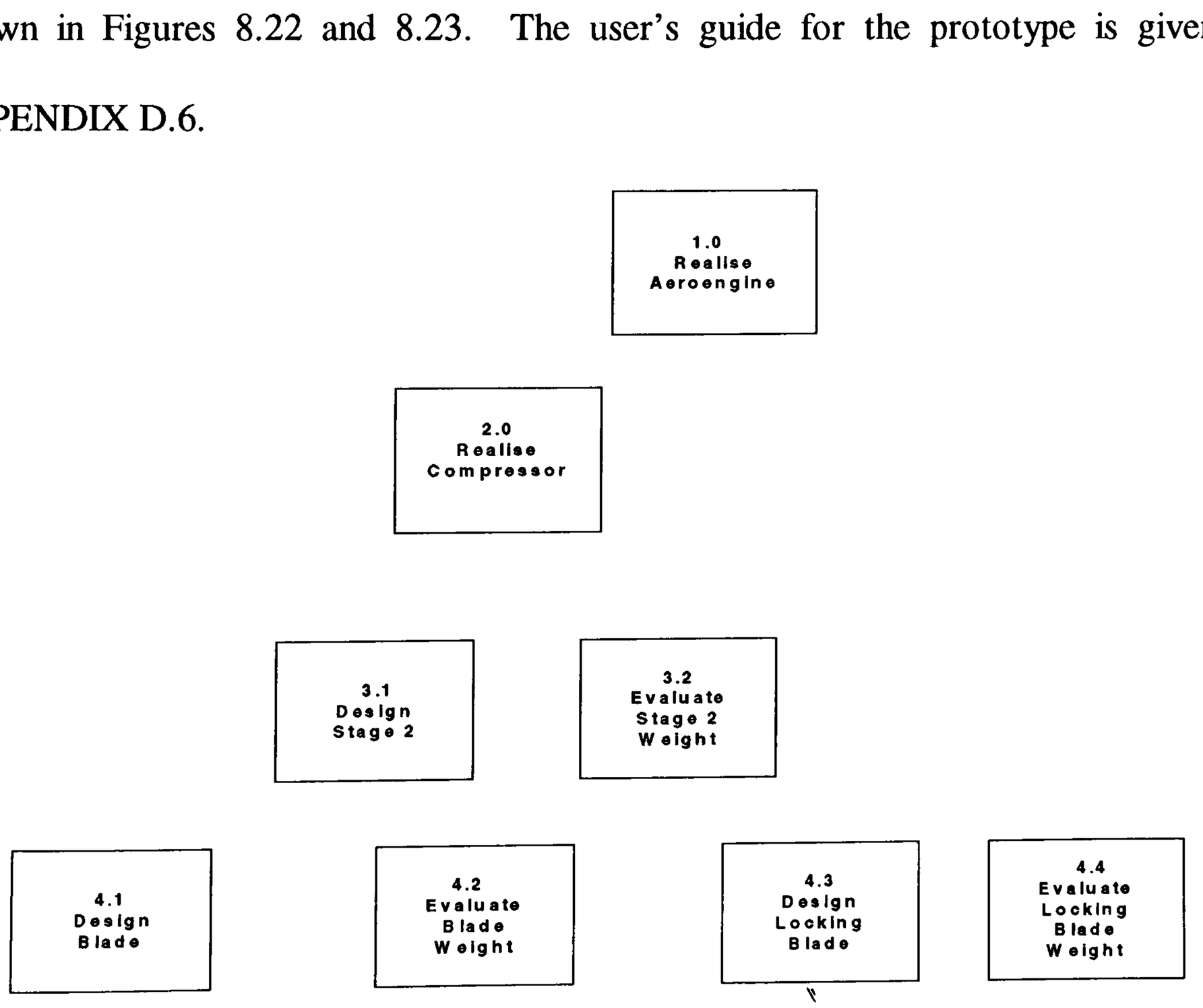
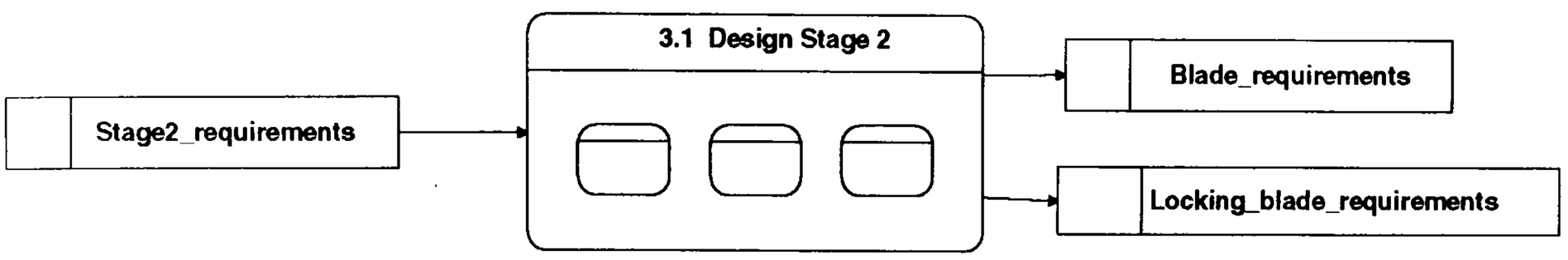
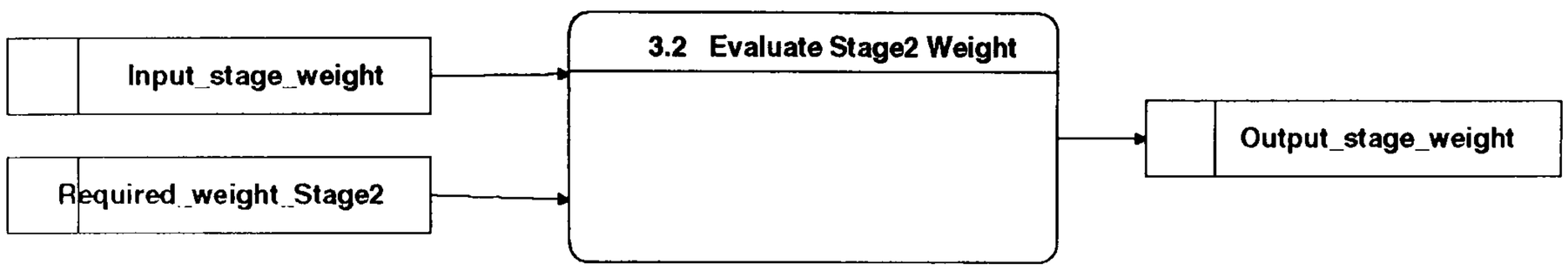
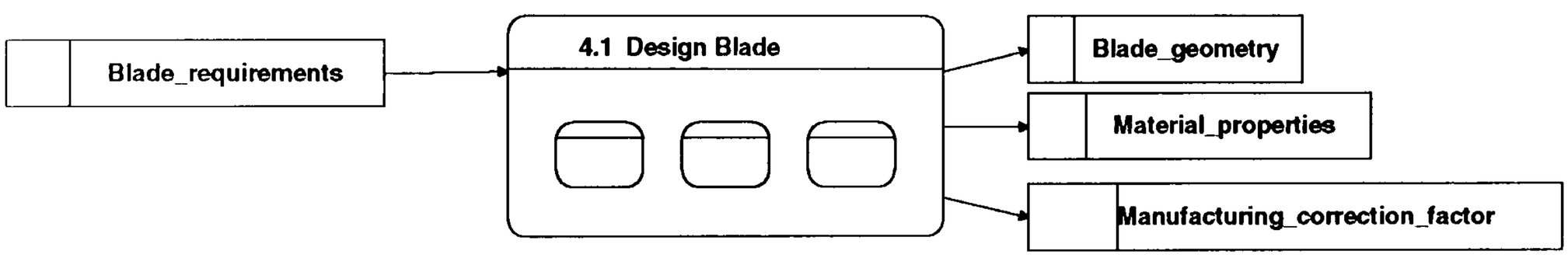
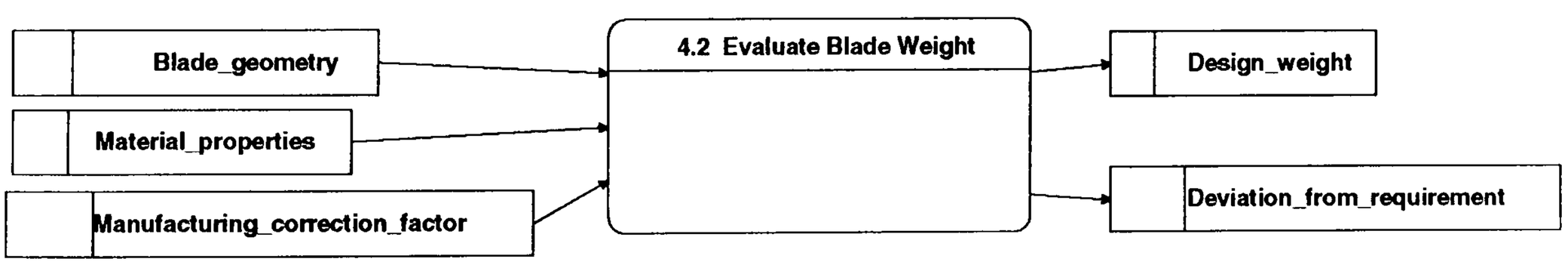


Figure 8.20 Process hierarchy









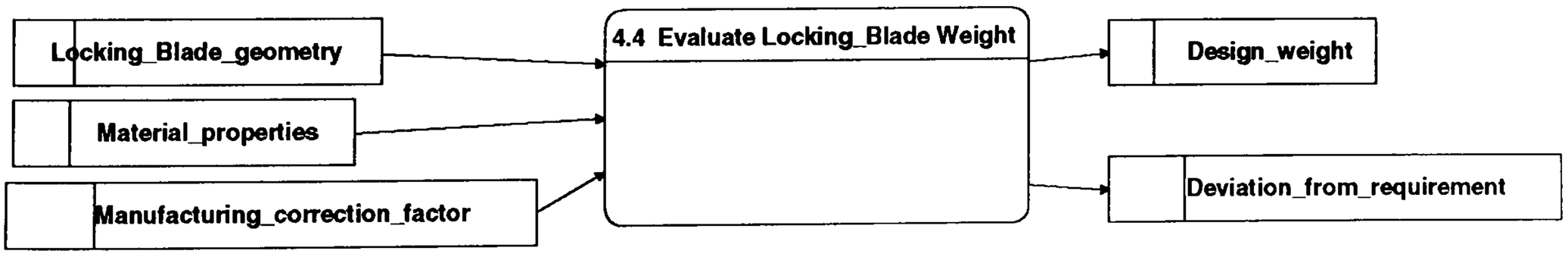


Figure 8.21 Activities and associated information

Table 8.6 Prototype modules used and tested

Data	Description	Prototype Module Used
Activity	Activity name, description	Project Data Manager Project
Input, Control and Output	Define Input, Control and Output Structures:- Class names, description Attribute names, description Class structure	Meta-model Manager Meta-class Manger Meta-atribute Manager Meta-relationship Manager Meta-database-class relationship Manager Meta-class-attribute relationship Manager
	Structure Creation and Modification Class creation Add an attribute to an existing class	Data-model Manager Class Manager Attribute Manager
	Data Entry	
Relate Activity and Structures	Input, Control and Output	Project Data Manager Project-InformationMap
Relate Activity and Structure and Data	Associate instances with activity	
Database Tree	Viewing	Meta-model Manager View Tree View Tool
Define the attribute maps	Consider 3 activities 1,2 and 3. Output of 1 and 2 is input of 3. Define the attribute maps between output of 1 and input of 3, And output of 2 and input of 3, if any.	Meta-model Manager Attribute map Manager Define map
Dependency among activities based on information content	Forward dependency – Successor Backward dependency – Predecessor	Analysis Tools (TreeView) Process Dependencies Information Content Multiple dependencies

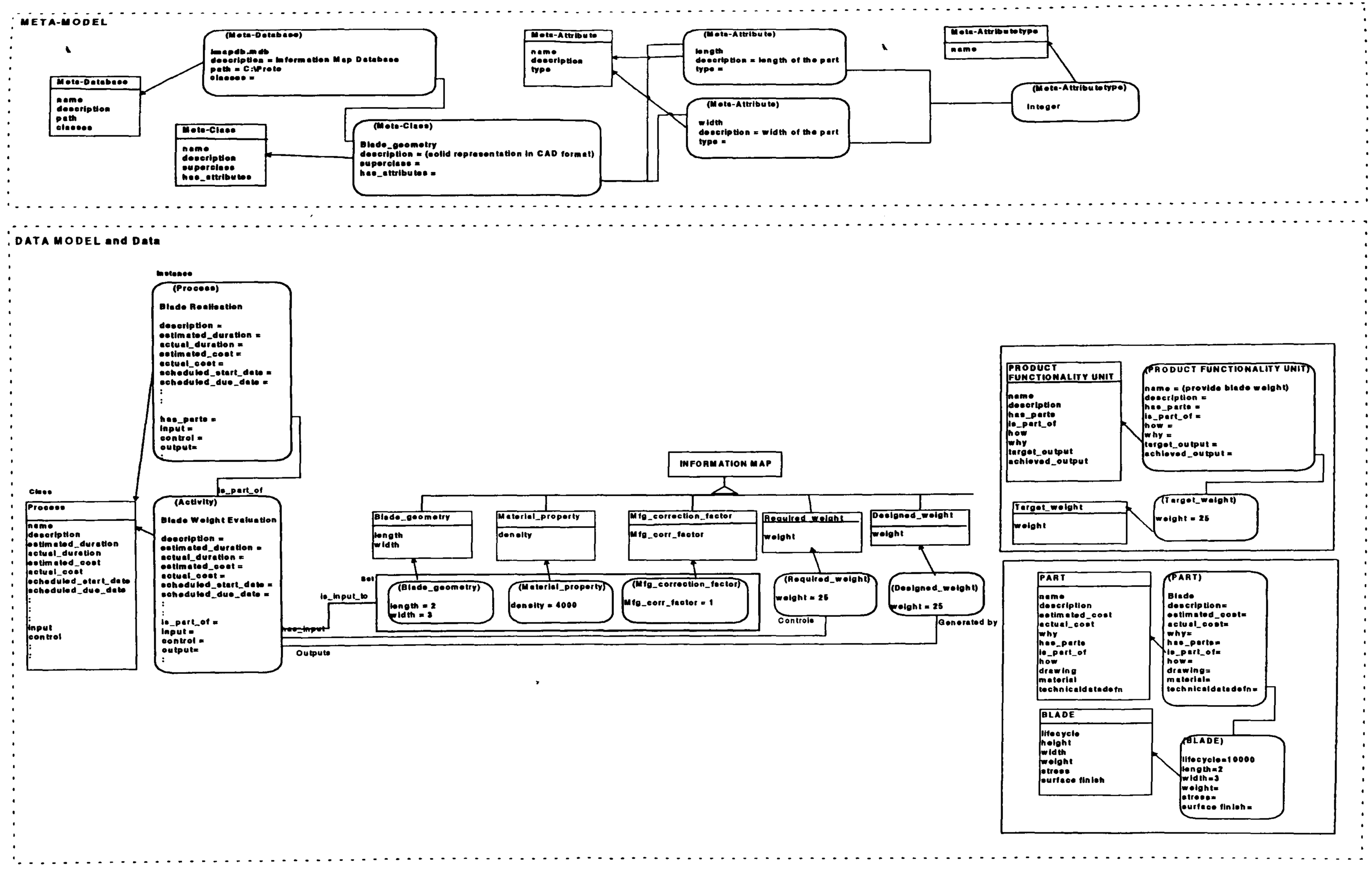


Figure 8.22 Process and data link for the activity 'blade weight evaluation'

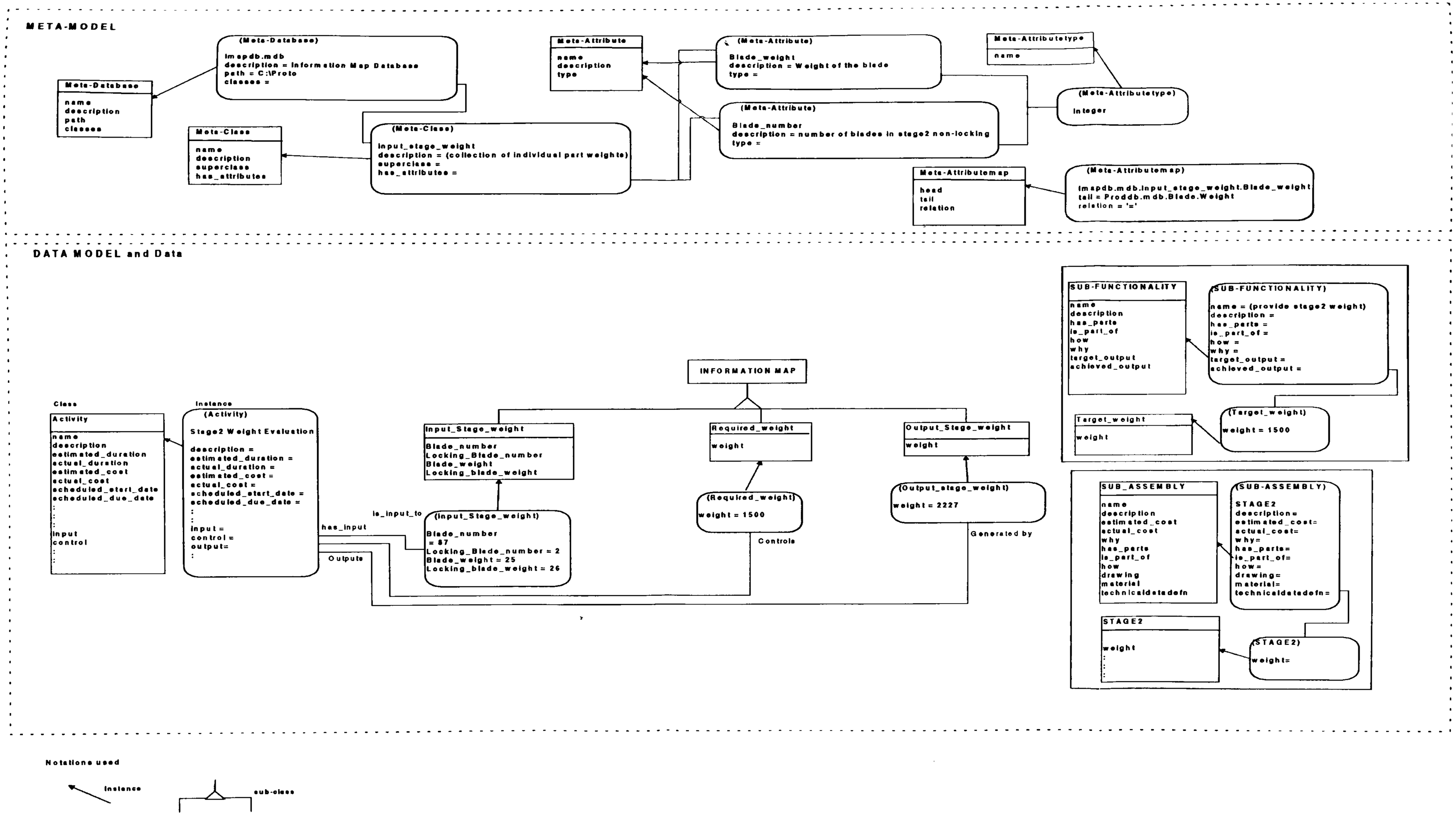


Figure 8.23 Process and data link for the activity 'stage 2 weight evaluation'

8.9 FEEDBACK

The ability of industries to compete in the global market depends on a number of factors, and the ability to capture and manage the knowledge and the associated data is rapidly becoming one of the key factors. The knowledge such as product structure, product information evolve out of the product introduction process. Current product data management systems meet some of the data management needs but most depend on a static data model. Current generations of product data management systems do not provide all the associations between data that the business is demanding in order to gain the best return on the electronic investments. These shortcomings are particularly noticeable within the new product introduction process.

The company was particularly impressed with this research work as the approach goes some way to addressing these shortcomings. Product introduction process is a dynamic process and new data types are created on a regular basis. The current method of handling these in the company is with Excel spreadsheets or local Access databases. These systems are local to the design team and do not form part of the company data base structure. It is felt that a system based on this research work that allows new databases to be created from existing classes would be of great benefit. Furthermore the approach developed in the research of allowing relationships between all business objects not just those formally associated with product data management systems will enable many improvements to be made.

Surveys within the company have shown that over 20% of design time is spent in finding information. The full benefits promised by the present generation product data management systems to reduce this 20% figure are not currently being realised. The

inability of current systems to handle all object relationships, and provide more dynamic data base systems is part of the problem. The company is very pleased with this research work, and they are confident that it has made a real and worthwhile contribution to the product data management problem. From an industrial point of view, it was stated that the sooner these research ideas are developed into a commercial system the better.

8.10 DISCUSSION

The information model allows gathering of product information as the data evolves out of the product introduction activities. The meta-model, a conceptual schema for data models that represent product introduction information, is used to facilitate the capture of the evolving information, as a structural base to provide the information to integrate the data models and as an assistant in understanding the relationships between the classes in different data models as it stores the information on attribute maps; attribute maps represent the derivation routes among the attributes of the classes in different data models. The meta-model manager allows the user to define the necessary class structures and modify the class structure, and relate them to the information on the relevant database, the data model manager creates the classes in the specified database, and the data managers allow the instantiation of these classes with the evolving data. The barrier of linking project management information with the technical information is overcome in the information model by providing the following:

1. information map classes to represent the technical information associated with an activity

2. attributes *input, control* and *output* of the type *set_of activity_informationmap* associations in the class *project (*or *activity)* and
3. a meta-database that stores the meta-structure of the product introduction information, especially the information map class structures.

The proposed information model models the product introduction process information at a number of levels:- managerial level - project model, resources model; technical level - product functionality model, product model; relationship level between managerial level and technical level - information map model; meta-schema level - meta model; schema level - data model; instances level - data; attribute level - attribute maps; and integration level - global model. The sub-models can be integrated into a single global model of the product introduction process information by registering the sub-models, and by creating the proxies and virtual classes with the use of the meta-model and object-relational concepts. User-interfaces for the relevant managers - meta model manager, data model manager, product functionality data manager, product data manager, resources data manager, project data manager and information map manager have been developed and tested using relational database concepts, MS-ACCESS and Visual Basic. However, to reap the full benefits of the design of the information model mentioned in this thesis, it would be necessary to implement the global database using object-relational database concepts.

The conceptual schema of the global database incorporates and unifies the schemata of local databases. Data is stored in local databases, and the proxies that can be created in the global database provide a direct link between the data in a local database and virtual classes in the global database; virtual classes may refer to the data from other

virtual classes or from classes in the global database. As the data is accessed through references (proxies, virtual classes and object identification codes), the data remains consistent. Apart from this, the object-relational database will automatically disallow invalid instance types through the object identification codes 'OIDs' which help to maintain the integrity of the database. Using proxies and virtual classes, data in all of the local databases can be accessed as if it belongs to a single database.

As the information model is going to be a shared one, and represents the structures (project structure, product structure, product functionality structure, resource structure and information map structures) and instances, users who are identified as super users or approved users would be initially generating the structures at the data model level using the data model manager (Figure 8.24). Other users would be given ownership and access privileges for instantiating these structures and accessing the data. As a part of their normal work, design engineers would enter and access the data in the database. When a design engineer identifies a change to be made in the data model, he/she would send a request to a super user or an approved user. The necessary data structures to store the information about users, databases, classes in the databases have been designed (APPENDIX C.5).

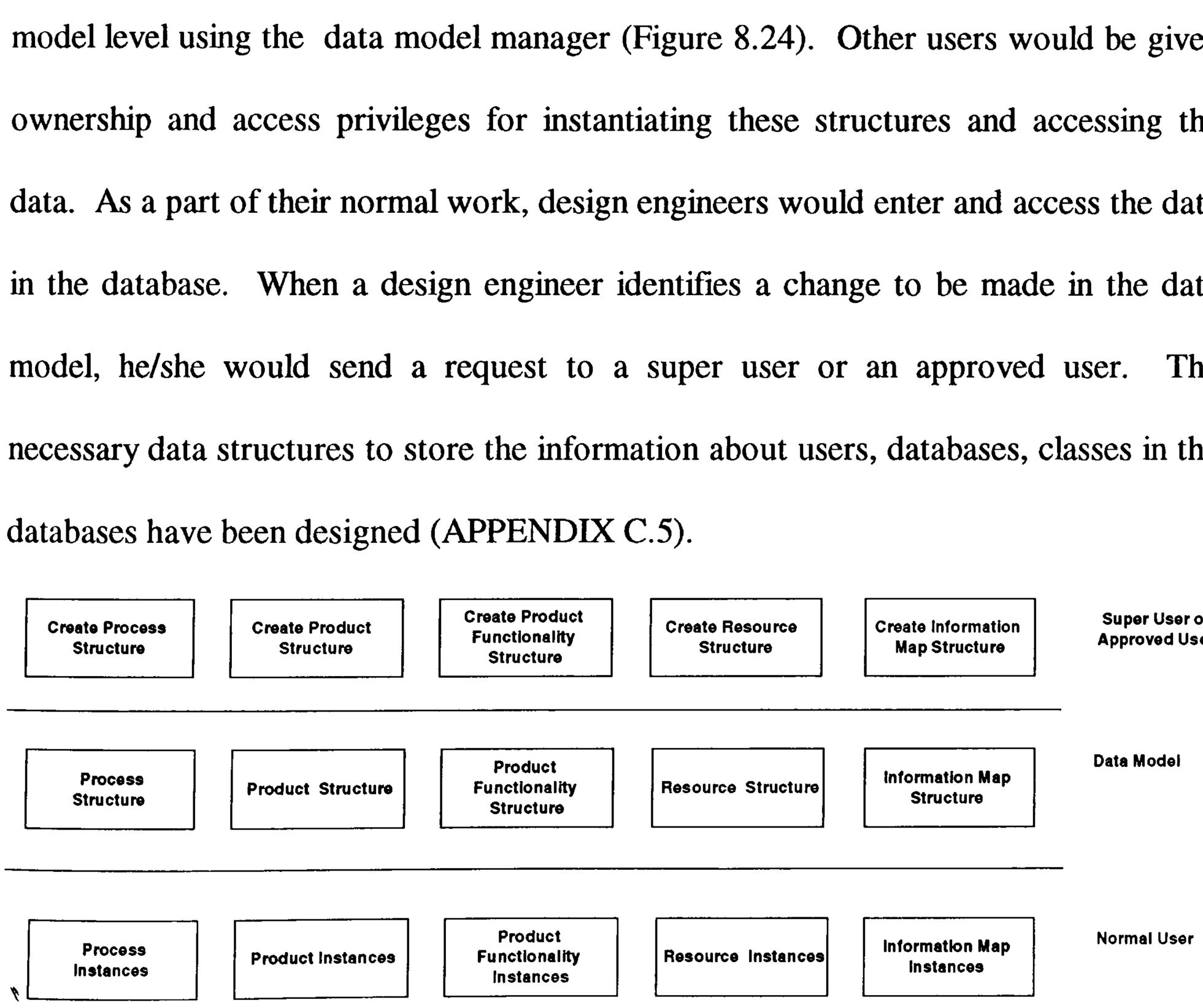


Figure 8.24 Users and security aspects of the information model

Communication acceleration at attribute level - The timing of access to information is particularly critical to a project's success because it can directly affect the cycle time to introduce a new product. The information map database provides information links between components to enable information flow. At the start of an activity, using the information maps attached to an activity, relevant attribute maps and the meta-model, one can easily access the information, and also trace the source from which the information is derived, and can also find out whether the information is ready or not. On the other side, when an attribute is filled in as an outcome from the activity, using that attribute as the source attribute, the three passes - pass 2, pass 3 and pass 4 of the algorithm shown in Figure 8.25 can be executed to support concurrent engineering.

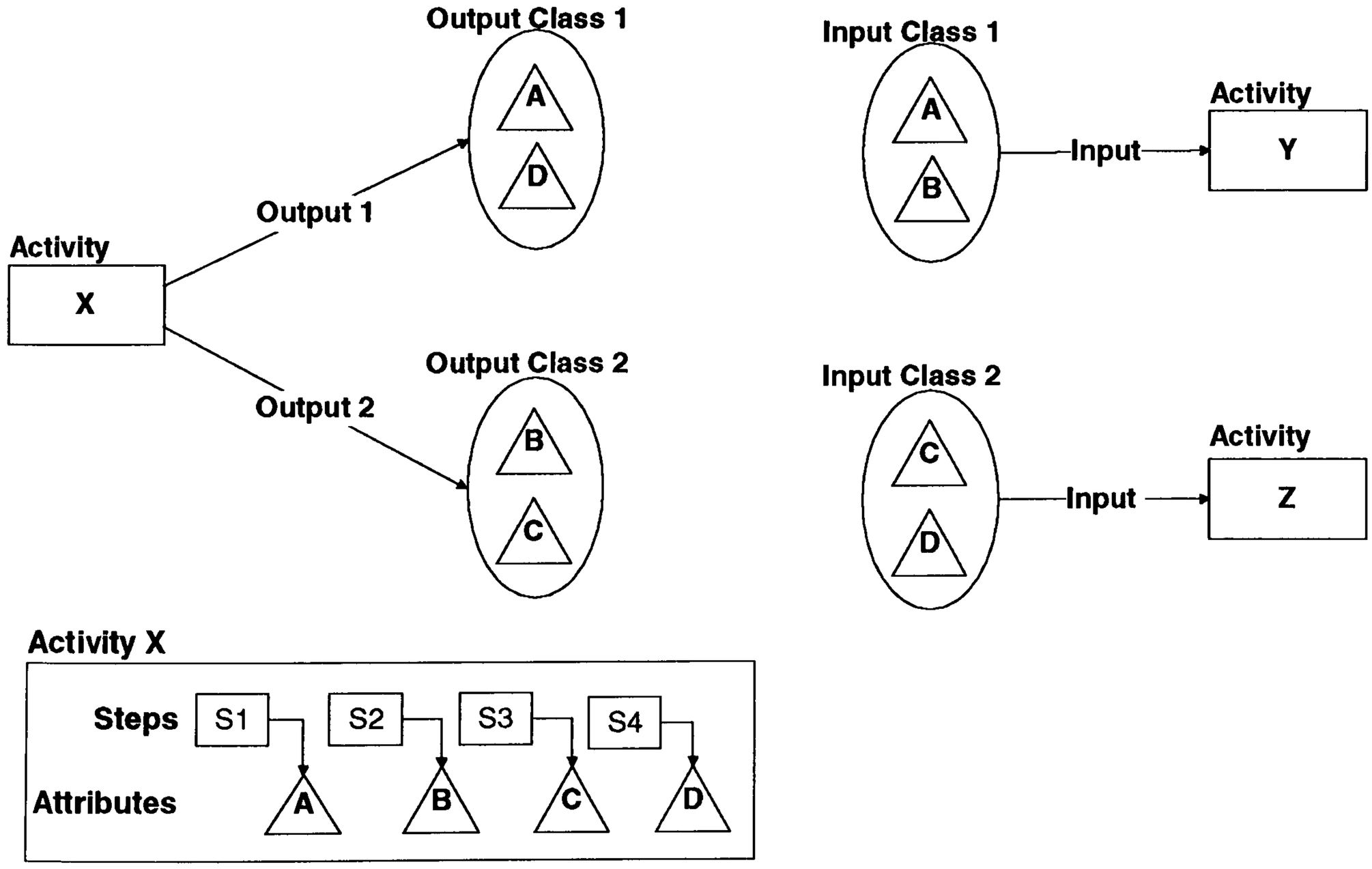


Figure 8.25 Acceleration at attribute level

This would allow communication acceleration, at attribute level, a level below the activities. At the completion of steps S1 and S2 of activity X (Figure 8.20), values of attributes A and B are available. Hence, activity Y can be triggered to start, even though activity X is not yet completed. Such communication acceleration at attribute level would require the storage of the internal logic (S1 → S2 → S3 → S4) of the activities.

Limitations - The current limitations of the information model are as follows:

Limitations imposed by local DBMS - Even though UniSQL/M allows the integration of various databases listed in APPENDIX G (under section G.2), there are certain limitations imposed by the registered local databases in building the global model of the product introduction process information using the meta-model. For example, SYBASE allows dynamic data modelling capabilities only when 'ddl in tran' option (i.e. data definition language statements within transactions) is set to true, and Microsoft SQL Server does not support remote schema modifications via the EVALUATE ON LDB statement. Such limitations imposed by the registered local database management systems remain in the implementation of the information model.

At present, the prototype has no mechanism to test for synonyms of the attribute names or class names; the structure of the dynamic classes are in the hands of the users of the meta-model manager and data model manager. There are chances for two attributes representing the same data but defined in the meta-database twice using two different attribute names. For example, attributes such as 'part number' and 'part id' could be representing the same data.

CHAPTER 9

9. CONCLUSIONS AND FURTHER RESEARCH

9.1 SUMMARY OF RESEARCH

Manufacturing industry is under constant pressure to increase the speed of product development and the product choices that are offered to customers, and product introduction is a vital process for most firm's growth and prosperity. A product is introduced by adding new functionalities or by improving the available functionalities of an existing product. Through the process of product introduction, ideas and needs are converted to the information from which technical systems and products can be made and sold. Two critical factors for its success are the management of the product introduction activities, and the quality and functionality of its output (i.e. the product) which itself depends on the quality of the product introduction process. The process is as effective as the decisions made within it, and as efficient as the speed with which the information required for each decision is made available. The process management includes project management, information flow management and information management. The foundation for the management of the PI process is an information infrastructure that supports the capture, change and transfer of evolving information.

This thesis reports an investigation into what information needs to be represented for the management of the product introduction process, and how it is to be represented. The research has resulted in the design of an integrated information model for product introduction information based on the idea of linking activities that generate the product information with their results using information mapping technique. A

prototype that demonstrates the primary features of the PI process has been developed to test the validity of the integrated information model. To demonstrate the use of the integrated information model, an activity-dependency analysis tool and algorithms for communication acceleration that enable effective information flow have been developed.

9.2 CONCLUSIONS

Process analysis - The product introduction process aims to generate the definition of a product that meets its required product functionalities taking into consideration the time and cost constraints on the PI process. The effectiveness and efficiency of the PI process are strongly influenced by the way the process is managed. The management of the PI process is highly dependent on the management of the activities of the PI process, information generated by the activities and the information flow among the activities. Mastering the information flow allows a better integration of the activities of the PI process, but data representation techniques for the information flow among the activities are missing. The association between activities that reuse the information and data, and the role the data plays with respect to the activity are very essential when integration is to be achieved, business processes are to be reengineered, or concurrent engineering is to be practised.

Information flow analysis - Information (product specification) is input to the product introduction activities, information (product definition) is generated from the PI activities, it is also used to control the process flow (via feedback) and may also be an input (changes in product specification). The link between the managerial and technical information is essential and critical as it is the output of the activities, i.e.

technical information that controls the decisions in managing the project. Improved co-ordination and flow of project and product information between tasks are necessary for rapid product introduction and minimising errors.

Information analysis - The product introduction information includes project management information (schedule, budget, resource and organisation) as well as product functionalities and product information. As the information such as product functionality can not be represented using the existing project management software tools, the tools cannot be used for managing the PI process effectively. Even though product functionality is a driving factor for the success of the product introduction project, very few models (KOF and CHROMOSOME model) address product functionality, and they represent product functionality as a form feature, not as a goal of the PI process. Modelling methodologies should tackle the problem of modelling product functionality in order to control the PI process effectively.

A great deal of knowledge is generated during the product introduction process. This evolving product knowledge need to be captured and represented in a structured form for easy dissemination and sharing among the PI resources i.e. teams. In general, product models are used to store such information. Formal information models for product information are beginning to appear, but the literature does not yet address the ways in which databases should provide structurally complex information to application processes that use the product information. Even though a process-chain driven approach to product modelling is identified as an important future research topic in database and related information technologies, there is no model that details how to link the process in the process chain that generates the product data with the

product model data.

Information representation analysis - It is clear from the literature that there is a lack of information models for the product introduction process. While designing the information model for the product introduction process , the features such as evolution, heterogeneity in structure, semantics and data structure call for special issues in its representation.

Evolution of information - The capture, change and transfer of evolving information become the main requirements to support the management of the product introduction process. Dynamic modification or extension is an essential feature of the information model for the PI process. A model with a meta knowledge has been identified for designing integrated information models that will aid in building information systems that can be dynamic, co-operative and distributed. Thus, a meta-model concept that supports evolution of the data model has been used in designing the information model for the PI process.

Heterogeneity in structure - The representation of the product introduction information calls for integrating heterogeneous structures such as hierarchical and relational. The decomposition-aggregation relationship that exists in the vertical dimension in the product introduction project is used to define the *flow of control* relationship between modules at different levels of abstraction and it is of a hierarchical type. The information dependency relationship (input, constraint and output) that exists in the horizontal dimension in the product introduction project is used to define *flow of information* relationship between modules at the same level of abstraction, or between

modules which are not related by a direct flow of control from one to the other, and it is of a network type. The information generated by the activities of the product introduction process is about the product, and object-oriented models are used to represent the product data as they involve inheritance.

Heterogeneity in semantics - The PI process is multidisciplinary and each discipline participating in the product introduction process has its own view of the data needed to define the product. The view of a particular discipline optimises the organisation and content of the product data from the perspective of the team members working in that discipline. The views include project management view, design view, materials view, product information view, etc. Representing the different semantics in a single framework is difficult, but essential for the management of the product introduction process. This heterogeneity in semantics leads to heterogeneity in data structure.

Heterogeneity in data structure - The product introduction information involves structural data and repetitive data intertwined, and requires the same data to be represented at structural level and data level within a single framework. In existing database systems there is a strict borderline between structural data (classes) and repetitive data (objects); and instantiation is used to link the class and the instance levels. An extension to the instantiation concept, to consider classes as objects, is required to represent the structural and repetitive data in a single framework

9.3 ACHIEVEMENTS

The research achievements include - ① devising the requirements for information management, ② information mapping, a representation technique to represent the

associations between activity and information, ③ design of the integrated information model and development of a prototype to test its validity and ④ development of the tools to demonstrate the use of the information model.

Requirements for information management - It has been found that the following models/systems need to be designed and developed to manage the information relevant to the product introduction process:- models that represent goals (product functionalities), activities, human and computational agents (PI resources), and product information; an information model that integrates the above models based on the definition of semantic relationships between the models; a model management system that allows for continual evolution of the integrated information model mentioned in the previous step; a data management system that will take care of the evolving data that is generated by instantiating the information model.

Information maps - The semantic relationships between product functionality, product introduction process or project, product introduction resources and product have been represented using information maps which assemble sets of related information that are traditionally available at the same time but not in a consolidated form, thus facilitating easy access and use. The structure of the information map would contain pointers to the related objects that are stored in different databases, and other properties of the relationship among the related objects. It would guarantee easy retrieval of information as a group and enable information flow among the activities of the PI project by pointing to the information necessary to start an activity. The dependencies among the activities can be traced easily by following the information maps and attribute maps.

Design of the integrated information model - A three dimensional information model of the product introduction information from three related but different viewpoints has been designed at abstract level, and detailed data structures that constitute the building blocks of the information model have been designed. As the lead time and project cost are information related to the activities of the PI project, and product cost and product functionality are information related to the product, i.e. results of the activities, the design of an integrated information model would be based on linking the activity plan with the results of the activity using information maps. The first dimension deals with the handling of the project management (PI project, PI resources) and technology knowledge (product functionalities, product), the form in which the technical information are required by the activities of the project, the form in which the output information is generated by the activities (information map), and the form in which they are structured (product model) for further usage. The 'representation of relationships' has played a major role in the design of the information model. A method for the development of the integrated model by integrating the product functionality model, the project model, the product introduction resource model and the product model using the information map model, the meta-model and object-relational concepts has been proposed. A prototype implementing the important features has been developed to test the validity of the integrated information model. The highlights of the proposed integrated information model include the - ① representation of evolving information using meta-modelling technique, ② representation of project management view and the technical view in a single architecture using information maps and ③ representation of heterogeneous data in a single framework using information mapping concepts, the meta-modelling technique and an extension to the instantiation concept.

Development of tools - In order to demonstrate the use of the integrated information model, the following two tools have been developed: ① activity-dependency analysis tool and ② communication accelerator. The activity dependency analysis tool identifies direct successor, indirect successor, direct predecessor and indirect predecessor of a given activity using the information maps attached to the activity and the attribute maps stored in the meta-model. To develop the communication accelerator, algorithms have been devised to answer the question of who should be alerted when the output data of an activity becomes available.

The integrated information model can be applied for the management of the introduction of a product with completely new functionalities, or improved functionalities based on a current product, in the case of any complex products such as automobiles and aeroengines.

9.4 LIMITATIONS

The current limitations of the information model are as follows:

1. Limitations imposed by local DBMS - An object-relational database system is suggested for implementing the overall model. There are certain limitations imposed by the registered local databases in building the global model of the product introduction process information using the meta-model. For example, SYBASE allows dynamic data modelling capabilities only when 'ddl in tran' option (i.e. data definition language statements within transactions) is set to true. Such limitations imposed by the registered local database management systems remain in the implementation of the information model.

2. At present, the prototype has no mechanism to test for synonyms of the attribute names or class names; the structure of the dynamic classes are in the hands of the users of the meta-model manager and data model manager. There is a chance for two attributes representing the same data to be defined in the meta-database twice using two different attribute names. For example, attributes such as 'part number' and 'part id' could be representing the same data.

9.5 RECOMMENDATIONS FOR FURTHER RESEARCH

This research work can be further extended with the following: exploring concurrency, project management system, real-time tool and provision for a basic structure database.

Exploring concurrency - the activity dependency analysis tool can identify the dependency among activities using the information maps attached to the activity and attribute maps. The dependency relationship between the activities defines the order of execution of activities. An analysis tool to explore the concurrency among the activities can be developed that helps to find ways of reducing the product introduction project time.

Project management system - the relationships between activities and information (product functionality, resource, product) can be represented as an activity-information relationship matrix; the elements of this matrix will be I/C/R/O denoting the relationship input, constraint, resource or output respectively. An analysis of this matrix can be used for scheduling and management of the activities. A project management system to interface with the integrated information model can be developed to manage the build up of the information repository.

Design of a product can be managed using parameters that are critical in their effect on product functionalities. A detailed *product functionality-parameter dependency* picture of the relationship between the product functionality in the upper level and the parameters of parts in the lower level can be obtained starting from the information maps attached to the product functionality and attribute maps. An equivalent picture of the relationship between the activity that use and/or generate the product functionality and the product functionality can also be obtained using the information maps attached to the activity and attribute maps. This information can be used in managing the overall product introduction project.

Real-time tool - The information model represents the technical information and the information on PI resources associated with an activity. Thus, it represents the information that a person requires. Representation of the minimum level of completeness of the information at which it would be useful to the PI resource, and of control mechanisms such as when the information is required in terms of both completeness and timing would make it a real-time tool.

Basic structure database - Similar or identical activities may be performed in the design of different subsystems of a product. A database of basic, standard structure definitions for various information on product functionality, project, resource, product can be created and maintained for re-use. This will enable the users who enter the information in the meta-model to pull the basic structures and alter according to their requirements.

REFERENCES

1. AKIYAMA K., 1991, *Function Analysis - Systematic Improvement of Quality and Performance*, (Cambridge : Productivity press Inc.).

2. ALMASI G., MONTAN V., QIU S., SHMIDT E., TRAPP G., 1992, An Information Sharing System based on an EXPRESS model, CERC Technical Report Series, (West Virginia : West Virginia University, Oct, CERC-TR-TM-92-019).

3. AMALNIK M.S., 1994, Integrated Product Development at Design and Manufacturing, Company, National and International level based on Computer Based System and Concurrent Engineering, *Proceedings of Factory 2000 - Advanced Factory Automation Conference*, (IEE, 3-5 Oct), pp. 249-256.

4. ANDERSON D.C., CRAWFORD R.H., 1989, Knowledge Management for preliminary Computer-aided mechanical design, *Proceedings of Organisation of Engineering Knowledge for Product Modelling in Computer Integrated Manufacturing*, edited by T.Sata, (Elsevier), pp. 15-38.

5. ANDREASEN M.M., 1990, Theory of Domains, *Proceedings of Function Workshop*, University of Cambridge.

6. ANDREASEN M.M., 1998, Conceptual Design Capture, *Proceedings of Design Reuse Engineering Design Conference '98*, (London : Professional Engineering Publishing), pp. 21-29. (including the slides from the presentation of the keynote paper).

7. ANDREASEN M.M., HANSEN C.T., MORTENSEN N.H., 1996, The structuring of products and product programmes, *Proceedings of the 2nd WDK workshop on product structuring*, (Netherlands : Delft University of technology, 3-4 Jun).

8. ANGUS J., MURDOCH T., 1993, Supporting Product Definition Process, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 1725-1732.

9. BADIRU A.B., 1988, *Project Management in Manufacturing and High technology operations*, (New York: John Wiley & Sons).

10. BADIRU A.B., 1996, *Project Management in Manufacturing and High technology operations*, Second Edition, (New York: John Wiley & Sons).

11. BAUERT F., 1993a, Creative & Systematic Team-Work Using Abstract Concepts and Concrete Entities, 1993, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp 355-358.

12. BAUERT F., ABEL C.A., EDWARDS K.L., CHONG W-T., EVRON I., 1993b, *The Management Assistant Tools to Support Managerial Activities of Engineering Design Projects*, Engineering Design Centre, Engineering Department, Cambridge University, CUED/C-EDC/TR 15.

13. BAXTER J.E., 1994a, A functional data model for assemblies used to verify product design specifications, *Proceedings of the Institution of Mechanical Engineers*, (Part B - Journal of Engineering Manufacture), **208**(B4), pp. 235-244.

14. BAXTER J.E., 1994b, *A Functional Data Model For Assemblies*, Ph.D. thesis, (The University of Leeds, May).

15. BEECKMAN D., 1989, CIM-OSA: Computer Integrated Manufacturing - Open System Architecture, *International Journal of Computer Integrated Manufacturing*, **2**(2), pp.94-105.

16. BEECKMAN D., 1990, CIM-OSA - An Illustrative Example of How to Apply the Modelling Framework, *CIMCON '90*, (Gathersburg, NIST Special Publication, May), pp. 197-215.

17. BJORKE O., 1989, Product Modelling for Design and Production, *Proceedings of Organisations of Engineering Knowledge for Product Modelling in Computer Integrated Manufacturing*, T. Sata (Ed.), Elsevier, pp. 141-177.

18. BLESSING L.T.M., 1993, A process-based approach to computer supported engineering design, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 1393-1400.

19. BLESSING L.T.M., 1994, *A process-based approach to computer supported engineering design*, Ph.D. thesis, (Netherlands : University of Twente), pp. 139, 140, 169.

20. BLOOR M.S., PENNINGTON A.DE, HARRIS S.B., HOLDSWORTH D., MCKAY A., SHAW N.K., 1991, Towards Integrated Design and Manufacture, in: *Design for Manufacture – Strategies, Principles and Techniques*, (Addison-Wesley Publishing Company), pp.258-269.

21. BOOCH G., 1986, Object-Oriented Development, *IEEE Transactions on Software Engineering*, **SE-12**(2), Feb., pp. 211-221.

22. BOWEN Inc., 1995, *UniSQL's Object-Relational Data Management Technology*, The Bowen Group Inc., Apr.

23. BROOKS B., 1995, Time Equals Money - But where does it all go?, *IEE Colloquium on Concurrent Engineering - Getting it Right First time*, (London, 8 June), pp. 4/1 - 4/23.

24. BS6046: PART 1, 1984, British Standard Use of Network Techniques in Project Management, Part 1. Guide to the use of management, planning, review and reporting procedures, (British Standards Institution).

25. BS6046: PART 2, 1992, British Standard Use of Network Techniques in Project Management, Part 2. Guide to the use of graphical and estimating techniques, (British Standards Institution).

26. BS6046: PART 3, 1992, British Standard Use of Network Techniques in Project Management, Part3. Guide to the use of Computers, (British Standards Institution).

27. BS7000: PART 10, 1995, British Standard Design Management System, Glossary of terms used in design management, (British Standards Institution).

28. BS7000: PART 2, 1997, Design Management System: Guide to managing design of manufactured products, (British Standards Institution).

29. BS7000: PART 4, 1996, Design Management System: Guide to managing design in construction, (British Standards Institution).

30. BUDILL E.J., 1989, How Process Logistics Planning Can Enhance the Effectiveness of Simultaneous Engineering, *Proceedings of the Simultaneous Conference*, pp. 73-81.

31. CARVER G.P., BLOOM H.M., 1991, Concurrent Engineering through Product Data Standards, (U.S.: Department of Commerce, May).

32. CERC, 1992, Minutes - CERC's First Workshop on Product Development Process Modelling and Characterisation, Research Note, *CERC Technical Report Series*, (West Virginia : West Virginia University, CERC-TR-RN-92-006), pp. 14.

33. CHAKRABARTI A., 1993, Towards A Theory for Functional Reasoning in Design, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 1-8.

34. CHAKRABARTI A., 1993, Towards A Theory for Functional Reasoning in Design, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 1-8.

35. CHAPMAN C.B., 1996, Design Automation through Knowledge Based Systems, *Warwick Manufacturing Group - Course Notes*, University of Warwick, U.K.

36. CHEN P.P., 1976, Entity-Relationship Model - Towards a Unified View of Data, *ACM Transactions on Database Systems*, **1**(1), pp. 9-36.

37. CLEETUS K.J., REDDY R., 1992, Concurrent Engineering Transactions, *Proceedings of CE & CALS 1992 Washington conference and Exposition*, (Washington DC, 1-4 Jun).

38. CLEETUS K.J., UEJIO W.H. (Editors), 1989, *Blackboard for Design Evolution, Red book of Functional Specifications for the DICE Architecture*, Working draft, pp. 75-93.

39. CLEETUS K.J., 1992, Definition of Concurrent Engineering, *CERC Technical Report Series,* (West Virginia : West Virginia University).

40. CLEETUS K.J., 1995, Modelling Evolving Product Data for Concurrent Engineering, *Engineering with Computers*, **11**, pp. 167-172.

41. CLEETUS K.J., CASCAVAL G.C., MATSUZAKI K., 1996, PACT – A Software Package to Manage Projects and Coordinate People, (IEEE: *http://cybermarche.cymar.com/pact/pactpaer.html*).

42. CODD E.F., 1980, Data models in Database Management, *Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling,* (Brodie M.L., Zilles S.N., eds.), (Pingree Part, Colorado, June), pp. 112-114.

43. COOPER R.G., 1993, *Winning at New Products - Accelerating the Process from Idea to Launch*, Second Edition, (Addison-Wesley Publishing Company).

44. CRABTREE R.A., BAID N.K., FOX M.S., 1993, An Analysis of Coordination Problems in Design Engineering, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 285-292.

45. CUTTS G., 1987, *Structured System Analysis and Design Methodology,* (London : Paradigm Publishing Ltd), pp. 19.

46. DATE C.J., 1991, *An Introduction to Database Systems*, Fifth Edition, (Reading : Addison-Wesley), pp 269-270.

47. DATE C.J., DARWEN H., 1998, *Foundation for Object/Relational Databases: The Third Manifesto*, (Addison-Wesley).

48. DE MAIO A., 1994, A Multi-Project Management Framework for New Product Development, *European Journal of Operational Research*, 78, pp. 178-191.

49. DEMARCO T., 1979, *Structured Analysis and System Specification*, (New Jersey : Yourdon Press), pp. 47-69.

50. DOUMEINGTS G., CHEN D., VALLESPIR B., FENIE P., 1993, GIM (GRAI Integrated Methodology) and its evolutions A methodology to design and specify Advanced Manufacturing Systems, *Information Infrastructure Systems for Manufacturing*, Edited by Yoshikawa H., Goossenaerts B.V., IFIP, (North Holland : Elsevier Science), pp. 101-120.

51. DOUMEINGTS G., WANE N.O., GIRARD P., MARCOTTE F., 1994, Architecture and Methodology for concurrent Engineering, IFIP, (North Holland : Elsevier Science), pp. 165-177.

52. DOYLE P., 1994, *Marketing Management and Strategy*, (London: Prentice Hall), pp. 189-216.

53. DRONGELEN I.C.K., DE WEERD-NEDERHOF P.C., FISSCHER O.A.M., 1996, Describing the Issues of Knowledge Management in R&D: Towards a Communication and Analysis Tool, *R & D Management*, 26(3), pp. 213-229.

54. DUNCAN W.R., 1996, *A Guide to the Project Management Body of Knowledge*, (Upper Darby, USA : PMI publications), pp. 1.2, 1.3

55. DUTTON J.E., 1993, Commonsense Approach to Process Modelling, *IEEE Software*, **10**(July), pp.56-64.

56. EASTMAN C.M., FERESHETIAN N., 1994, Information Models for use in Product Design: a Comparison, *Computer-Aided Design*, **26**(7), July, pp. 551-572.

57. ELMARAGHY H.A., ZHANG K.F., CHU H., 1993, A Function-oriented Modeler Prototype, *Design for Manufacturability*, Ed. Guichelaar P.J., (New York : The American Society of Mechanical Engineers, DE), **52**, pp. 57-62.

58. EPPINGER S.D., WHITNEY D.E., ROBERT P.S., DAVID A.G., 1989, Organising the tasks in Complex Design Projects, *Lecture notes in Computer Science, Computer aided Cooperative Product development*, MIT-JSME workshop, Cambridge University, Springer Verlag, Nov, pp 229-252.

59. EPPINGER S.D., WHITNEY D.E., SMITH R.P., GEBALA D.A., 1994, A Model-Based Method for Organising Tasks in Product Development, *Research in Engineering Design*, pp. 1-13.

60. ERENS F., MCKAY A., BLOOR S., 1993, Shortcomings of Today's Design Frameworks, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 1279-1286.

61. ERENS F., MCKAY A., BLOOR S., 1994, Modelling Product families in an engineering framework, *MOSES Report Series –7*, (Leeds, UK : University of Leeds).

62. EVANS J.R., 1993, *Applied Production and Operations management*, Fourth edition, (New York: West Publishing Company), pp. 621-655.

63. EVERSHEIM W., ROZENFELD H., BOCHTLER W., GRAESSLER R., 1995, A Methodology for an Integrated Design and Process Planning Based on a Concurrent Engineering Reference Model, *Annals of the CIRP*, **44**(1), pp. 403-406.

64. FOWLER J., 1996, Reviewer's Guide to ISO 10303-221, PDT Solutions, *http://www.pdtsolutions.co.uk/standard/papers/ap221/reviewer.html.*

65. FOX J., 1993, *Quality through Design - The key to successful product delivery*, (London : Mc-Graw-Hill).

66. FOX J., 1994, Designing for Function, *Engineering Designer*, Jan/Feb, pp.12-13.

67. FREEMAN P., NEWELL A., 1971, A Model for Functional Reasoning in Design, *Proceedings of the Second International Joint Conference on Artificial Intelligence*, (London : Portland Place, 1-3 Sep).

68. GUI J.K., MANTYLA M., 1994, Functional Understanding of Assembly Modelling, *Computer-Aided Design*, **26**(6), June 1994, pp. 435-451.

69. HALES C., 1987, *Analysis of the Engineering Design Process in an Industrial Context*, Ph.D. thesis, (Cambridge: University of Cambridge).

70. HAMMER M., McLEOD D., 1981, Database Description with SDM: A Semantic Database Model, *ACM Transactions on Database Systems*, **6**(3), Sep, pp. 351-386.

71. HAUPTMAN O., 1986, Influence of task types on the relationship between communication and performance: the case of software development, *R & D Management*, 16(2), 127-139.

72. HAUPTMAN O., HIRJI K.K., 1996, The Influence of Process Concurrency on Project Outcomes in Product Development: An Empirical Study of Cross-Functional Teams, *IEEE Transactions on Engineering Management*, **43**(2), pp. 153-164.

73. HAYES R.H., WHEELWRIGHT S.C., CLARK K.B., 1988, *Dynamic Manufacturing - Creating the Learning Organization*, (New York: Free Press).

74. HEIN L., 1994, Design Methodology in Practice, *Journal of Engineering Design*, **5**(2), pp. 165-181.

75. HOGARTH P., TABESHFAR K., 1993, The Influence of Design Visualisation in Reducing Investment in New Product Development, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 127-133.

76. HOLLINS B., HURST S., HOLLINS G., 1993, Design Management Processes that can be used by Practitioners, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 656-663.

77. HORDVIK U., OEHLMANN R., 1992, Support of Collaborative Engineering through a Shared High Level Product Model - Basic Requirements and Concepts, in: *Integration in Production Management Systems*, Pels H.J., Wortmann J.C., (Eds.), (North-Holland : Elsevier Science Publishers), pp. 37-51.

78. HUNDAL M.S., 1993, Engineering and Management for Rapid Product Development, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 588-595.

79. JAGANNATHAN V., CLEETUS J., MATSUMOTO A.S., KANNAN R., LEWIS J.W., 1991, Computer Support for Concurrent Engineering, *CERC Technical Report Series*, (West Virginia : West Virginia University, Jul).

80. JAGANNATHAN V., KARINTHI R., SOBOLEWSKI M., ALMASI G., 1993, Model-based Information Access, *CERC Technical Report Series*, (West Virginia : West Virginia University, Apr, CERC-TM-93-007).

81. KARINTHI R., JAGANNATHAN V., MONTAN V., PETRO J., RAMAN R., TRAPP G., 1992, Promoting Concurrent Engineering Through Information Sharing, *CERC Technical Report Series*, (West Virginia : West Virginia University, Nov, CERC-TR-TM-92-10).

82. KATZY B.R., EVERSHEIM W., DOBBERSTEIN M., FEUERBORN K., FRIEDRICH J., KRAH O., MULLER G., STEPPRATH F.J., 1993, CIMOSA Pilot Implementation for Technology Transfer, *Information Infrastructure for Systems Manufacturing*, Edited by Yoshikawa H., Goossenaerts B.V., IFIP, (North Holland : Elsevier Science), pp. 225-236.

83. KERZNER H., 1995, *Project Management - A system approach to planning, Scheduling and Controlling*, (New York: Van Nostrand).

84. KIM W., 1995, Object-Relational Database Technology, *A UniSQL Whitepaper*, (Austin : UniSQL Inc., http://www.unisql.com/whtpaper.htm).

85. KOKOL P., 1993, Meta-modelling: How, Why and What?, *Software Engineering Notes*, 18(2), pp. 25-26.

86. KRAUSE F.L., 1989, Knowledge Integrated Product Modelling for Design and Manufacture, *Proceedings of Organisation of Engineering Knowledge for Product Modelling in Computer Integrated Manufacturing*, T. Sata (Ed.), (Elsevier), pp. 179-223.

87. KRAUSE F.L., OCHS B., 1992, *Potentials of Advanced Concurrent Engineering Methods, Manufacturing in the Era of Concurrent Engineering*, Edited by Halevi G., Weill R.D., IFIP, (North-Holland: Elsevier Science), pp. 15-27.

88. KRAUSE F.L., KIMURA F., KJELLBERG T. et al., 1993, Product Modelling, *Annals of the CIRP*, **42**(2), pp. 695-706.

89. KUSIAK A., 1993, *Concurrent Engineering - Automation, tools and Techniques*, (John Wiley & Sons).

90. LEWIS W.P., HAIYING W., 1998, Factors affecting the successful development of new products, *Proceedings of the Design Reuse - Engineering Design Conference '98*, Edited by Sivaloganathan, S., Shahin T.M.M., (Suffolk: Professional Engineering Publishing Limited, UK), pp. 409-416.

91. LINDBERG L., 1993, Notes On Concurrent Engineering, *Annals of the CIRP*, 42(1), pp. 159-162.

92. LINDEMAN D, WIJAYA L, 1992, Managing Design Structure Data in a Concurrent Engineering Design Environment, *Engineering Data Management: Key to Integrated Product Development*, (ASME), pp. 97-104.

93. LOCK D., 1984, *Project Management*, (Worcester: Billings and sons Limited), pp. 16-17.

94. LOHSE G., 1993, Methodical Product Development - Experience of a Practical Study, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 200-207.

95. LONDONO F., CLEETUS K.J., NICHOLS D.M., IYER S., KARANDIKAR H.M., REDDY S.M., POTNIS S.M., MASSEY B., REDDY A., GANTI V., 1992, Coordinating a Virtual Team, *CERC Technical Report Series*, CERC, West Virginia University.

96. MANGANELLI R.L., KLEIN M.M., 1995, *The Reengineering Handbook, A step-by-step guide to Business transformation*, (USA: Amacom).

97. MARLOW G., SCHULZ S., 1994, Concurrent Engineering Enablers - Engineering Libraries, *Proceedings of the Annual Reliability and Maintainability Symposium*, pp. 315-319.

98. MCGRATH M.E., ANTHONY M.T., SHAPIRO A.R., 1992, *Product Development: Success Through Product and Cycle-time Excellence*, (London: Butterworth-Heinemann).

99. MENON U., SYAN C.S., 1992, Frameworks for Concurrent Engineering and Rapid Prototyping, *Proceedings of the Eighth International conference on CAD/CAM, Robotics and Factories of the Future*, (France, Aug 17-19), 1, pp. 249-261.

100. MOLINA A.G., 1995, *A Manufacturing Model to Support Data-Driven Applications for Design and Manufacturing*, Ph.D. thesis, (Loughborough University, March).

101. MORTENSEN N.H., ANDREASEN M.M., 1996, Designing in an interplay with a product model - explained by design units, (Budapest, Hungary, *TMCE '96*).

102. MUNRO-FAURE M., 1994, *Implementing Total Quality Management*, (London: Pitman Publishing), ISBN 0 273 038486, pp. 20.

103. MURDOCH T., 1993, *Configuration Evaluation and Optimisation of Technical Systems*, Ph.D. thesis, (Cambridge University, Sep), pp. 18.

104. OAKLAND J.S., 1995, *Total Quality Management: Text with Cases*, Student Ed., (Oxford: Butterworth-Heinemann Ltd), pp. 12-17.

105. PAHL G., BEITZ W., 1984, *Engineering Design*, Edited by Wallace K., (London: Design Council).

106. PAHL G., BEITZ W., 1996, *Engineering Design - A Systematic Approach*, Edited by Wallace K., Blessing L.T.M., Bauert F., (London: Springer_Verlag).

107. PANSE R. (presenter on behalf of ESPRIT AMICE Consortium), 1990, CIM-OSA - A Vendor Independent CIM Architecture, *CIMCON '90*, (Gathersburg : NIST Special Publication, May), pp. 177-196.

108. PARKER A., 1997, Engineering is not enough, *Manufacturing Engineer*, (December), pp. 267-271.

109. PECKHAM J., MARYANSKI F., 1988, Semantic Data Models, *ACM Computing Surveys*, **20**(3), Sep, pp. 153-189.

110. PEIFER T., EVERSHEIM W., KONG W., WECK M. (Ed.), 1994, *Manufacturing Excellence, the competitive edge*, (London : Chapman & Hall).

111. PIETRAS C.M., COURY B.G., 1994, The development of cognitive models of planning for use in the design of project management systems, *International Journal of Human-Computer Studies*, 40, pp. 5-30.

112. PRESSMAN R.S., 1992, *Software Engineering A Practitioner's Approach*, (Singapore: McGraw Hill).

113. RANKY P.G., 1994, *Concurrent/Simultaneous Engineering (Methods, Tools and Case studies)*, (Guildford: CIMware).

114. REED R.G., 1993, An Application of Extended Function Logic to the Design of a Wearable Computer, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 251-255.

115. ROSENAU M.D., MORAN J.J., 1993, Managing the Development of New Products - Achieving Speed and Quality Simultaneously Through Multifunctional Teamwork, (New York: Van Nostrand Reinhold).

116. ROSENTHAL S.R., 1992, *Effective Product Design and Development - How to Cut Lead time and Increase Customer Satisfaction*, (Illinois : Business One Irwin).

117. ROY R., ALLCHURCH M.J., 1997, Development of a Manufacturing Engineering System for the Motor Industry, *Proceedings of the Portland International Conference on Management of Engineering and Technology*, (PICMET, July 27-31), pp. 668-671.

118. RUFFLES P.C., 1995, *Project DERWENT - A new approach to product definition and manufacture*, Rolls-Royce Aerospace group, Research Lecture Hand-out, Department of Engineering, University of Warwick.

119. RUMBAUGH J., BLAHA M., PRELMERLANI W., EDDY F., LORENSEN W., 1991, *Object-Oriented modelling and design*, (London: Prentice Hall).

120. RZEVSKI G., 1993, The Integrated Product Development Process: Issues and Methods, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 493-498.

121. SATA T., 1993, Concurrent Engineering, *Journal of the Institute of Electrical Engineers of Japan*, **113**(3), (Mar), pp 178-182.

122. SCASSO R.H., LARENAS G.S., 1991, Project-breakdown Structure: the Tool for Representing the Project System in Project Management, *International journal of project management*, 9(3), Aug, pp. 157-161.

123. SCHEER A.W., 1993, *Architecture of Integrated Information Systems (ARIS), Information Infrastructure for Manufacturing*, Edited by Yoshikawa H., Goossenaerts B.V., IFIP, (North-Holland: Elsevier Science), pp. 85-99.

124. SCHEER A.W., 1994, *Business Process Engineering - Reference Models for Industrial Enterprises*, (Berlin : Springer-Verlag) pp. 517-601.

125. SCHENCK D., WILSON P., 1994, *Information Modelling the EXPRESS way*, (New York : Oxford University Press).

126. SHAH J.J., ROGERS M.T., 1988, Functional Requirements and Conceptual Design of the Feature-Based Modelling System, *Computer-Aided Engineering Journal*, **5**(1), pp. 9-15.

127. SHAW N.K., BLOOR M.S., PENNINGTON A.DE, 1989, Product Data Models, *Research in Engineering Design*, **1**, pp. 43-50.

128. SHENHAR A.J., LAUFER A., 1995, Integrating Product and Project Management - A New Synergistic Approach, *Engineering Management Journal*, 7(3), Sep, pp. 11-15.

129. SHORTER D.N., 1990, Progress Towards Standards for CIM Architectural Frameworks, *CIMCON '90*, (Gathersburg, NIST Special Publication, May), pp. 216-231.

130. SMITH K., 1991, Representing and Managing Constraint Data in A Computer-Based Cooperative Product Development Scenario, CERC Technical Report Series, CERC, West Virginia University, May.

131. STARK J., 1992, *Engineering Information Management Systems - Beyond CAD/CAM to Concurrent Engineering Support*, (Van Nostrand).

132. STOCKBURGER H., 1993, Simultaneous Engineering in Space Development, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 650-655.

133. THIRUPATHI D., ROY R., 1997a, Concurrent Engineering in Product Introduction: Some Requirements for Information Management, *Proceedings of the First International Conference: Managing Enterprises - Stakeholders, Engineering, Logistics and Achievements* (ME-SELA'97, Loughborough University, UK, 22-24 July), pp. 561-566.

134. THIRUPATHI D., ROY R., 1997b, Towards an Information Model for Concurrent Engineering in Product Introduction, *Proceedings of the First Post-Graduate Symposium on Knowledge Exchange - Manufacturing, Logistics and Management* (KE-MLM'97, Loughborough University, 24 July), pp. 13-21.

135. THIRUPATHI D., ROY R., 1998, Development of an Information Model for Concurrent Engineering in Product Introduction, *Proceedings of the Engineering Design Conference* (EDC '98, Brunel University, 28 June), pp. 603-611.

136. ULLMAN D.G., 1993, A New View on Function Modeling, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 21-28.

137. UniSQL Inc., 1995, *UniSQL Object-Relational Database Management System and Technical Architecture Overview*, (Texas: UniSQL Inc.)

138. UniSQL, 1995, *UniSQL's Object-Relational Data Management Technology*, (Texas: UniSQL Inc., April).

139. UniSQL/M, 1996, *UniSQL/M User's Manual*, (Texas: UniSQL Inc.).

140. UniSQL/X, 1996, *UniSQL/X User's Manual*, Volume 1, (Texas: UniSQL Inc.).

141. VAN VEEN E.A., 1992, *Modelling Product Structures by Generic Bills-of-Materials*, (London : Elsevier), pp. 37-38.

142. VERNADAT F., 1993, CIMOSA: Enterprise Modelling and Enterprise Integration using a Process-Based Approach, *Information Infrastructure for Manufacturing*, Edited by Yoshikawa H., Goossenaerts B.V., IFIP, (North Holland : Elsevier Science), pp. 65-84.

143. VONDEREMBSE M.A., WHITE G.P., 1991, *Operations Management*, Second Edition, (New York: West Publishing Company), pp. 114-157.

144. WALLACE G., 1995, Computer-Aided Concurrent Engineering: the SCOPES solution, *World Class Design to Manufacture*, **2**(4), pp. 35-43.

145. WALLACE K., BLIGH T., 1996, *An Introduction to the Design Process*, Department of Engineering, University of Cambridge, Feb.

146. WU B., 1994, *Manufacturing Systems Design and Analysis, Context and Techniques* (London: Chapman and Hall), pp. 27-65.

147. ZACHMAN J.A., 1987, A Framework for Information System Architecture, *IBM Systems Journal*, **26**(3), pp. 276-292.

148. ZHANG W., WERFF K.VAN DER, 1993, Guidelines for Product Data Model Formulation Using Database Technology, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp 1618-1626.

149. ZHU G., GAO J., HU W., ZHOU J. YU J., 1993, A product definition model for concurrent design, *Proceedings of the International Conference on Engineering Design* (ICED '93, The Hague, 17-19 August), pp. 1602-1609.

150. ZIRGER B.J., MAIDIQUE M.A., 1990, A Model of New Product Development: An Empirical Test, *Management Science*, 36(7), pp. 867-883.

APPENDIX A

GLOSSARY OF TERMS

Aggregation — A special form of association, between a whole and its parts, in which the whole is composed of the parts.

Assembly — A part which is made up of two or more parts.

Association — A relationship among instances of two or more classes describing a group of links with common structure and common semantics.

Attribute — ① A representation of a trait, quality or property that is a characteristic of an entity or ② a named property of a class describing a data value by each object of the class

Cardinality — The specification of the number of instances of one construct that can be associated with one instance of a related construct.

Class — A description of group of objects with similar properties, common behaviour, common relationships, and common semantics.

Constraint — A particular kind of property that specifies a restriction on other properties of an entity, or on relationships.

Control Parameters — These are the parameters that the designers uses to effect the function required of the system. They are the parameters that are within the control of the designer and that affect the function. They are also the parameters that are varied during the experimental work (Fox 1994).

Critical Parameters — These are the control parameters that are critical (or paramount) in their effect on the function. If a critical parameter goes out of the range prescribed to it in the design, then the function is likely to experience a failure (Fox 1994).

Data type — A representation of a value domain.

Design (noun) — ① Set of instructions (specifications, drawings, schedules etc.) necessary to construct an artefact ② artefact itself (BS7000:Part 10 1995).

Design (verb)	Generation of information by which a required product can become a reality (BS7000:Part 10 1995).
Development	A process by which a product is brought to a standard at which it is ready to be manufactured or provided respectively (BS7000:Part 10 1995).
Effective process	An effective process is defined as one that results in a product satisfying the actual need (Blessing 1993); in other words, project effectiveness is about meeting product quality or functionality (Hauptman and Hirji 1996).
Efficient process	An efficient process is defined as one that is both effective and in which the applied resources do not exceed the planned resources (Blessing 1993) i.e. meeting project budget and schedule (Hauptman and Hirji 1996).
Entity	① Item of interest in the real world ② a modelling construct that is a representation of some item of interest in the real world.
Failure modes	A failure mode occurs when the response from a function exceeds (either positively or negatively) the limit expected from the design (Fox 1994).
Form	The form of the product is roughly defined by the spatial constraints that provide the envelope in which the product operates (Ullman 1997).
Form development	Is the evolution of components, how they are configured relative to each other and how they are connected to each other (Ullman 1997).
Function	In mechanical engineering, this term is used to describe *what* a device does (Ullman 1997).
Generalisation	Relationship between a class and one or more refined or specialised versions of it.
Heterogeneous databases	Databases that differ in things such as the underlying data model structure
Information models	Information models provide a structured description of the information entities which exists within an enterprise, and the relationships between them.
Instance	An object described by a class.

Latitude	This is the full range of response as described by the failure modes. In other words, it is the satisfactory working range of the function (Fox 1994).
Meta-model	Information model for the information that can be expressed during modeling
Noise factors	These are factors (parameters) directly affecting the function which are outside the control of the designer (Fox 1994).
Object	A concept, abstraction, or thing with crisp boundaries and meanings for the problem at hand; an instance of a class.
Population	An assignment of instances to a class is referred to as a population of that class.
Process	A process is defined as a sequence of interrelated activities that has a set of goals, constraints, inputs and outputs.
Product	Results of activities or processes (BS7000:Part 10 1995).
Product functionality	The purpose or intended use of a product (Freeman and Newell 1971). It is also known as the product function.
Product model	A representation of the data that describes a particular product throughout its life-cycle.
Project	A project is a temporary endeavour undertaken to create a unique product. Temporary means that every project has a definite beginning and definite end. Unique means that the product is different in some distinguishing way from all similar products (Duncan 1996).
Project management	Project management is the application of knowledge, skills, tools and techniques to project activities in order to meet or exceed the needs and expectations from a project. The needs are identified requirements and expectations are unidentified requirements (Duncan 1996).
Property	Particular aspect of an entity. Properties may represent values, constraints, behaviour, etc.
Relationship	An association between two constructs in a model.

Responses	These are the outputs from the function which are measurable. They may be noise factors for the next function within the system (Fox 1994).
Schema	A formalised arrangement of data; data structure; a framework of data.
Simple type	An elementary representation that cannot be further subdivided. Typical simple types are numbers, strings of characters and boolean values (i.e., true or false)
Specialisation	The creation of subclasses from a superclass by refining the superclass.
Viewpoint (or Views)	Shows different ways of looking at a set of data.

APPENDIX B

SCHEMA DEFINITION

B.1 INTRODUCTION

This appendix provides the definitions of the schemata involved in the product introduction information model for the management of the product introduction process. The overall schema is named as *IIM_Schema* where IIM stands for Integrated Information Model, and it includes the following schemata : ① schema of the product introduction process information (Process_Schema), ② schema of the product functionality information (ProductFunctionality_Schema), ③ schema of the product introduction resource information (Resource_Schema), ④ schema of the product information (Product_Schema) and ⑤ schema of the information map (InformationMap_Schema). A meta-model would be necessary to provide the evolution of the data structures. The schema of the meta model is also given in this appendix.

```
SCHEMA IIM_Schema;
        SCHEMA Process_Schema;          (* also called Project_Schema *)
        SCHEMA ProductFunctionality_Schema;
        SCHEMA Resource_Schema;
        SCHEMA Product_Schema;
        SCHEMA InformationMap_Schema;
        SCHEMA MetaModel_Schema;
END_SCHEMA;
```

B.2 PRODUCT INTRODUCTION PROCESS SCHEMA

```
SCHEMA Process_Schema              (* Project_Schema)

        USE FROM InformationMap_Schema (InformationMap);
        USE FROM Resource_Schema (Resource, Skill);

ENTITY Process                     (* also called Project *)
        name                       : string;
        description                : string;
        why                        : set_of  Process_Functionality;
        estimated_duration         : numeric;
        actual_duration            : numeric;
        estimated_cost             : numeric;
        actual_cost                : numeric;
        scheduled_start_date       : date;
        scheduled_due_date         : date;
        actual_start_date          : date;
        actual_due_date            : date;
        predecessor                : set_of  Process_ Process;
        successor                  : set_of  Process_ Process;
        has_parts                  : set_of  Process_ Process;
        is_part_of                 : set_of  Process_ Process;
        top_forward                : set_of  Process_ Process;
        top_backward               : set_of  Process_ Process;
        input                      : set_of  Process_InformationMap;
        control                    : set_of  Process_InformationMap;
        estimated_resource         : set_of  Process_Skill;
        actual_resource            : set_of  Process_Resource;
        target_output              : set_of  Process_InformationMap;
        achieved_output            : set_of  Process_InformationMap;
END_ENTITY;


ENTITY Activity
        SUBTYPE OF Process;
        WHERE
        is_activity                : SELF.has_parts = NULL set;
END_ENTITY;


ENTITY Process_Process
        process1                   : Process;
        process2                   : Process;
END_ENTITY;
```

```
ENTITY Process_InformationMap
        process                 : Process;
        InfMap                  : InformationMap;
END_ENTITY;

ENTITY Activity_InformationMap
        SUBTYPE OF Process_InformationMap;
        WHERE
         is_activity_infmap     : SELF.process.has_parts = Null set;
 END_ENTITY;

ENTITY Process_Skill
        process                 : Process;
        skill                   : Skill;
        level                   : string;
END_ENTITY;

ENTITY Process_Resource
        process                 : Process;
        resource                : Resource;
        role                    : string;
END_ENTITY;

ENTITY Process_Functionality
        process                 : Process;
        functionality           : Functionality;
        END_ENTITY;

END_SCHEMA;
```

B.3 PRODUCT FUNCTIONALITY SCHEMA

```
SCHEMA ProductFunctionality_Schema
                USE FROM InformationMap_Schema (InformationMap);
                USE FROM Resource_Schema (Resource);

        ENTITY Functionality
                name                    : string;
                description             : string;
                has_parts               : set_of Functionality_Functionality;
                is_part_of              : set_of Functionality_Functionality;
                how                     : set_of Functionality_Functionality;
                why                     : set_of Functionality_Functionality;
                top_forward             : set_of Functionality_Functionality;
                top_backward            : set_of Functionality_Functionality;
                input                   : set_of Functionality_InformationMap;
                noise_factor            : set_of Functionality_InformationMap;
```

```
        target_output          : set_of Functionality_InformationMap;
        achieved_output        : set_of Functionality_InformationMap;
END_ENTITY;

ENTITY Functionality_Functionality
        functionality1         : Functionality;
        functionality2         : Functionality;
END_ENTITY;

ENTITY FunctionalityUnit
        SUBTYPE OF Functionality;
        how                    : set_of Functionality_InformationMap;
        WHERE
        is_functionalityunit   : SELF.has_parts = NULL set;
END_ENTITY;

ENTITY Functionality_InformationMap
        functionality          : Functionality;
        informationmap         : InformationMap;
END_ENTITY;

ENTITY Parameter
        name                   : string;
        description            : string;
        unit_of_measure        : string;
END_ENTITY;

ENTITY CriticalParameter
        SUBTYPE OF Parameter;
        nominal value                  : numeric;
        range of tolerance             : Range;
        exp_date_of_definition         : date;        (* planned date *)
        latitude_desired               : Range;
        latitude_achieved              : Range;
END_ENTITY;


ENTITY Functionality_Parameter
        functionality  : Functionality;
        parameter      : Parameter;
END_ENTITY;


ENTITY Functionality_Response
        SUBTYPE OF Functionality_Parameter;
        latitude       : Range;
        failure_mode   : set_of FailureMode;
END_ENTITY;
```

```
ENTITY Range
        lower_limit             : numeric;
        upper_limit             : numeric;
END_ENTITY;

ENTITY FailureMode
        name                    : string;
        description             : string;
        limits                  : Range;
END_ENTITY;

        ENTITY Design_intent
                design_intent           : set_of CriticalParameter;
        END_ENTITY;

END_SCHEMA;
```

B.4 PRODUCT INTRODUCTION RESOURCE SCHEMA

```
SCHEMA Resource_Schema

        ENTITY Resource
                name            : string;
                description     : string;
        END_ENTITY;

        ENTITY HumanResource
                SUBTYPE OF Resource;
                managed_by      : set_of HumanResource_HumanResource;
                manages         : set_of HumanResource_HumanResource;
                has_parts       : set_of HumanResource_HumanResource;
                is_part_of      : set_of HumanResource_HumanResource;
                resource_of     : set_of HumanResource_InformationMap;
        END_ENTITY;

        ENTITY Team
                SUBTYPE OF HumanResource;
        END_ENTITY;

        ENTITY Person
                SUBTYPE OF HumanResource;
                designation             : string;
                department              : string;
                organisation            : string;
                office_address          : Address;
                email_address           : string;
                telephone_number        : string;
```

```
        fax_number          : string;
        skill               : set_of Person_Skill;
        WHERE
                SELF.has_parts := Null set;
END_ENTITY;

ENTITY Address
        street              : string;
        city                : string;
        postcode            : string;
        country             : string;
END_ENTITY;

ENTITY HumanResource_HumanResource
        resource1     : HumanResource;
        resource2     : HumanResource;
        role          : string;
END_ENTITY;

ENTITY Person_Skill
        person        : Person;
        skill         : Skill;
        level         : string;
END_ENTITY;

        ENTITY  Skill
                name          : string;
                description   : string;
        END_ENTITY;

END_SCHEMA;
```

B.5 PRODUCT SCHEMA

```
SCHEMA Product_Schema

        USE FROM ProductFunctionality_Schema (Functionality);
        USE FROM InformationMap_Schema (InformationMap);

ENTITY Product
        SUBTYPE OF Information;
    name                : string;
    description         : string;
    estimated_cost      : numeric;
    actual_cost         : numeric;
    why                 : set_of  Product_Functionality;
    has_parts           : set_of  Product_Product;
```

```
    is_part_of          : set_of Product_Product;
    top_forward         : set_of Product_Product;
    top_backward        : set_of Product_Product;
    how                 : set_of Product_ProductionMethod;
    drawing             : set_of Product_Drawing; (* information on drawings *)
    material            : set_of Product_Material; (* information  on material *)
    technicaldatadefn   : set_of Product_TechnicalData;
END_ENTITY;

ENTITY Assembly
    SUBTYPE OF Product;
END_ENTITY;

ENTITY Part
    SUBTYPE OF Product
    WHERE
        is_part          : SELF.has_part = Null set;
END_ENTITY;

ENTITY Product_Functionality
    product       : Product;
    functionality : Functionality;
END_ENTITY;

ENTITY Product_Product
    product1      : Product;
    product2      : Product;
    quantity      : numeric;
END_ENTITY;

ENTITY Product_TechnicalData
    product       : Product;
    technicaldata : TechnicalData;
END_ENTITY;

ENTITY TechnicalData
END_ENTITY;

(*  The following is a sample class that would be generated dynamically  *)

        ENTITY Blade
                SUBTYPE OF TechnicalData
            variant_id     : string;
            variant_name   : string;
            weight         : numeric;
            life_cycle     : numeric;
        END_ENTITY;

END_SCHEMA;
```

B.6 INFORMATION MAP SCHEMA

```
SCHEMA InformationMap_Schema

        USE FROM ProductFunctionality_Schema (Functionality);
        USE FROM Resource_Schema (Resource);
        USE FROM Product_Schema (Product);

    ENTITY InformationMap
        (*
            Its sub-entities  would be generated dynamically and the definitions of
            these entities would contain references to the entities in other
            information bases.
        *)
    END_ENTITY;

    (* Following are the sample Information Map classes *)

    ENTITY Blade_geometry
            SUBTYPE OF InformationMap;
            length          : numeric;
            width           : numeric;
    END_ENTITY;

    ENTITY Material_property
            SUBTYPE OF InformationMap;
            name            : string;
            density         : numeric;
            uts             : numeric;
            proof_stress    : numeric;
    END_ENTITY;

    ENTITY Technical_Report_on_Blade
            SUBTYPE OF InformationMap;
            title                           : string;
            part_description                : string;
            material_details                : Material_property;
            blade_geometry_details          : Blade_geometry;
        END_ENTITY;

END_SCHEMA;
```

B.7 META MODEL SCHEMA

```
SCHEMA MetaModel_Schema;

        ENTITY MetaDatabase
                name            : string;
                description     : string;
                path            : string;
                classes         : set_of MetaClass;
        END_ENTITY;

        ENTITY MetaClass
                name            : string;
                description     : string;
                superclass      : set_of MetaClass
                has_attributes  : set_of MetaClass_MetaAttribute;
        END_ENTITY;

        ENTITY MetaAttribute
                name            : string;
                description     : string;
                type            : MetaAttributetype;
        END_ENTITY;

        ENTITY MetaAttributetype
                name            : string;
                description     : string;
        END_ENTITY;

        ENTITY MetaClass_MetaAttribute
                mclass          : MetaClass;
                mattribute      : MetaAttribute;
        END_ENTITY;

        (* Following are the data structures that represent the definition
                of the attribute maps *)

        ENTITY MetaAttributeMap
                target          : DCATriplet;   (* database, class, attribute triplet *)
                source          : set_of DCATriplet;   (* database, class, attribute triplet *)
                derivation_rule : string;
        END_ENTITY;

        ENTITY DCATriplet
                database        : MetaDatabase;
                class           : MetaClass;
                attribute       : MetaAttribute;
        END_ENTITY;
END_SCHEMA;
```

APPENDIX C

PRODUCT INFORMATION

C.1 PRODUCT UNITS AND PRODUCTION METHODS

A component of the product may be either purchased or manufactured. Thus the source of the part or component is either a supplier or a manufacturing method that manufactures it. The output information of a manufacturing method would point to the corresponding part. The relationships between the production method and the physical structure of the product are shown in Figure C.1. The production method would have input, control, resource and output that can be represented as sets of associations between the production method, the corresponding product units and the resources. These associations can be defined in a similar manner as it has been done for product introduction process using information maps.

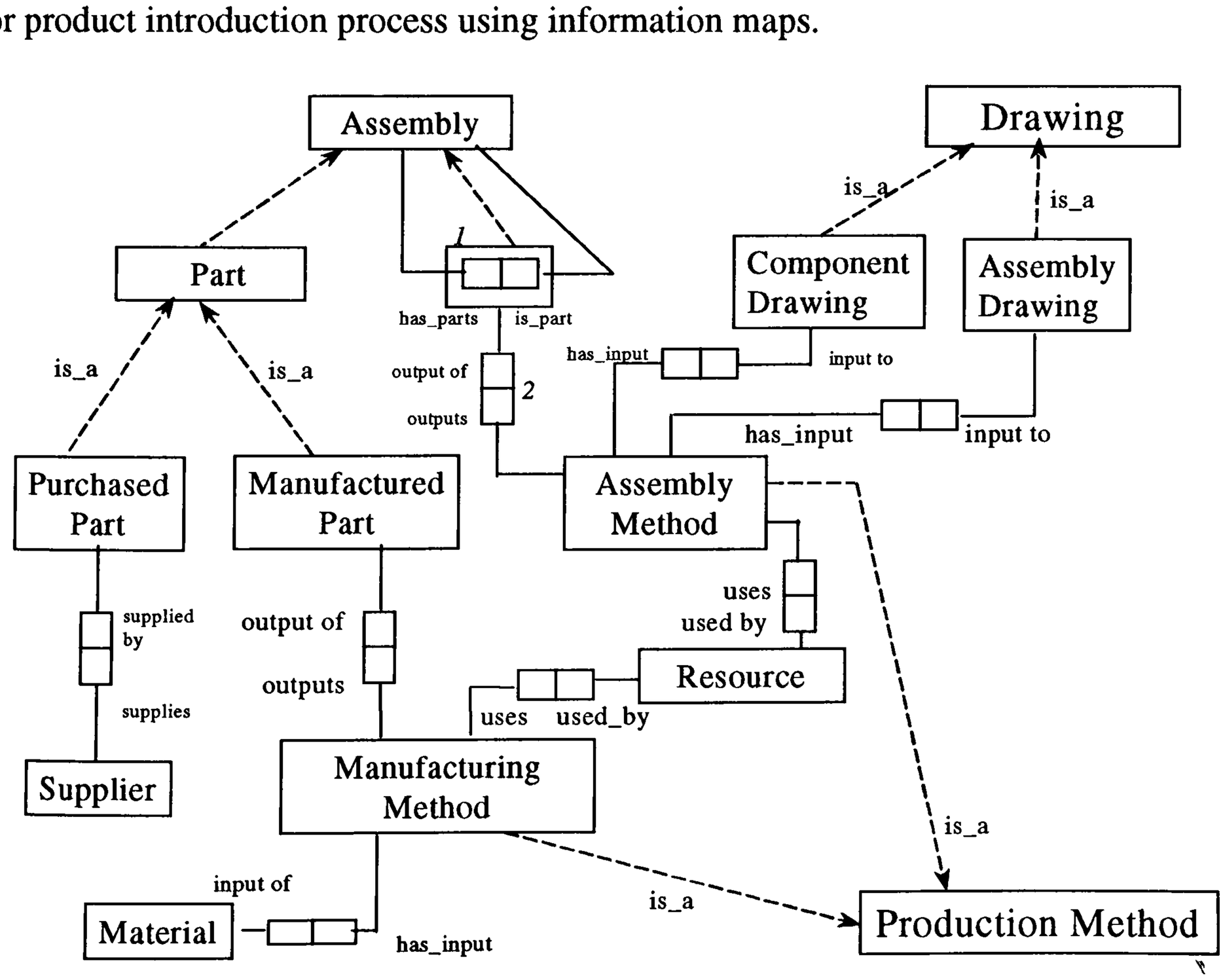


Figure C.1 Relationships between product units and production method

C.2 PRODUCT UNITS AND DRAWINGS

Data structures for representing information on drawing files of the product units and relevant associations are shown as follows. The following entities would be included in the product data model, and a link from the data structure *Product* to the information on drawings would be provided through the attribute *Drawing* of the type *Product_Drawing* association.

```
ENTITY Drawing
        name            :   string;
        description     :   string;
        has_parts       :   set_of  Drawing_Drawing;
        is_part_of      :   set_of  Drawing_Drawing;
        location        :   Hardware_Software_File;
        (*      Location of a drawing would link to the computer,
                software and the file in which the drawing is stored.
        *)
END_ENTITY;

 ENTITY AssemblyDrawing, PartDrawing
                SUBTYPE OF  Drawing;

    ENTITY Product_Drawing
            product         :   Product;
            drawing         :   Drawing;
            represents      :   string;  (* description of what the drawing represents *)
    END_ENTITY;
```

C.3 PRODUCT UNITS AND MATERIALS

Data structures for representing the information on materials of the product units and relevant associations are shown as follows. The following entities would be included in the product data model, and a link from the data structure *Product* to the information on material would be provided through the attribute *Material* of the type *Product_Material* association.

```
ENTITY Material
        name                :   string;
        description         :   string;
        materialdatadefn    :   set_of Material_MaterialData;
```

```
        source                  :   set_of Material_Supplier;
        why or where_used       :   set_of Material_Product;
END_ENTITY;

ENTITY Product_Material
        product                 :  Product;
        material                :  Material;
        quantity                :  numeric;     (* quantity of material required *)
END_ENTITY;

ENTITY Material_Product
        material                :  Material;
        product                 :  Product;
END_ENTITY;

ENTITY Material_Supplier
        material                :  Material;
        supplier                :  Supplier;
END_ENTITY;

ENTITY MaterialData
        (* The sub data structures would be generated dynamically *)
END_ENTITY;
```

C.4 PRODUCT FUNCTIONALITY AND PARAMETERS

The classes and attributes that are necessary to represent the relationships between product functionality and parameters are shown in Tables C.1 to C.6.

Table C.1 Data structure for parameters

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Parameter***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string
unit_of_measure	string

Table C.2 Data structure for associations among functionality and parameters

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Functionality_Parameter***
Super structure name	Nil
Attribute	**Domain**
Functionality	*Functionality*
Parameter	*Parameter*

Table C.3 Data structure for associations among functionality and response parameters

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Functionality_Response***
Super structure name	*Functionality_Parameter*
Attribute	**Domain**
latitude	*Range*
failure_mode	*set_of FailureMode*

Table C.4 Data structure for failure modes

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***FailureMode***
Super structure name	Nil
Attribute	**Domain**
name	string
description	string
limits	*Range*

Table C.5 Data structure for critical parameters

Schema name	IIM_Schema	
Sub-schema name	ProductFunctionality_Schema	
Data structure name	***CriticalParameter***	
Super structure name	*Parameter*	
Attribute	**Domain**	**Remarks**
nominal value	numeric	
range of tolerance	*Range*	
exp_date_of_definition	date	planned date
latitude_desired	*Range*	
latitude_achieved	*Range*	

Table C.6 Data structure for *Range*

Schema name	IIM_Schema
Sub-schema name	ProductFunctionality_Schema
Data structure name	***Range***
Super structure name	*Nil*
Attribute	**Domain**
lower_limit	numeric
upper_limit	numeric

C.5 INFORMATION USERS

Data structures necessary for providing security over the information in the databases are shown as follows:

```
ENTITY Group
        SUBTYPE OF HumanResource;
        accessrights    :   set_of User_Information_Accessrights;
END_ENTITY;

ENTITY InformationUser
        SUBTYPE OF HumanResource;
        host_log        :   Host_Logging;
        database_log    :   Database_Logging;
        access_rights   :   set_of User_Information_Accessrights;
END_ENTITY;
```

```
ENTITY Host_Logging
        host            :  Host;
        log             :  Logging;
END_ENTITY;

ENTITY Database_Logging
        database        :  Database;
        log             :  Logging;
END_ENTITY;

ENTITY Logging
        login           :  string;
        password        :  string;              (* encrypted and stored *)
END_ENTITY;

TYPE AccessRights = ENUMERATION OF
        (select, update, delete, insert);
END_TYPE;

ENTITY User_Information_Accessrights
        user            :  InformationUser;
        information     :  MetaClass;
        access          :  AccessRights;        (* enumerated data type *)
END_ENTITY;

ENTITY Host
        name                    : string;
        type                    : string;
        network_address         : string;
        port_id                 : string;
END_ENTITY;

ENTITY  Database
        name                    : string;
        type                    : string;
        software                : string;
END_ENTITY;
```

APPENDIX D

USER INTERFACE AND FORMS

D.1 INTRODUCTION

The prototype has been developed using MS-ACCESS and Visual Basic. This appendix shows the forms that have been designed and used in the development of the prototype, data structures defined to store the attribute maps, user-interface developed for managing the definition of attribute maps, and the user's guide.

D.2 FORMS DESIGNED

Table D.1 List of forms designed and their purpose

Name of the form and the VB file	Purpose
Forms	
AAtfrm (AAtfrm.frm)	Attribute Definition Maintenance
Aclatadf (aclatadf.frm)	Class-Attribute Addition
Aclatdrf (aclatdrf.frm)	Class-Attribute Drop
AclAtF (AClAtf.frm)	Class-Attribute Relationship Maintenance
Aclcrf (Aclcrf.frm)	Class Creation
Acldrf (acldrf.frm)	Class Drop
AclF (AClf.frm)	Class Definition Maintenance
Adbcl (Adbcl.frm)	Database-Class Relationship Maintenance
AdbNW (adbnw.frm)	New Database
AdbOp (ADBOP.frm)	Open a Database
AFunc (AFunc.frm)	Functionality Data Maintenance
Apdfniment (Apdfniment.frm)	Functionality-InformationMap Association Entry
Apdfnpdfnent (apdfnpdfnent.frm)	Functionality-Functionality Association Entry
Apdfnpdfnvw (apdfnpdfnvw.frm)	Functionality-Functionality Association View
Apdiment (Apdiment.frm)	Product-InformationMap Association Entry
Apdpdent (Apdpdent.frm)	Product-Product Association Entry
Apdpdvw (Apdpdvw.frm)	Product-Product Association View
Apiment (Apiment.frm)	Project-InformationMap Association Entry
Apimvw (Apimvw.frm)	Project-InformationMap Association View
Appent (Appent.frm)	Project-Project Association Entry
Appvw (Appvw.frm)	Project-Project Association View
AProc (AProc.frm)	Project Data Maintenance
Aprod (Aprod.frm)	Product Data Maintenance
AResrc (Aresrc.frm)	Resource Data Maintenance
Ariment (Ariment.frm)	Resource-InformationMap Association Entry
Arimvw (Arimvw.frm)	Resource-InformationMap Association View
Arrent (Arrent.frm)	Resource-Resource Association Entry
Arrvw (Arrvw.frm)	Resource-Resource Association View
AttrMap (AttrMap.frm)	Attribute Maps - Modify, Delete, View
Attypf (Attypf.frm)	Attribute-type Definition Maintenance
DbTreeview (DbTreeView.frm)	Database Tree View Tool
IInfModel (IInfModel.frm)	Integrated Information Model Manager Menu
MapAttr (MapAttr.frm)	Map Attributes i.e., attribute-map definition
MetaModel (AMetaModel.frm)	Meta Model Manager Menu
PrImTvw (PrImTvw.frm)	Project-Information Relationship Tree View
PrPrTvw (PrPrTvw.frm)	Project-Project Relationship Tree View
PrTreeview (PrTreeView.frm)	Project Tree View
	Run a given SQL
	View Data (General)

D.3 LIBRARIES DEVELOPED

Table D.2 List of libraries developed

Name of the Library	Purpose
AdeLib	Procedures for data maintenance
Aprope	Printing the contents of the meta-database i.e., Data Dictionary of the heterogeneous database
AtreeLib	Procedures for showing the data in tree view

D.4 META-MODEL MANAGER AND DATA MODEL MANAGER

Integrated Information Model for PI Process -- Meta Model Manager and Data Model Manager

Database	Class	Attribute	Attribute-Map	View	Help
New	New	New	Define Map	TreeView	Search
Open	Open	Existing	Alter Map	Data	
Close	Delete	Delete	Delete Map	Properties	
DB Classes	Super-class	Attribute Type ▶		SQL	
Go Back	Attribute ▶			Visual SQL	
	Create Class				
	Drop Class				
	Add an attribute				
	Drop Attribute				
	Set PrimaryKey				

Figure D.1 User interface of the meta-model manager and data model manager

D.5 ATTRIBUTE MAP MANAGER

The attribute map manager allows the creation, modification, deletion of the attribute map definitions. Two data structures have been designed to store the information on attribute maps:- one to store the information on the derived attribute (or destination attribute), and the other to store the information on the source attributes from which the destination attribute can be derived. When the attribute map manager has been implemented using relational database concepts, above two data structures [illegible]map_head' (figure D.2) and 'metaattrmap_tail'

(figure D.3) respectively. Figure D.4 shows the user interface for capturing the definition of an attribute map.

metaattrmap_head : Table

Field Name	Data Type	Description
id	AutoNumber	Identification code for an attribute map
dbname	Text	Database name
clname	Text	Class name
atname	Text	Attribute name
relation	Text	Relationship
express	Text	Expression

Figure D.2 Table structure of 'metaattrmap_head'

metaattrmap_tail : Table

Field Name	Data Type	Description
id	Number	Identification code for an attribute map
dbname2	Text	Database name
clname2	Text	Class name
atname2	Text	Attribute name
opname	Text	Operator name

Figure D.3 Table structure of 'metaattrmap_tail'

Table D.3 Details of figures

Figure Numbers	**Description**
Figures D.5 to D.7	user interface of meta-model manager
Figures D.8 to D.9	user interface of data model manager
Figure D.10	database structure in the tree view
Figures D.11 to D.14	user interface of project data manager
Figures D.15 and D.16	project tree and information map classes associated with an activity using tree view
Figures D.17 to D.19	activity dependency analysis tool

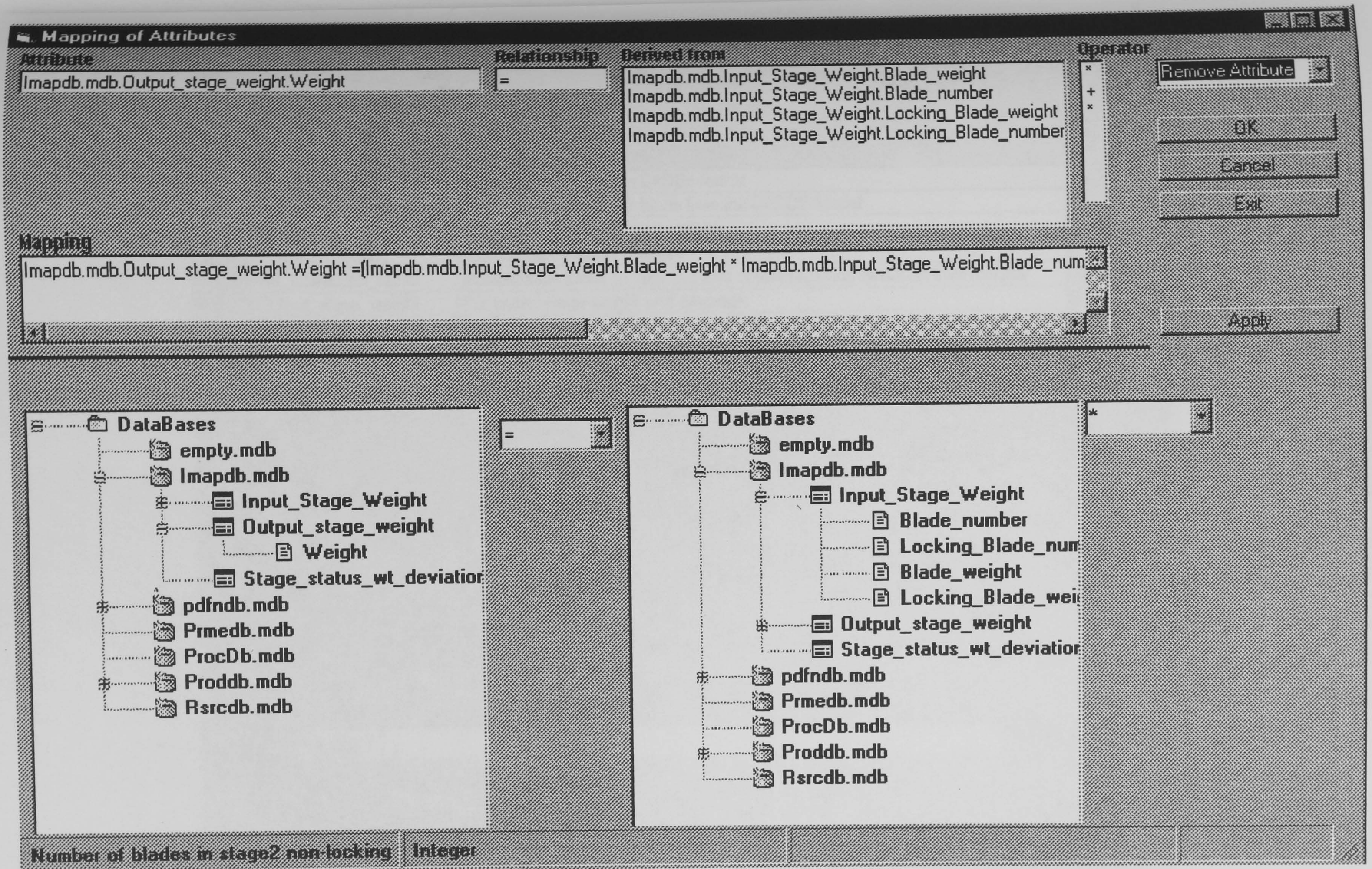


Figure D.4 User Interface for Defining an Attribute Map

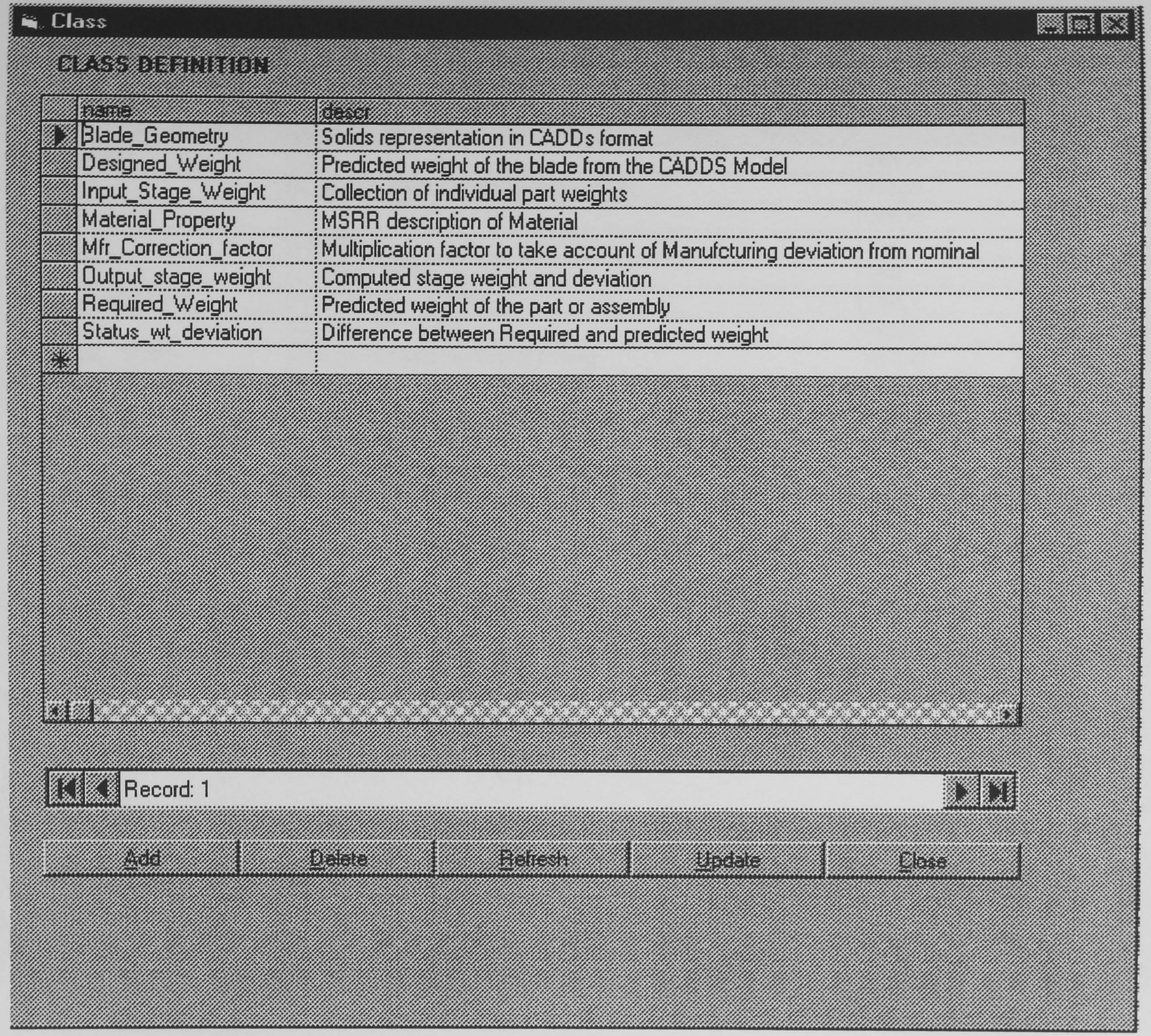


Figure D.5 Meta-model Manager - Class Definition Form

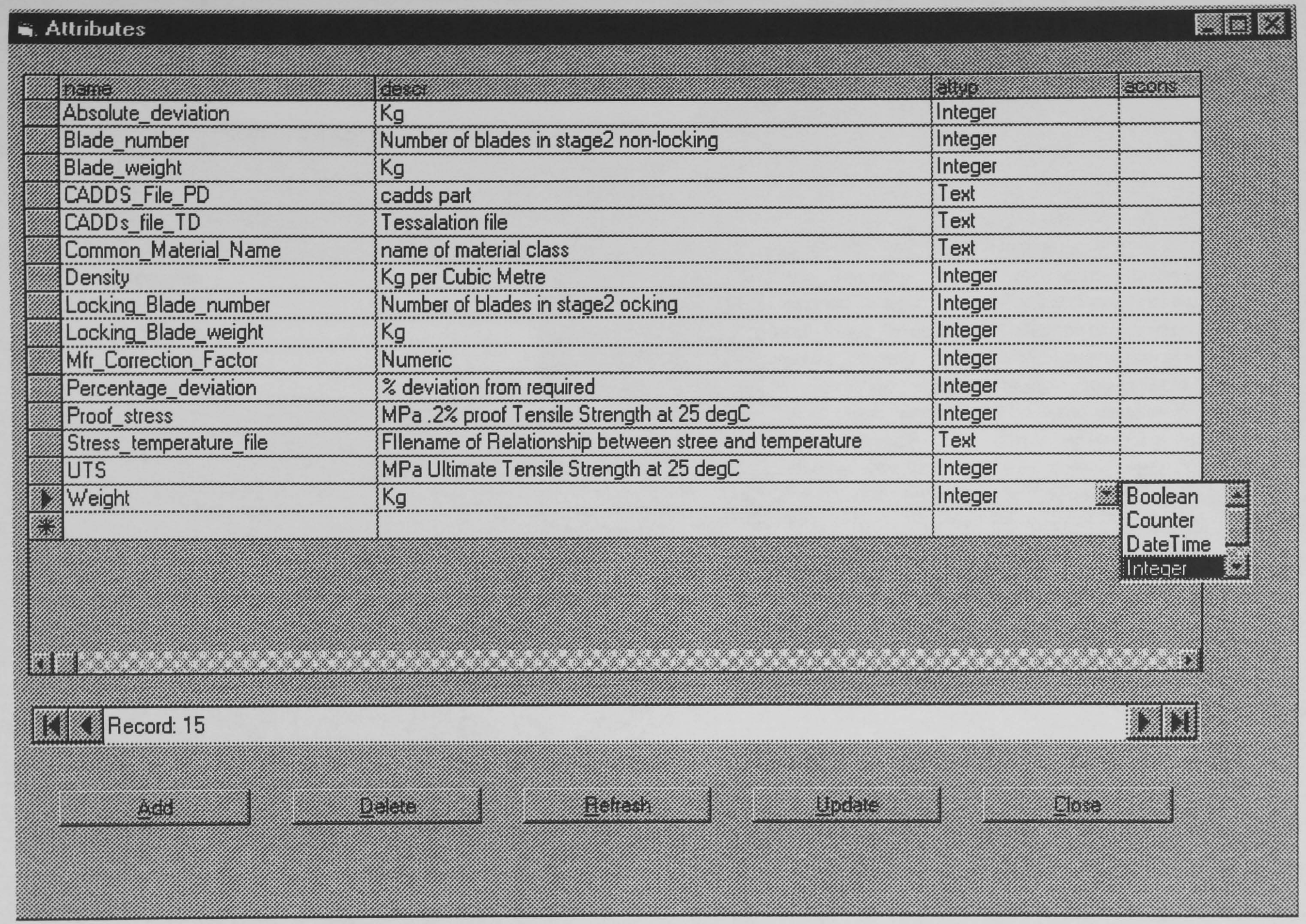


Figure D.6 Meta-model Manager - Attribute Definition Form

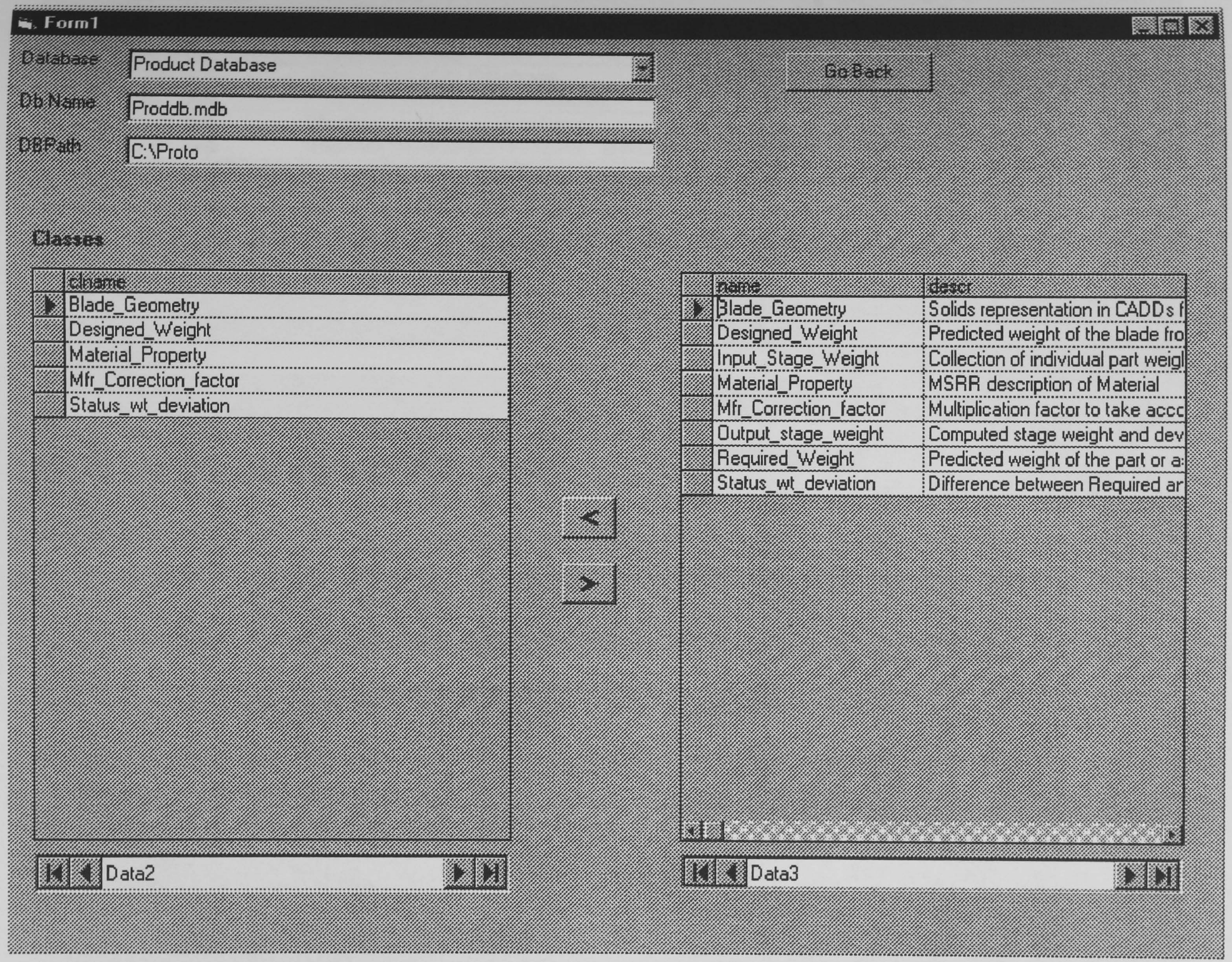


Figure D.7 Meta-model Manager - Database-Class Relationship Definition Form

Data Bases

	name	description	path
	empty.mdb	Empty Database	c:\Proto
	Imapdb.mdb	Information Map Database	C:\Proto
	pdfndb.mdb	Product Functionality Database	C:\Proto
	Prmedb.mdb	Production Method Database	C:\Proto
	ProcDb.mdb	Product Introduction Process Database	C:\Proto
▶	Proddb.mdb	Product Database	C:\Proto
	Rsrcdb.mdb	Product Introduction Resource Database	C:\Proto

Classes

	name	descr
	Blade_Geometry	Solids representation in CADDs format
▶	Material_Property	MSRR description of Material
	Mfr_Correction_facto	Multiplication factor to take account of Manufcturing deviati
	Designed_Weight	Predicted weight of the blade from the CADDS Model
	Status_wt_deviation	Difference between Required and predicted weight

Their Attributes

	name	descr
▶	Density	Kg per Cubic Metre
	UTS	MPa Ultimate Tensile Strength at 25 degC
	Proof_stress	MPa .2% proof Tensile Strength at 25 degC
	Stress_temperature_file	FIlename of Relationship between stree and temperatu
	Common_Material_Name	name of material class

Create

Go Back

Figure D.8 Data Model Manager - Class Creation Form

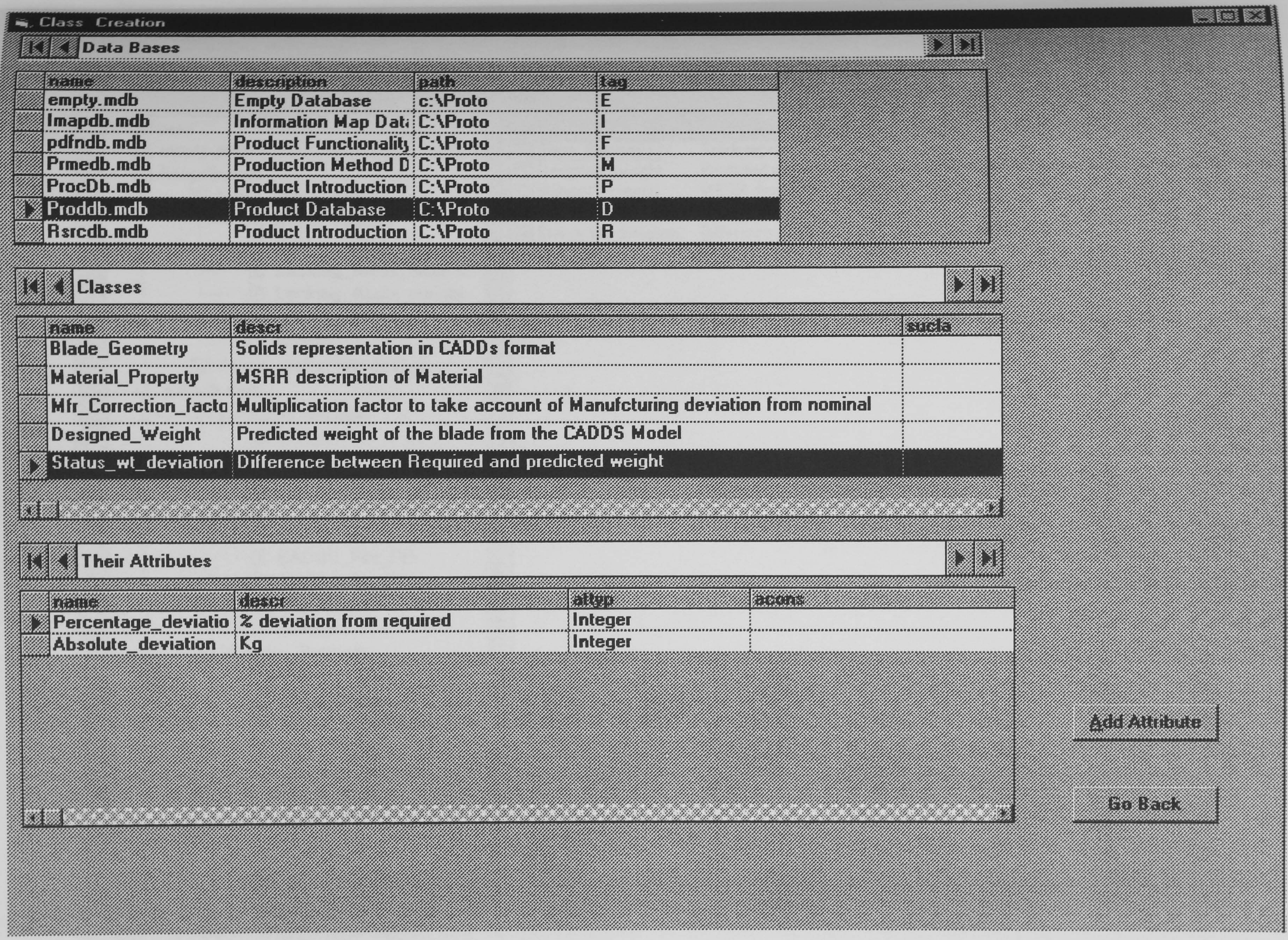


Figure D.9 Data Model Manager - Attribute Addition to an Existing Class Form

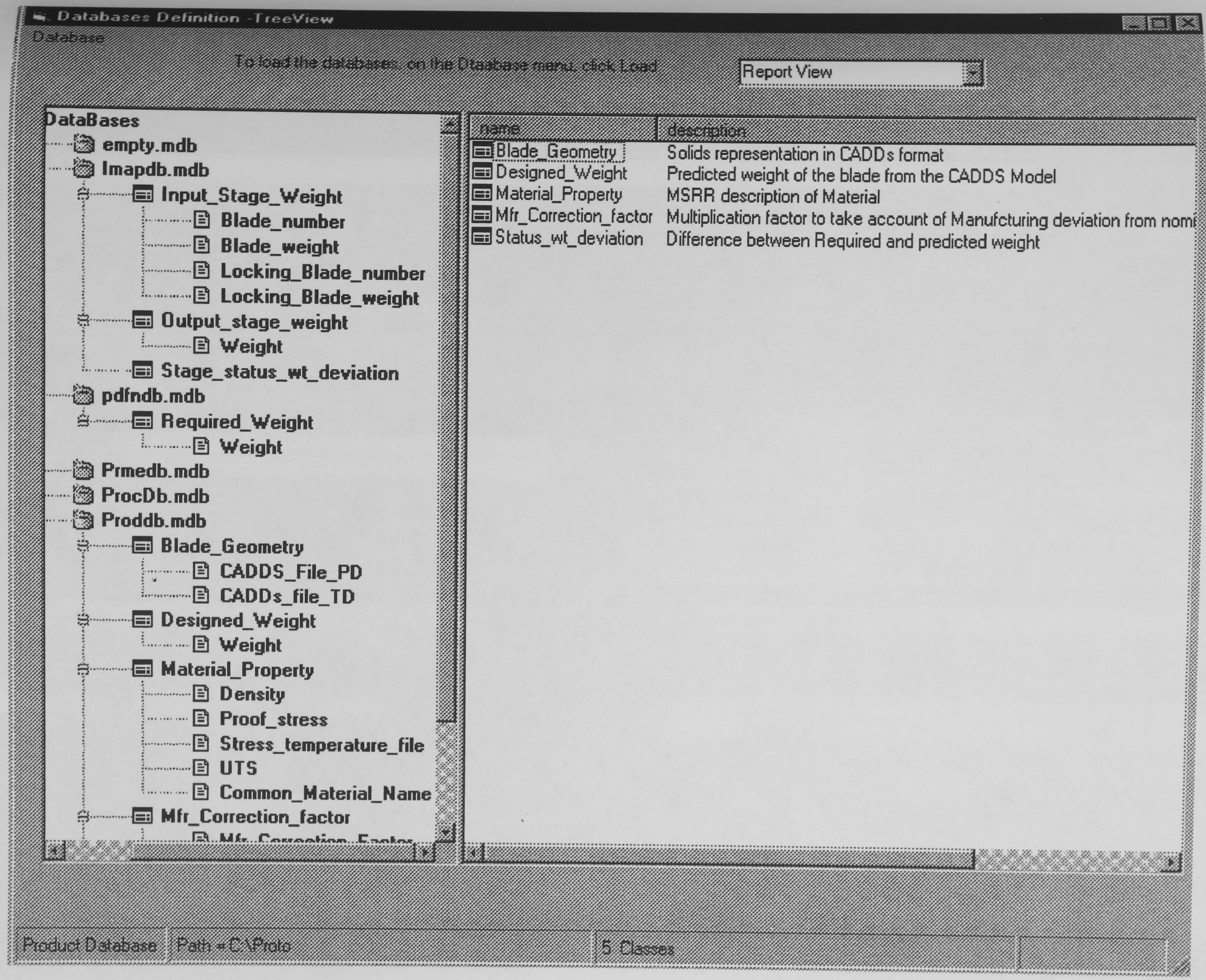


Figure D.10 Meta-model Manager - Data Dictionary Tree

PROCESS - DEFINITION

	name	description	
▶	Locking_blade_wt_evaluation	Compute the expected production weight from the CADDs model	
	Stage2_Weight_Evaluation	Compute the expected production weight from the individual part weights	
	weight_evaluation	Compute the expected production weight from the CADDs model	
*			

Record: 1

Add | Delete | Refresh | Update | Close

INFORMATION MAPS

Input | Control or Constraint | Resource | Output

	Input
▶	Blade_Geometry
	Material_Property
	Mfr_Correction_factor

Input Classes

	Overall Class List	descr
▶	Blade_Geometry	Solids representation in CADDs format
	Designed_Weight	Predicted weight of the blade from the CADDS Model
	Input_Stage_Weight	Collection of individual part weights
	Material_Property	MSRR description of Material
	Mfr_Correction_factor	Multiplication factor to take account of Manufcturing de
	Output_stage_weight	Computed stage weight and deviation
	Required_Weight	Predicted weight of the part or assembly
	Status_wt_deviation	Difference between Required and predicted weight

Information Map Classes or Tables

	id	CADDS_File_PD	CADDs_file_TD
▶	2	Lk12346_PD	Lk12346_TD

Input Instances

	id	CADDS_File_PD	CADDs_file_TD
▶	1	Lk12345_PD	Lk12345_TD
	2	Lk12346_PD	Lk12346_TD

Instances or Records

Figure D.11 Project Data Manager - Activity and Input Information Maps

PROCESS - DEFINITION

	name	description	
▶	Locking_blade_wt_evaluation	Compute the expected production weight from the CADDs model	
	Stage2_Weight_Evaluation	Compute the expected production weight from the individual part weights	
	weight_evaluation	Compute the expected production weight from the CADDs model	
*			

Record: 1

Add | Delete | Refresh | Update | Close

INFORMATION MAPS

Input | Control or Constraint | Resource | Output

	Input
▶	Blade_Geometry
	Material_Property
	Mfr_Correction_factor

Input Classes

	Overall Class List	descr
▶	Blade_Geometry	Solids representation in CADDs format
	Designed_Weight	Predicted weight of the blade from the CADDS Model
	Input_Stage_Weight	Collection of individual part weights
	Material_Property	MSRR description of Material
	Mfr_Correction_factor	Multiplication factor to take account of Manufcturing de
	Output_stage_weight	Computed stage weight and deviation
	Required_Weight	Predicted weight of the part or assembly
	Status_wt_deviation	Difference between Required and predicted weight

Information Map Classes or Tables

	id	CADDS_File_PD	CADDs_file_TD
▶	2	Lk12346_PD	Lk12346_TD

Input Instances

	id	CADDS_File_PD	CADDs_file_TD
▶	1	Lk12345_PD	Lk12345_TD
	2	Lk12346_PD	Lk12346_TD

Instances or Records

Figure D.11 Project Data Manager - Activity and Input Information Maps

PROCESS - DEFINITION

	name	description
▶	Locking_blade_wt_evaluation	Compute the expected production weight from the CADDs model
	Stage2_Weight_Evaluation	Compute the expected production weight from the individual part weights
	weight_evaluation	Compute the expected production weight from the CADDs model
*		

Record: 1

Add | Delete | Refresh | Update | Close

INFORMATION MAPS

Input | **Control or Constraint** | Resource | Output

	Control or Constraint
▶	Required_Weight

Control or Constraint Classes

	id	Weight
▶	1	24

Control or Constraint Instances

< >

	Overall Class List	descr
▶	Blade_Geometry	Solids representation
	Designed_Weight	Predicted weight of t
	Input_Stage_Weight	Collection of individu
	Material_Property	MSRR description of
	Mfr_Correction_factor	Multiplication factor t
	Output_stage_weight	Computed stage wei
	Required_Weight	Predicted weight of t
	Status_wt_deviation	Difference between I

Information Map Classes or Tables

	id	Weight
▶	1	24

Instances or Records

Figure D.12 Project Data Manager - Activity and Constraint Information Maps

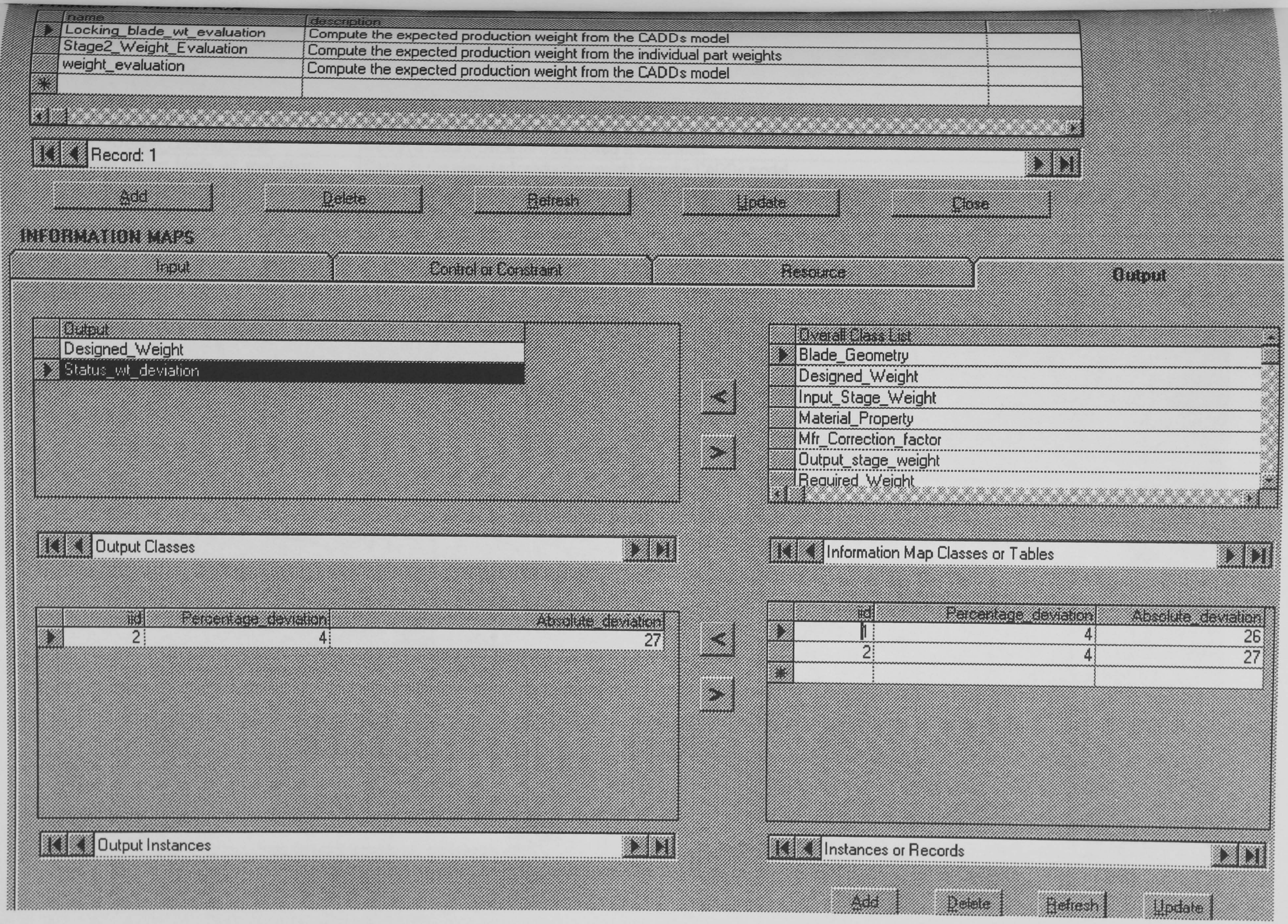


Figure D.13 Project Data Manager - Activity and Output Information Maps

Form1

PROCESS DEFINITION

	name	description
	Design_Blade	Design a blade for the required specification
	Design_Locking_Blade	Design a locking blade for the required specification
	Locking_blade_wt_evaluation	Compute the expected production weight from the CADDs model
▶	Stage2_Weight_Evaluation	Compute the expected production weight from the individual part weights
	weight_evaluation	Compute the expected production weight from the CADDs model
*		

Record: 4

Add | Delete | Refresh | Update | Close

Super Process | Sub Process | Predecessor Process | SuccessorProcess

	Predecessor Process
▶	weight_evaluation
	Locking_blade_wt_evaluation

Data2

	Overall Process List
▶	Design_Blade
	Design_Locking_Blade
	Locking_blade_wt_evaluation
	Stage2_Weight_Evaluation
	weight_evaluation

Data3

Figure D.14 Project Data Manager - Sequencing of Activities

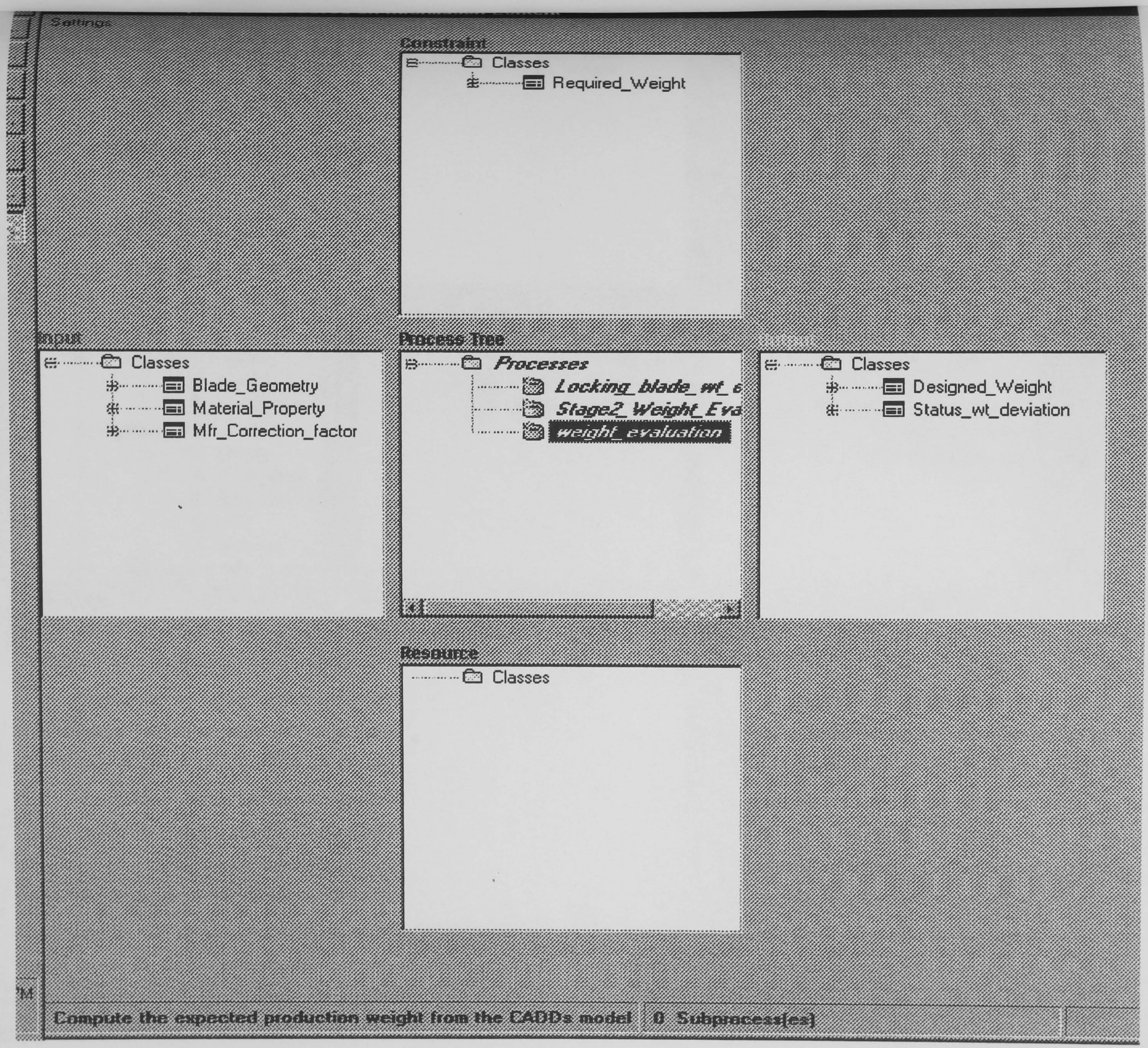


Figure D.15 Tool - Information Map Classes Associated With An activity

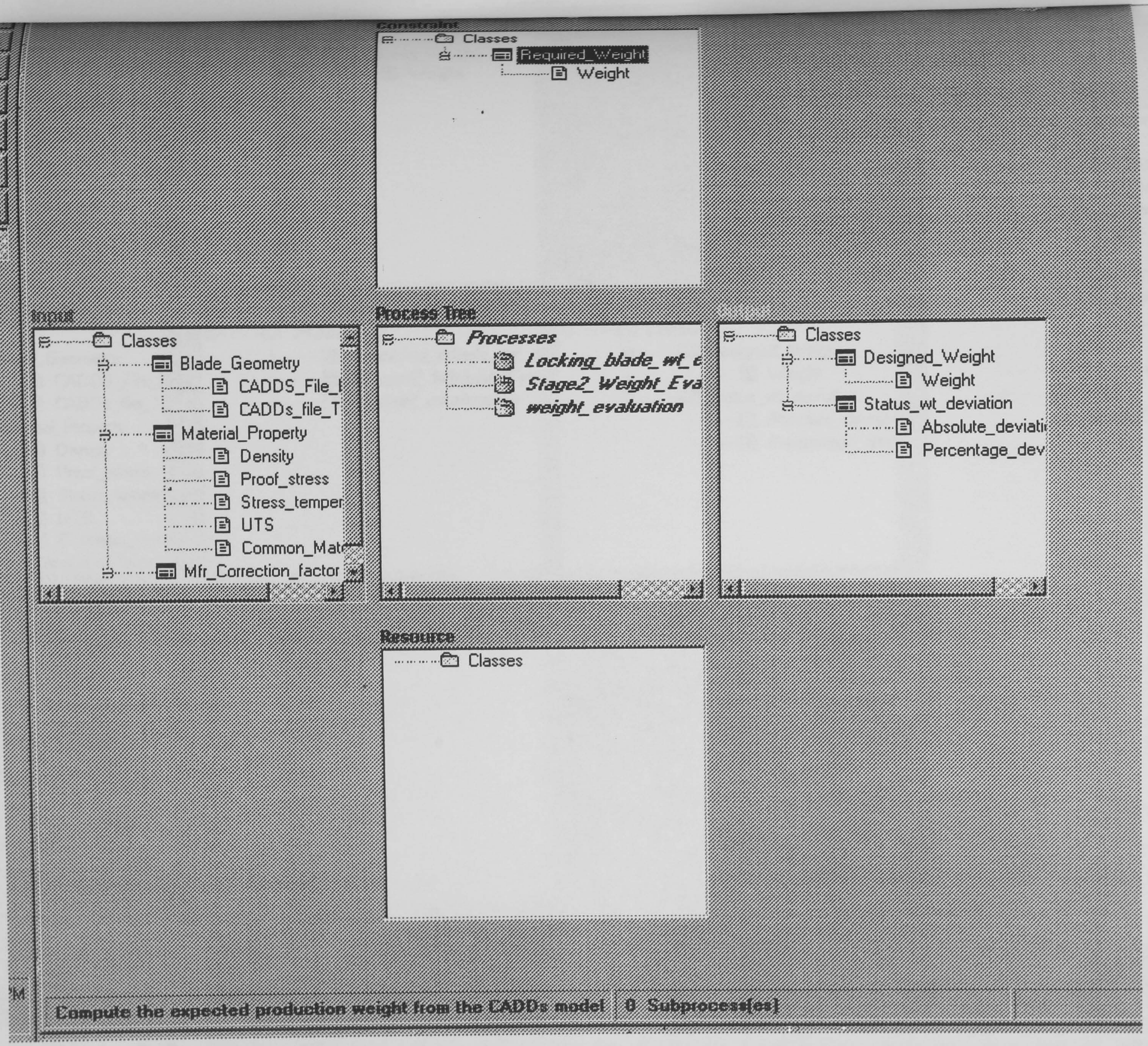


Figure D.16 Tool - Information Maps Associated With An activity (Detail Level i.e., Showing Attributes)

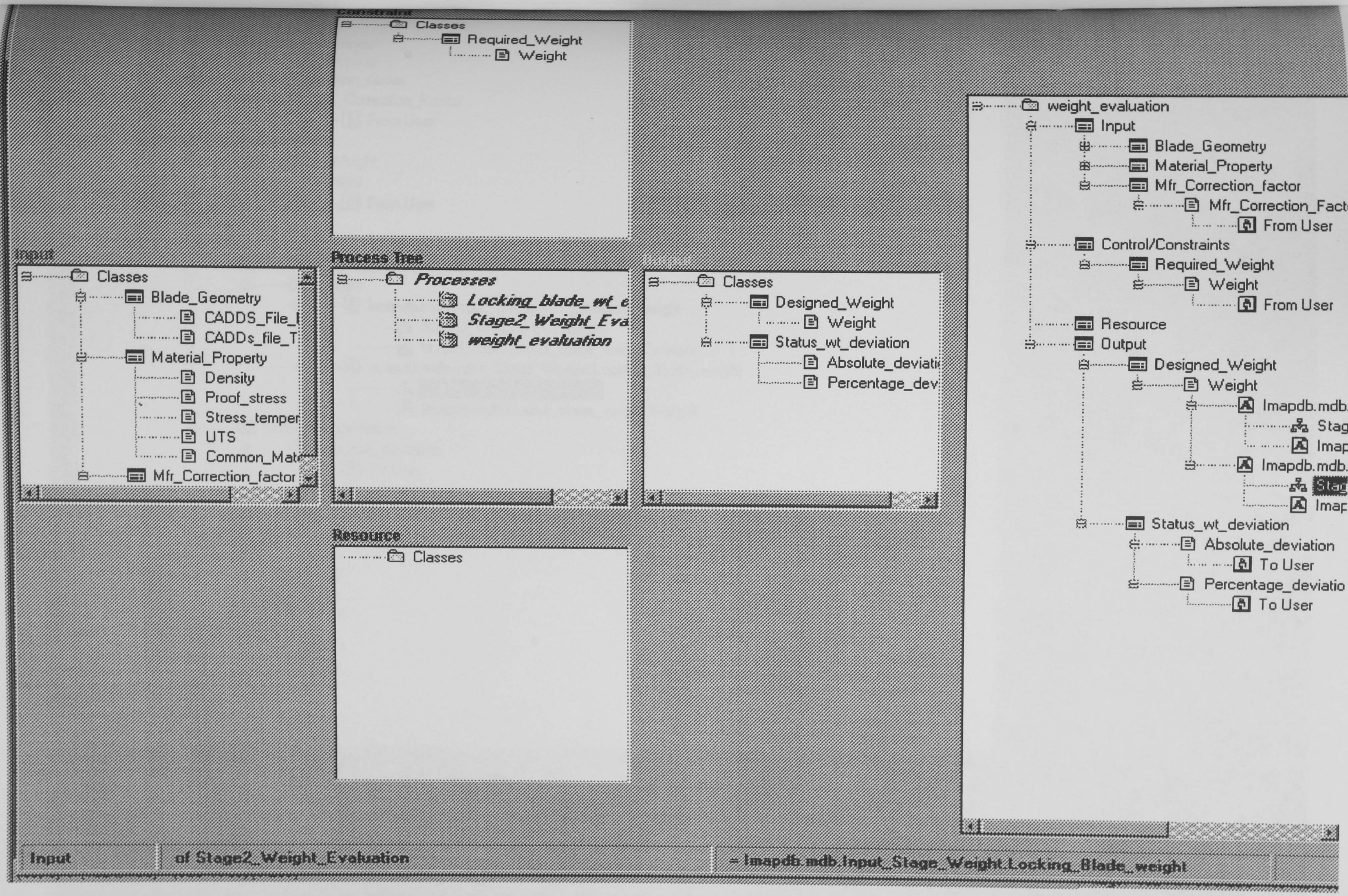


Figure D.17 Tool - Information Maps Associated With An Activity And Dependencies Among Activities Based On Information Content

- weight_evaluation
  - Input
    - Blade_Geometry
    - Material_Property
    - Mfr_Correction_factor
      - Mfr_Correction_Factor
        - From User
  - Control/Constraints
    - Required_Weight
      - Weight
        - From User
  - Resource
  - Output
    - Designed_Weight
      - Weight
        - Imapdb.mdb.Input_Stage_Weight.Blade_weight
          - Stage2_Weight_Evaluation
          - Imapdb.mdb.Output_stage_weight.Weight
        - Imapdb.mdb.Input_Stage_Weight.Locking_Blade_weight
          - Stage2_Weight_Evaluation
          - Imapdb.mdb.Output_stage_weight.Weight
    - Status_wt_deviation
      - Absolute_deviation
        - To User
      - Percentage_deviation
        - To User

Input | of Stage2_Weight_Evaluation | = Imapdb.mdb.Input_Stage_Weight.Locking_Blade_weight

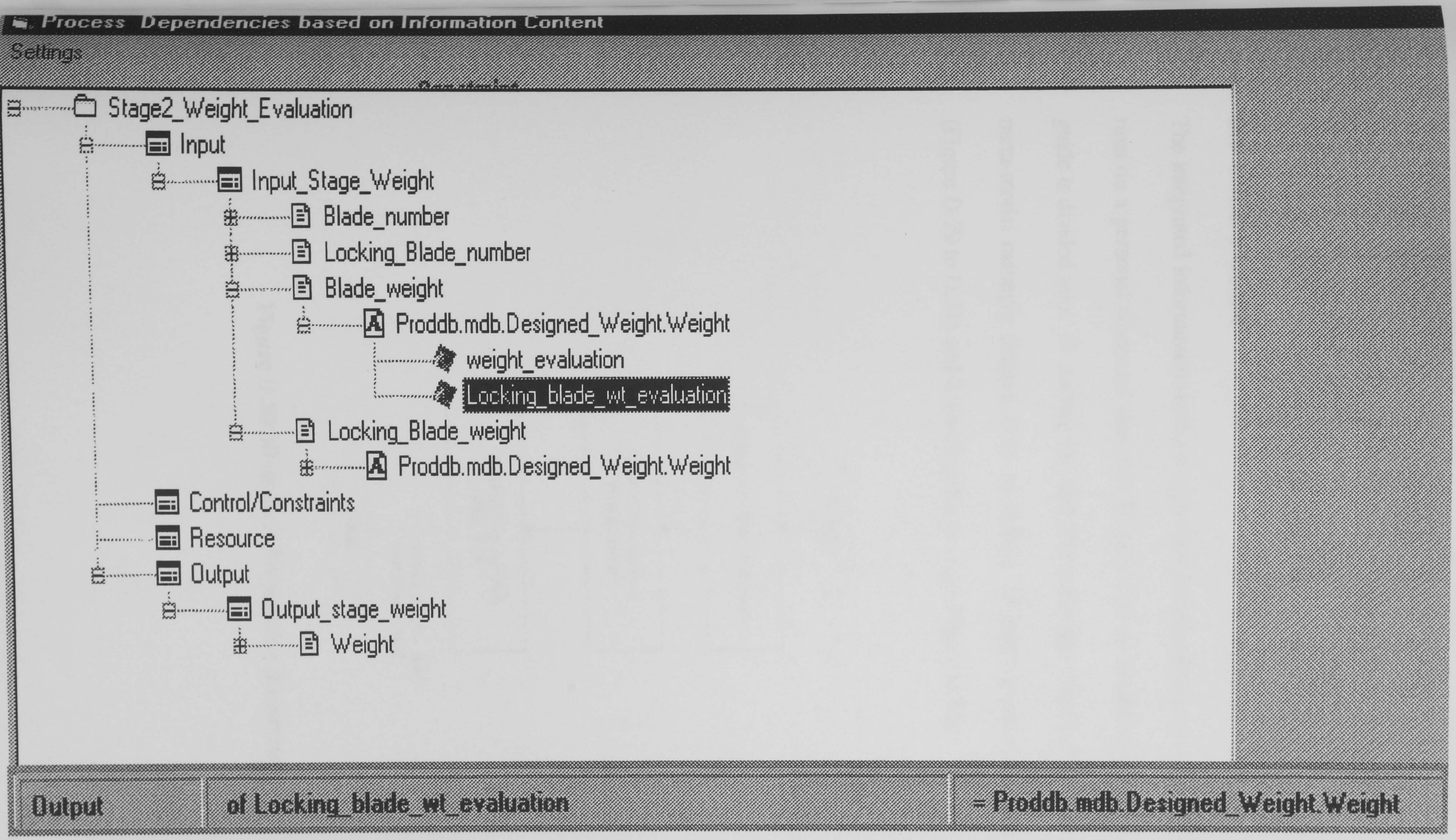


Figure D.19 Tool - Dependencies Among Activities (Predecessor) Based On Information Content

D.6 USER'S GUIDE

The integrated information model for product introduction process (IIM_PIP) prototype runs on a personal computer that runs Windows-95 or Windows-NT. The overall user's guide is divided into: ① starting the IIM_PIP prototype (Figure D.20), ② user's guide to meta-model manager (Figure D.21 to D.28), ③ user's guide to project data manager (Figure D.29 to D.31) and ⑤ user's guide to tools (Figure D.32).

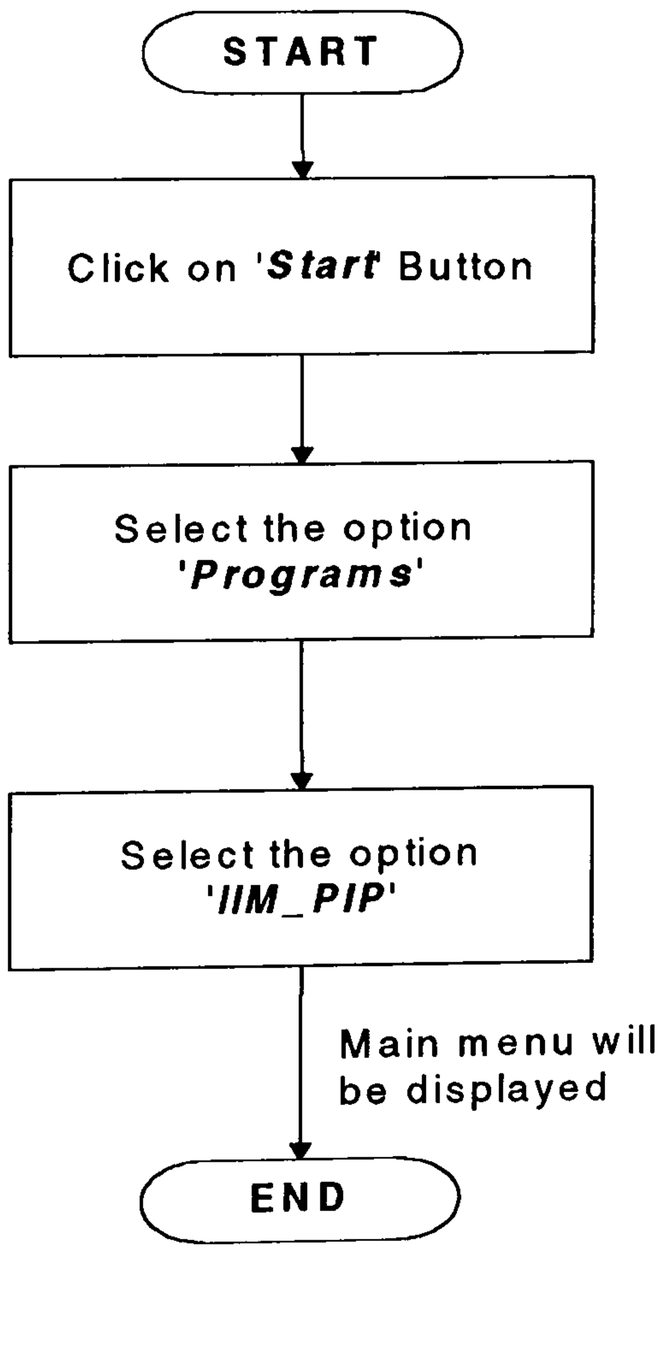


Figure D.20 Flowchart for starting the prototype

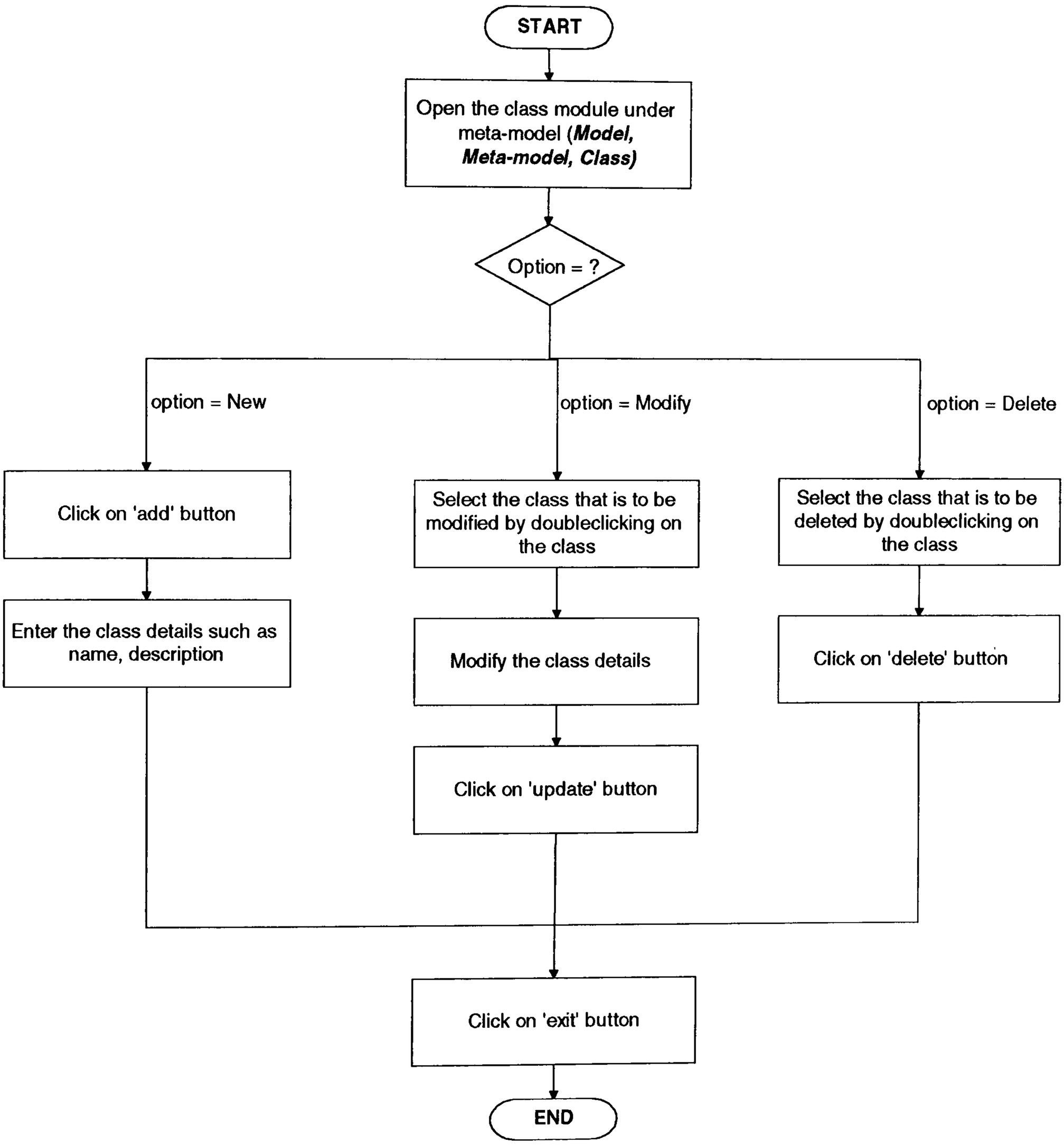


Figure D.21 Flowchart for meta-class maintenance

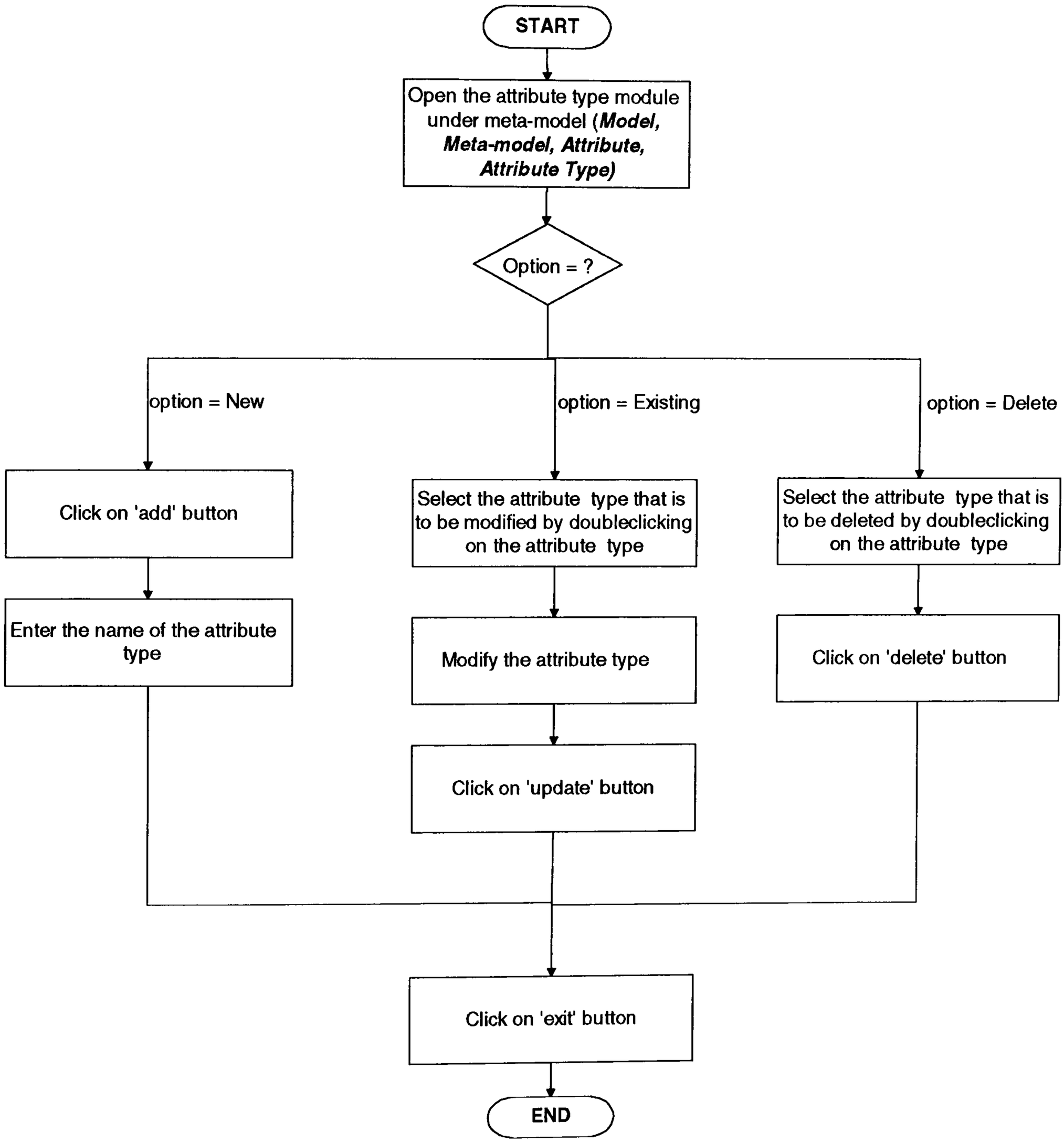


Figure D.22 Flowchart for meta-attribute type maintenance

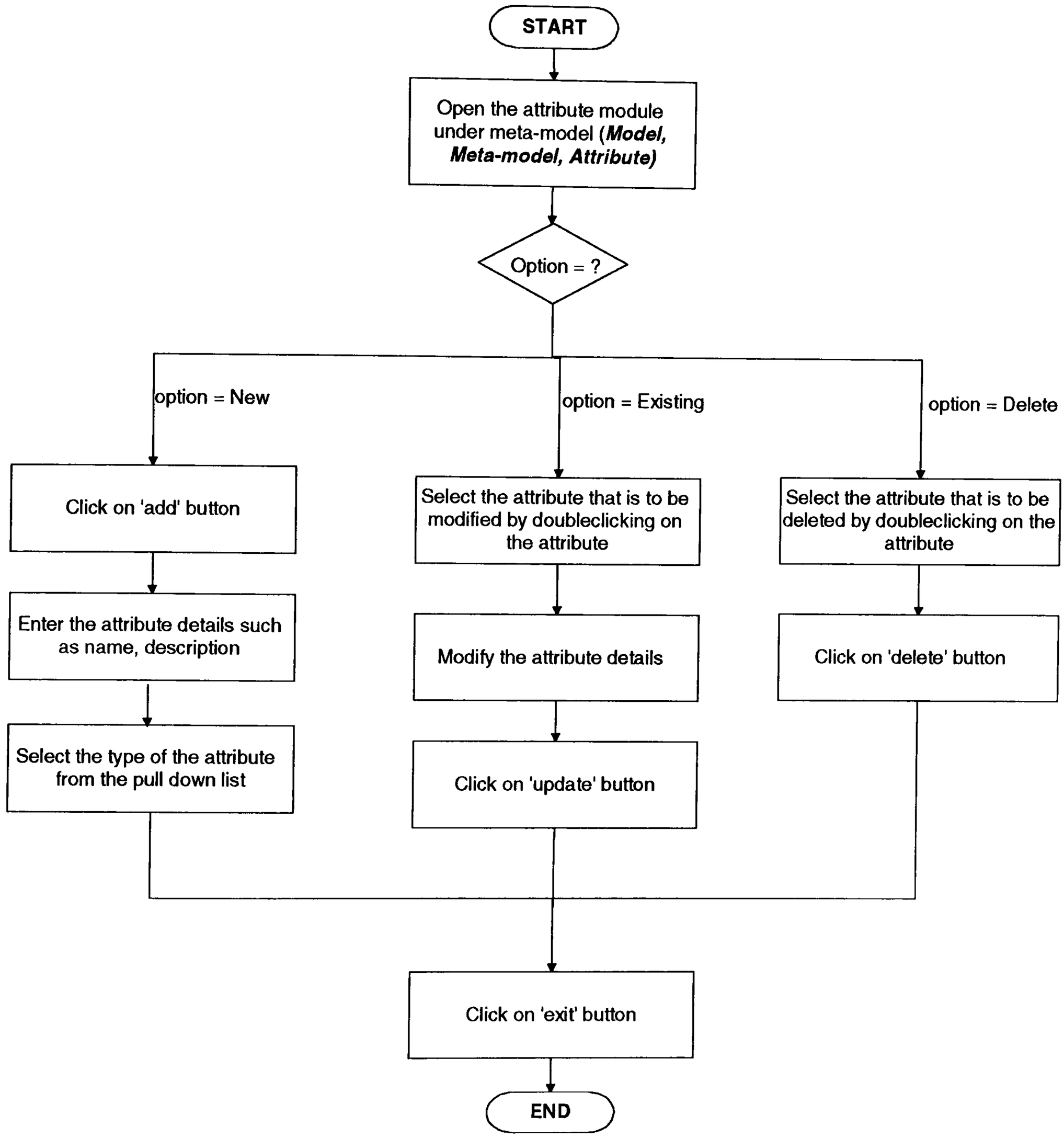


Figure D.23 Flowchart for meta-attribute maintenance

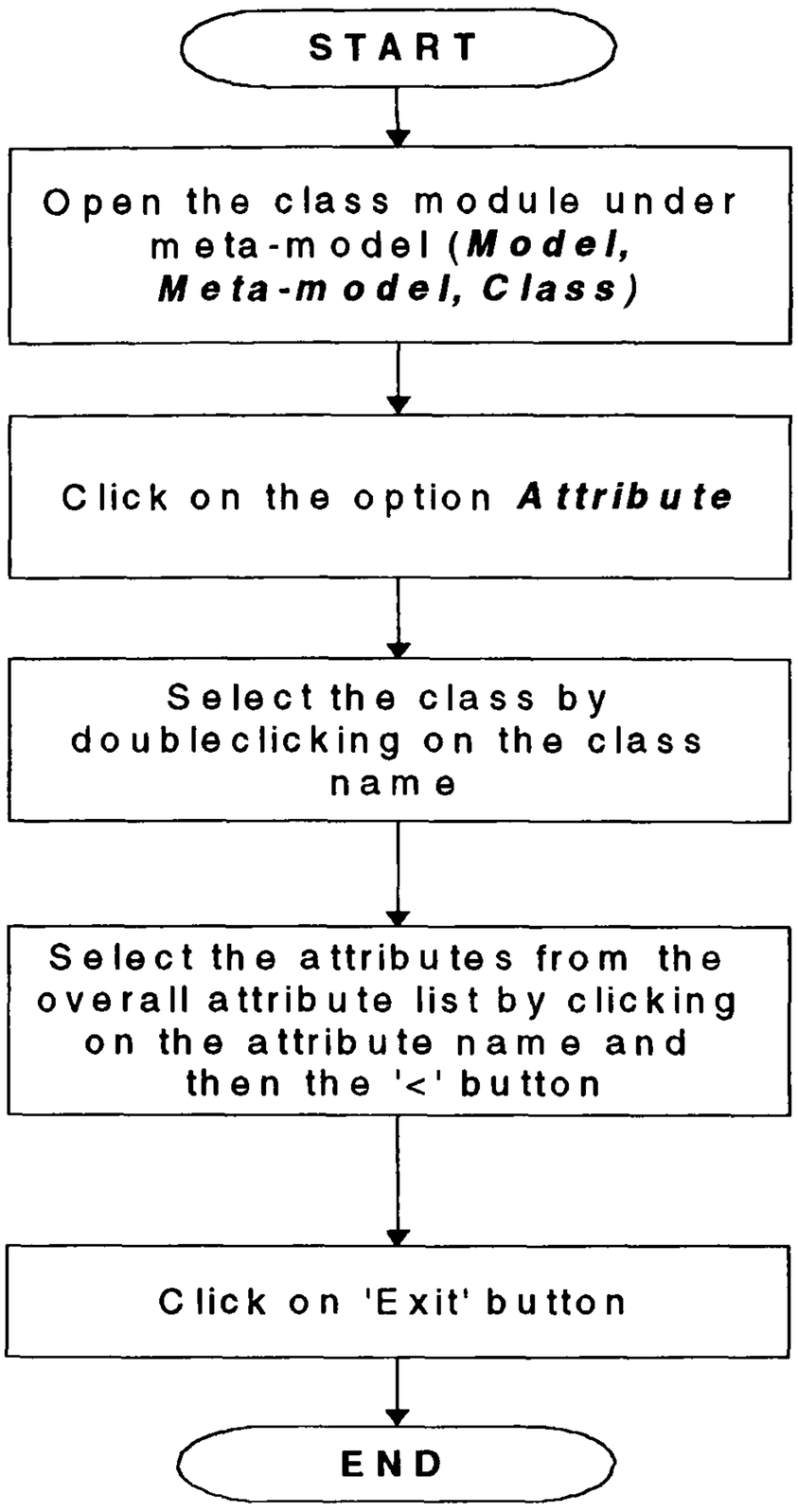


Figure D.24 Flowchart for relating meta-class and meta-attributes

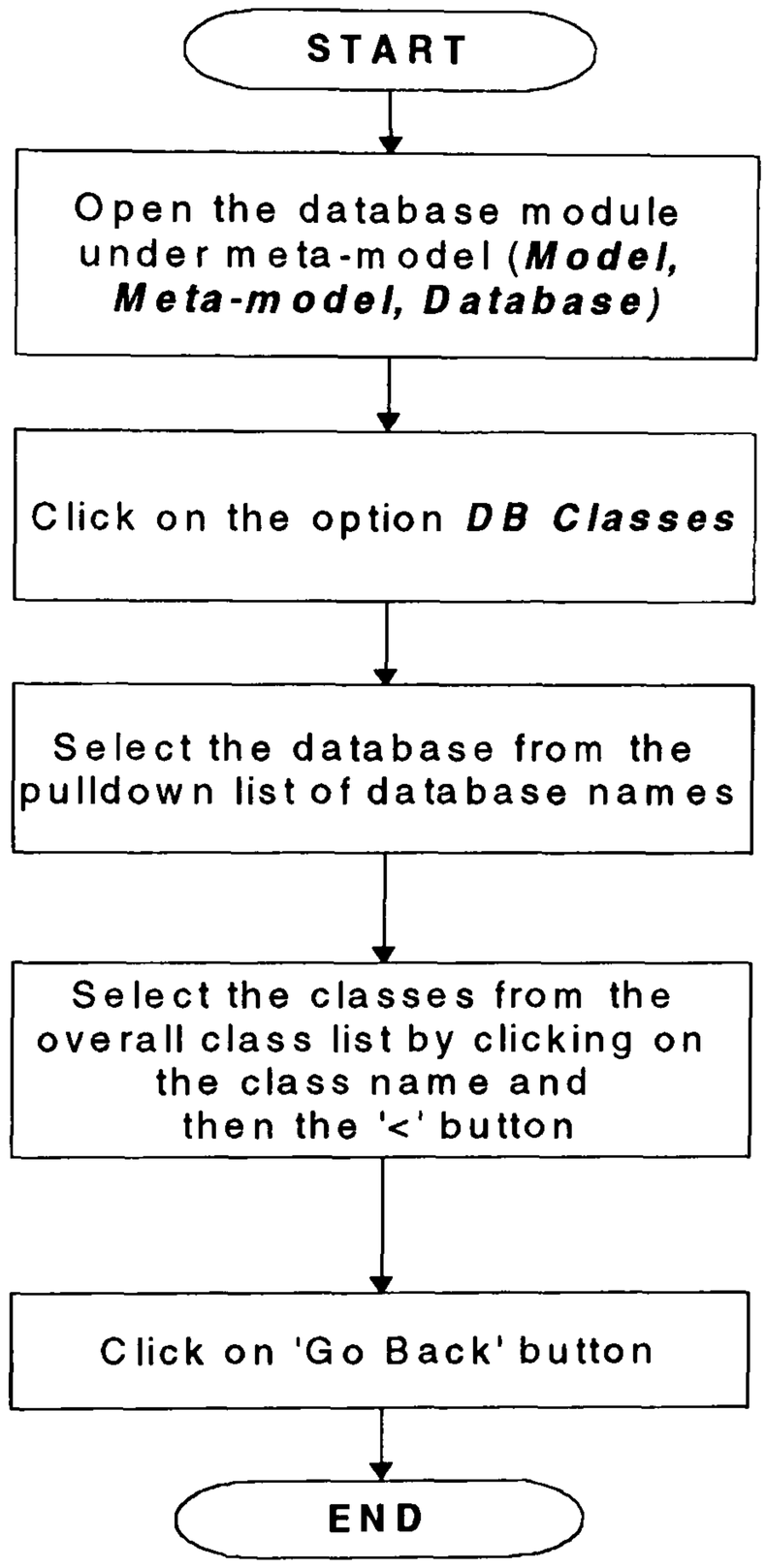


Figure D.25 Flowchart for relating meta-database and meta-class

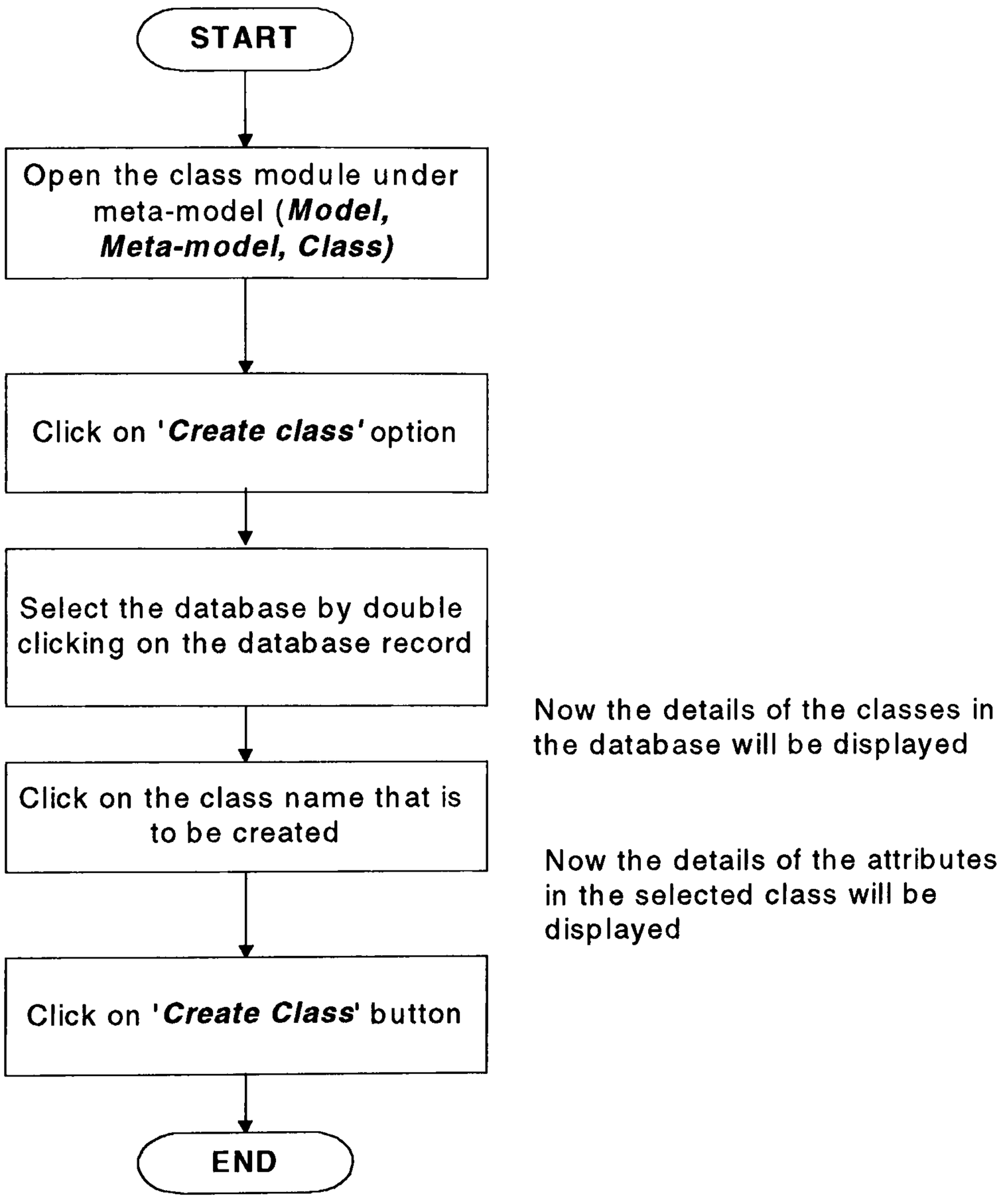


Figure D.26 Flowchart for creating a class

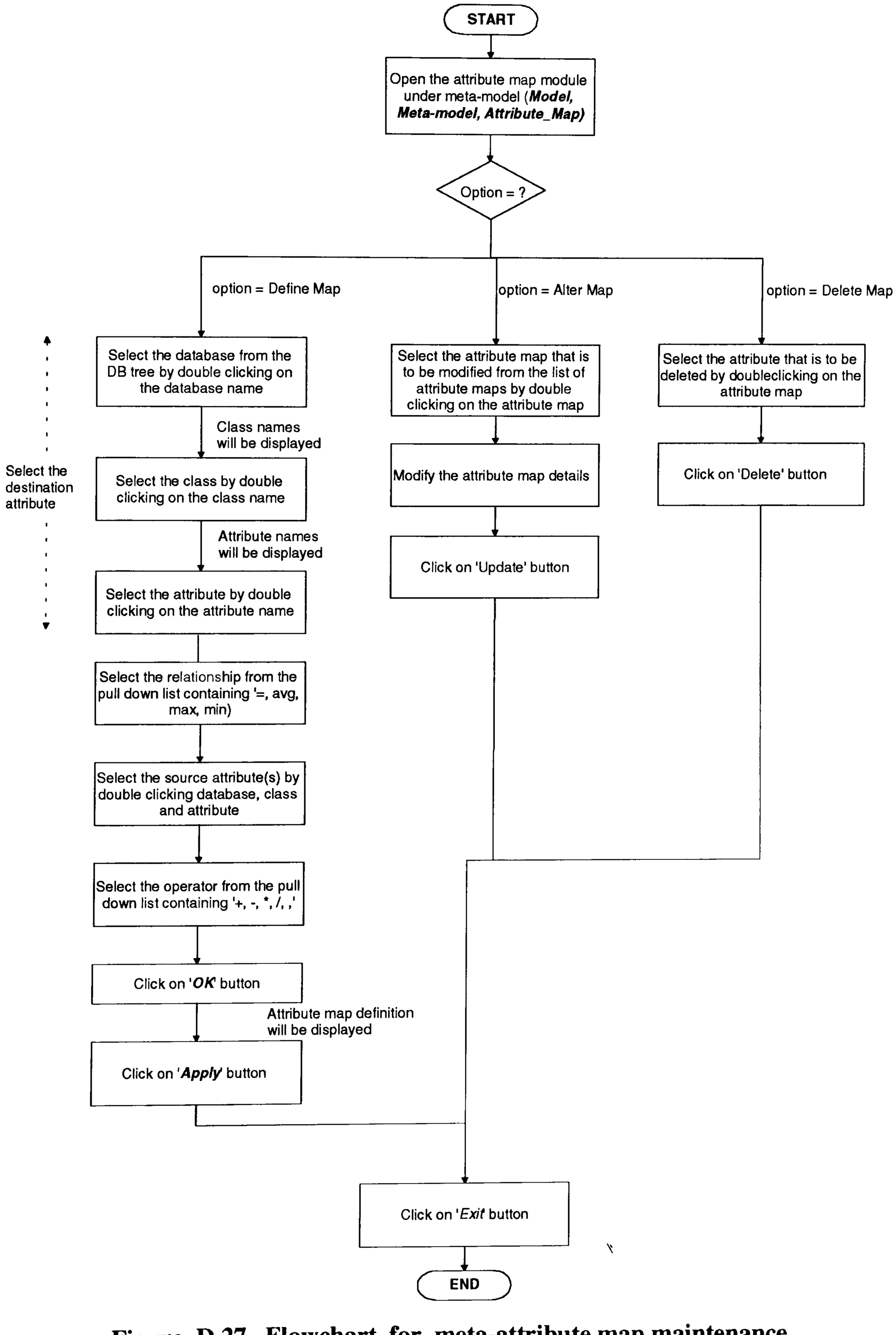


Figure D.27 Flowchart for meta-attribute map maintenance

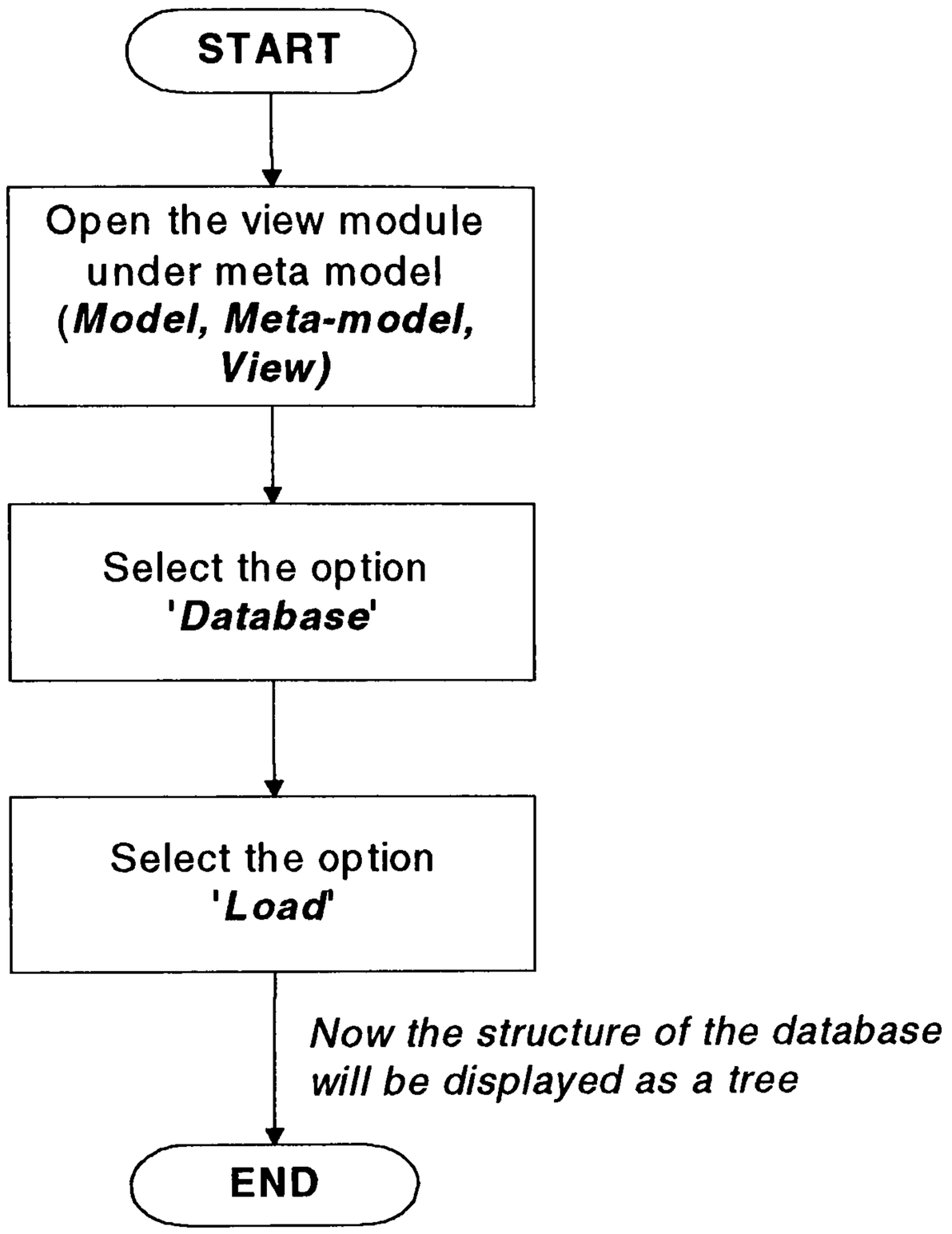


Figure D.28 Flowchart for viewing the database tree

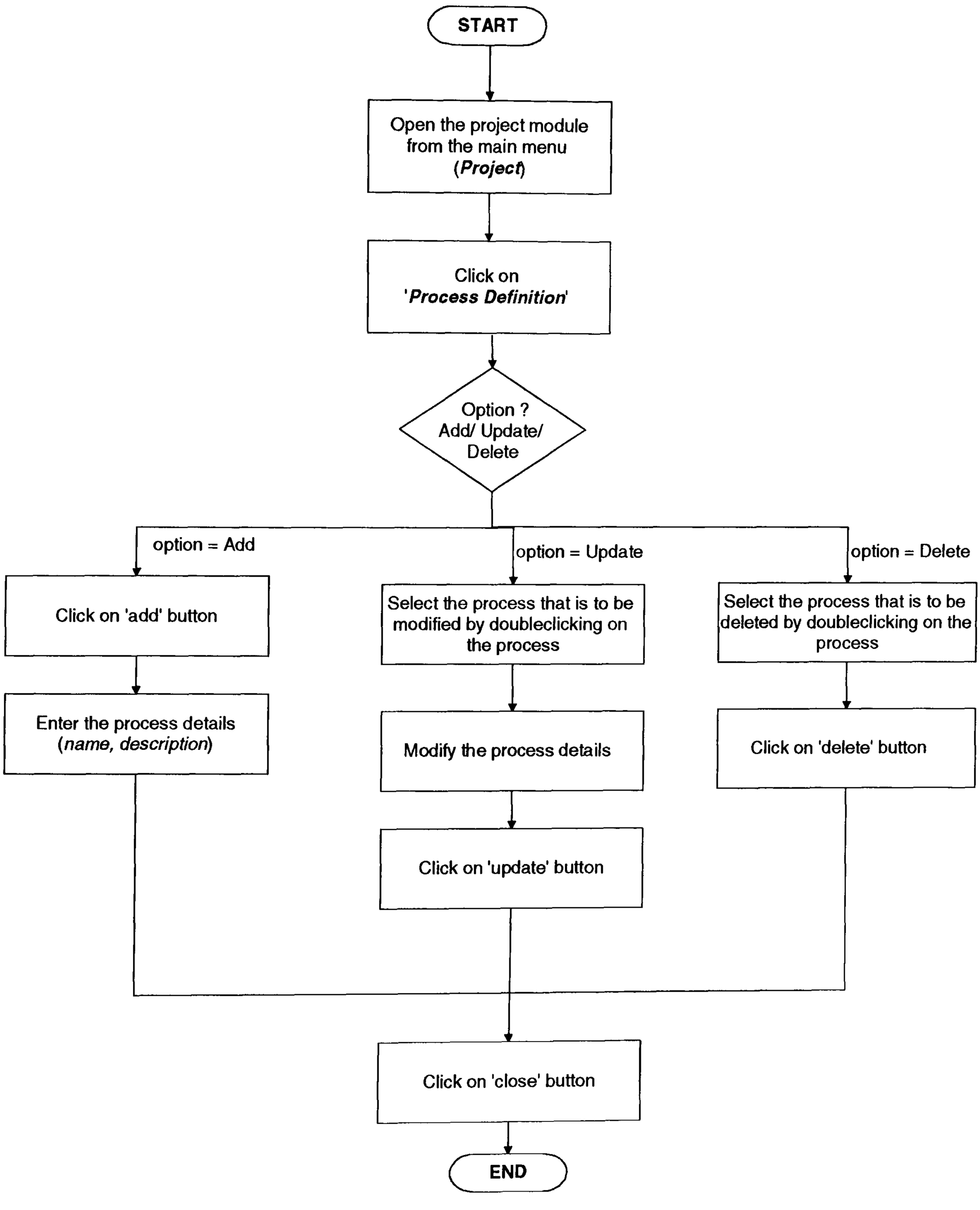


Figure D.29 Flowchart for process data maintenance

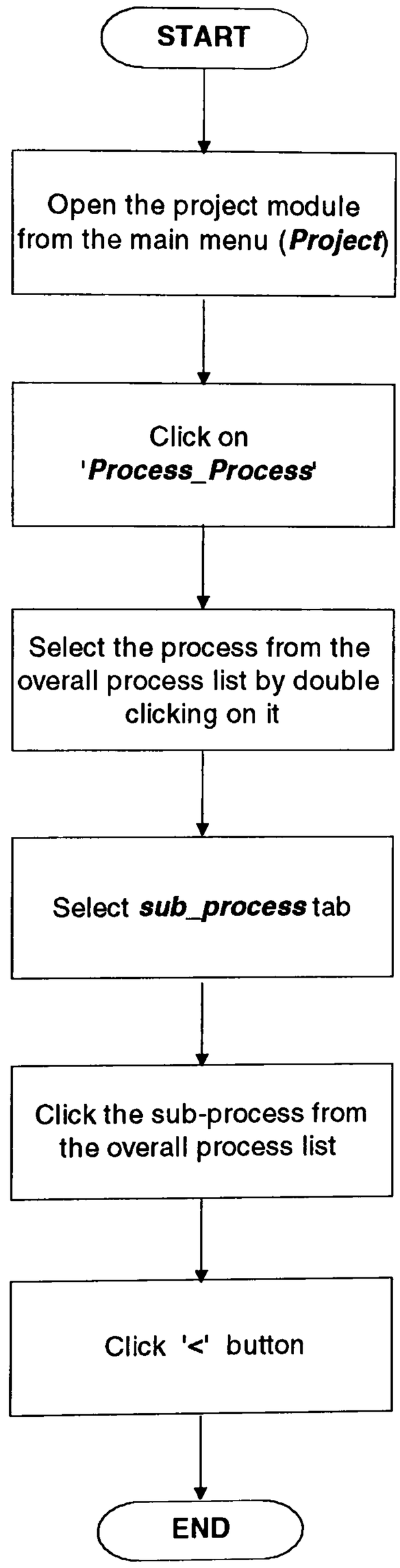


Figure D.30 Flowchart for maintenance of associations among processes

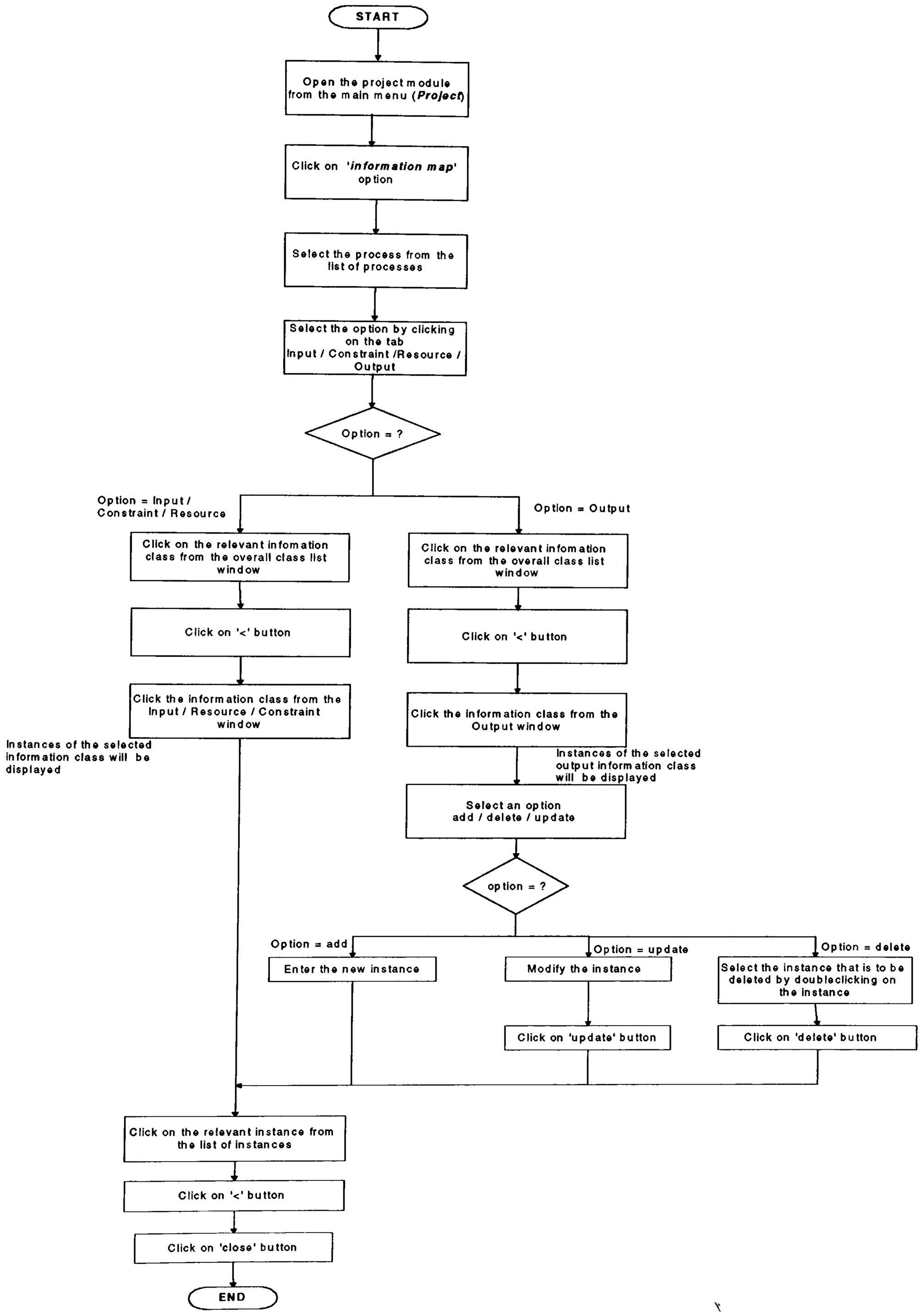


Figure D.31 Flowchart for maintenance of associations between process and information

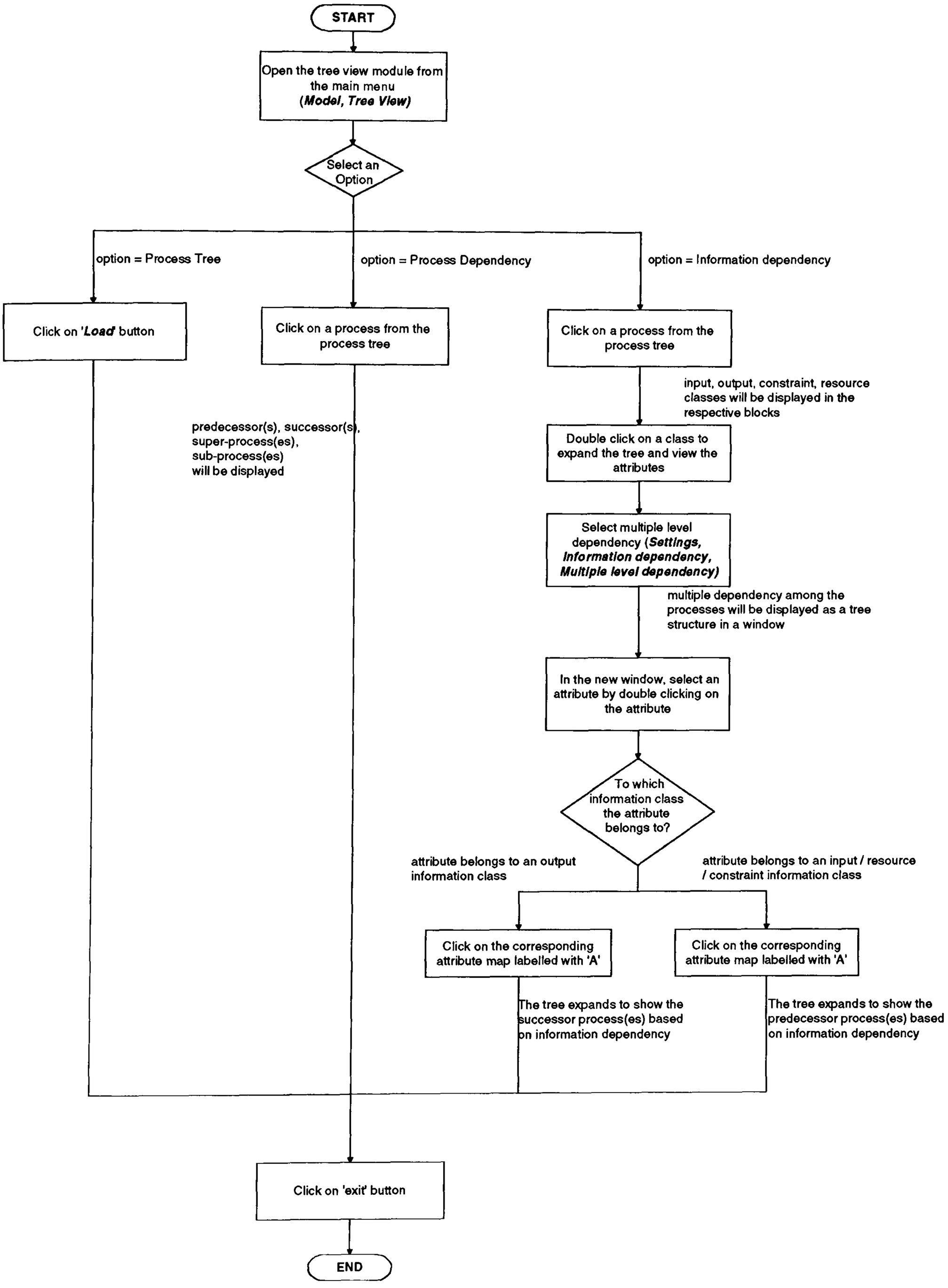


Figure D.32 Flowchart for viewing process dependencies based on associated information

APPENDIX E

REPRESENTATION OF PRODUCT INFORMATION

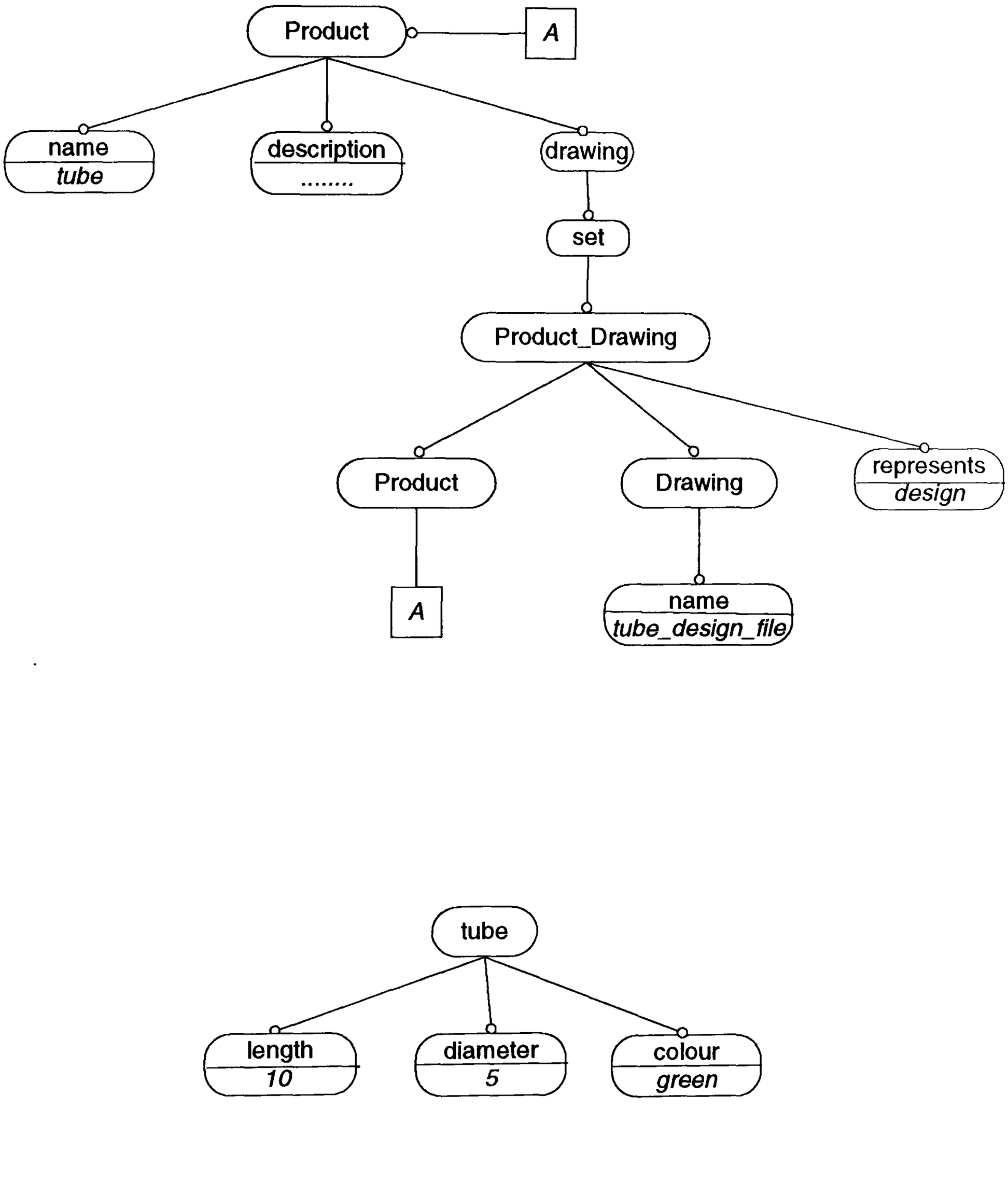


Figure E.1 A representation of the drawing information of the product

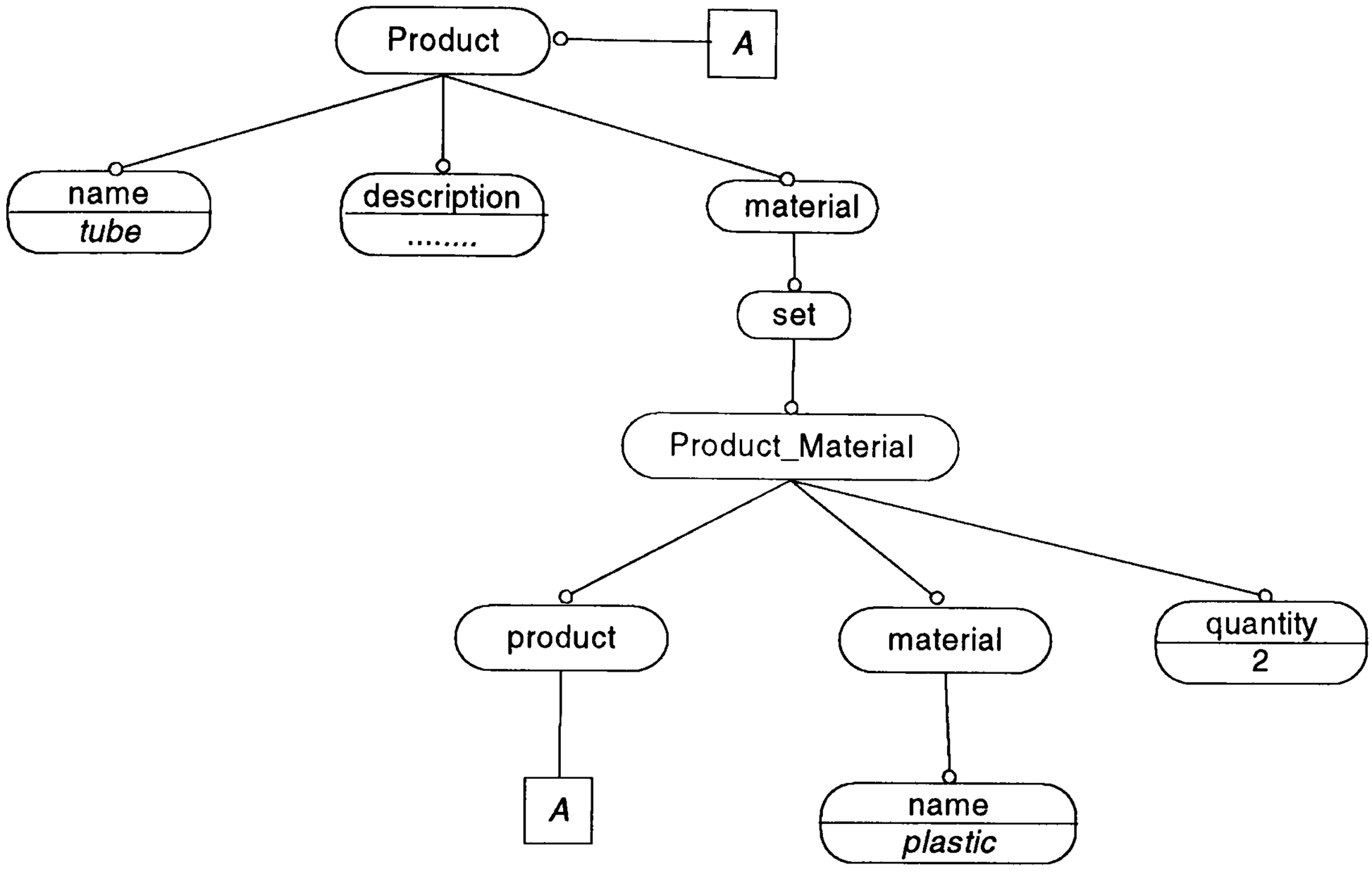


Figure E.2 A representation of the information on the material of the product

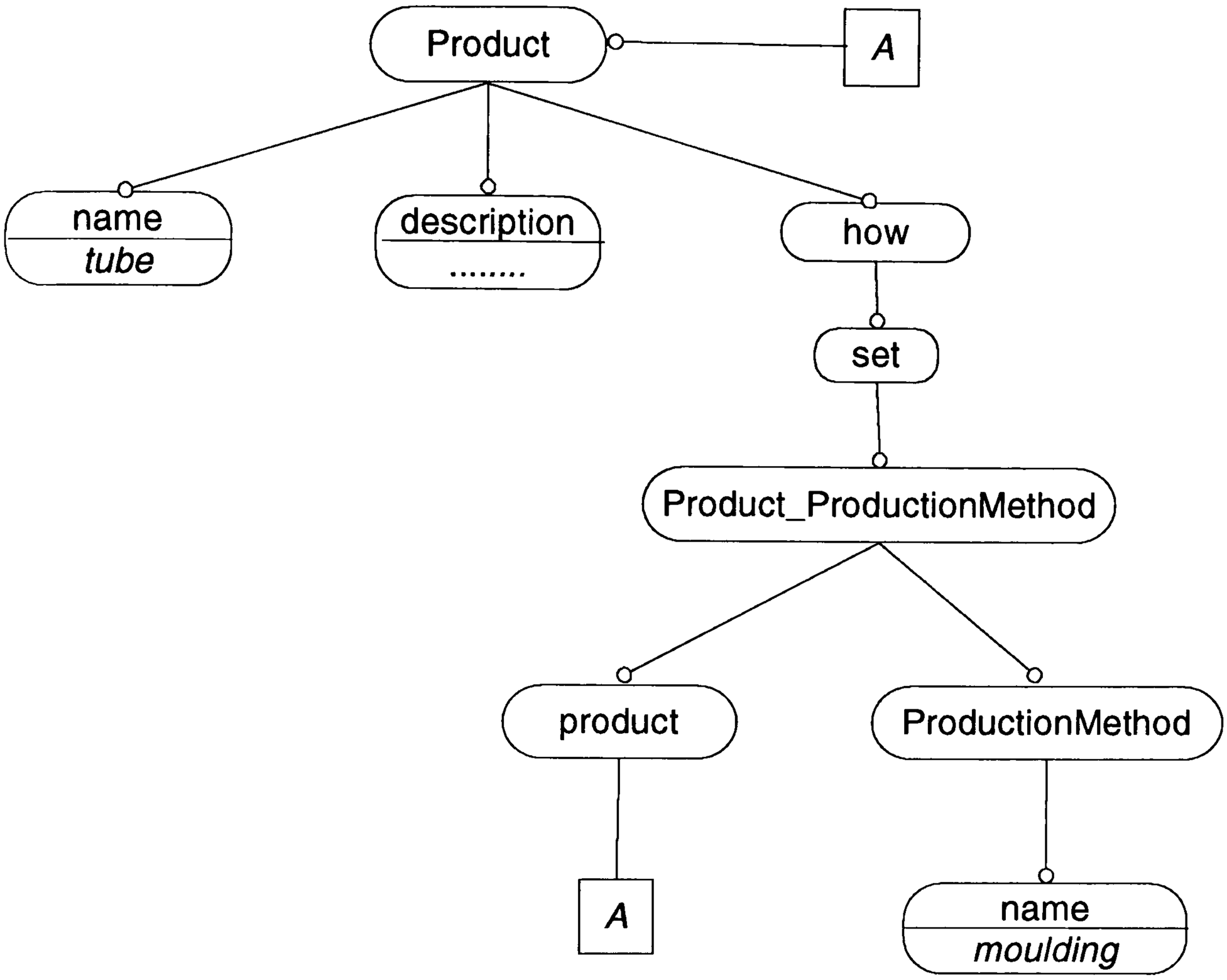


Figure E.3 A representation of the information on the production method of the product

APPENDIX F

EXPRESS-G

F.1 INTRODUCTION

This appendix given an informal explanation of a subset of EXPRESS-G thus enabling the reader to understand the EXPRESS-G diagrams in the thesis.

F.2 DESCRIPTION

EXPRESS-G is the graphical version of the EXPRESS data specification language. EXPRESS can be used to represent conceptual or real-world physical objects. EXPRESS models consist of a set of interrelated entity types, with each entity representing a collection of conceptual or real-world physical objects which have characteristics defined in terms of attributes. An attribute can be one of: an entity type, an atom type, a defined type, a selected type, an enumeration type or a set thereof. Atom types include integer, boolean, string, etc; a defined type is the user-defined type that is defined in the type definition section of the schema; and in case of enumeration type, for example, the status_of_a_project would be an enumeration of 'not yet started', 'started', 'completed'. EXPRESS-G represents these types as shown in Figure F.1. Attribute lines end in the circle at the attribute type. Attributes are identified by some text adjoining the attribute line which may have a suffix indicating the aggregation of a type and the limits on the number of elements in it. For example a list containing between 4 and 9 elements would be represented by L[4:9].

Some of the EXPRESS-G models in this thesis get a little complex. It would be confusing to have the attribute lines crossing each other so connectors are used as long as the connectors point to the right attribute.

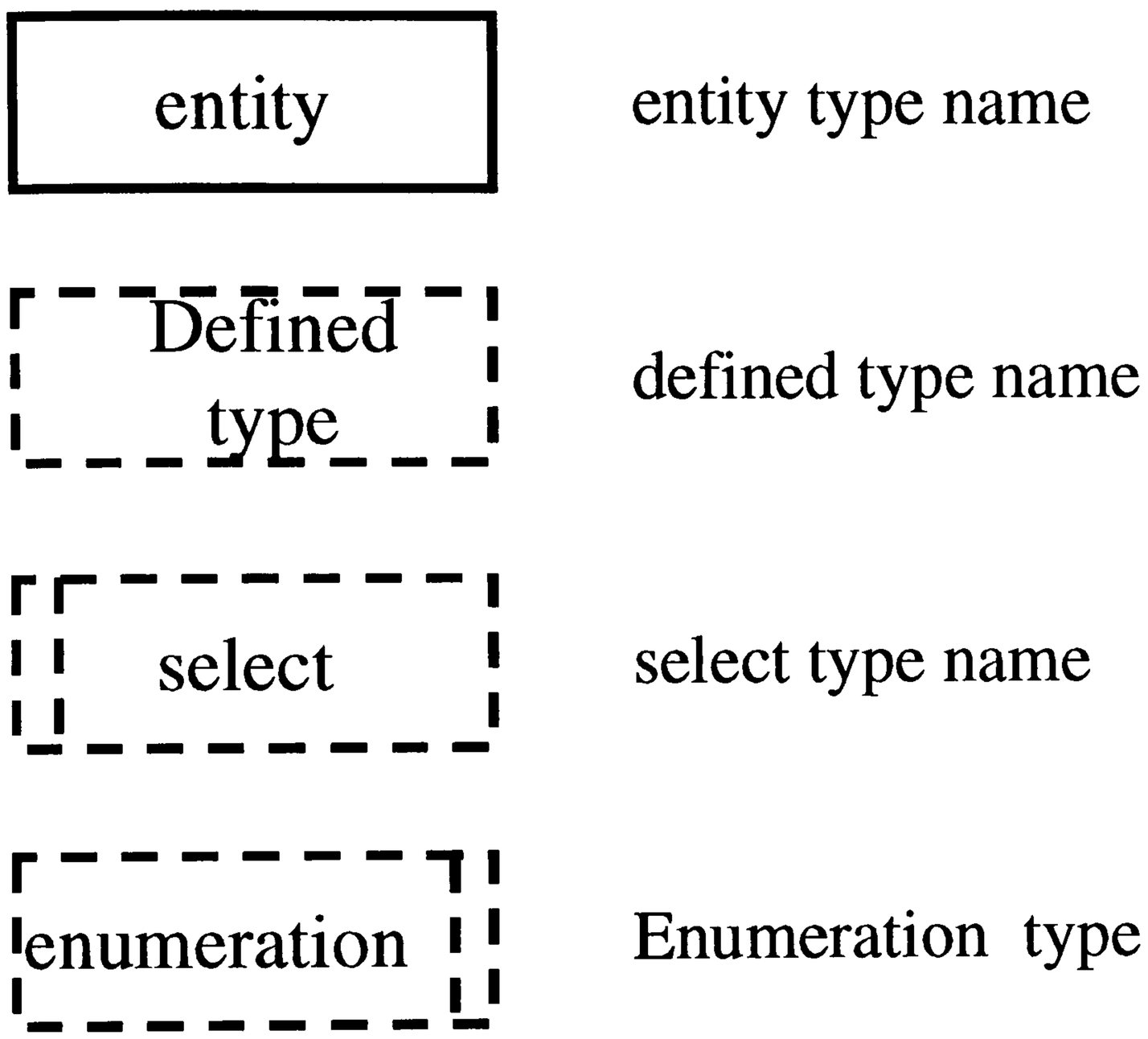


Figure F.1 Representations in EXPRESS-G

APPENDIX G

UniSQL

G.1 INTRODUCTION

UniSQL/M, a multidatabase management system can be used to develop the global database. It allows the users to access the data stored in other databases. UniSQL/X can be used to develop the local, object-relational databases. It is a unified database management system that extends the principles of relational database systems with features from object-oriented programming. This appendix gives details on registering local databases, virtual classes, special issues in authorisation on the global database entities and on access scope of the global database.

G.2 REGISTERING LOCAL DATABASES

The syntax and examples for registering a ORACLE relational database and a UniSQL/X database are shown in Figures G.1, G.2 and G.3 respectively. The databases belonging to the following database management systems - ORACLE (UNIX / VMS / WINDOWS-NT), INGRES, SYBASE, UniSQL/X, Vax Rdb/VMS, INFORMIX, IBI's Enterprise Data Access (EDA) and Microsoft SQL server for Windows NT can be registered with UniSQL/M for integration.

REGISTER LDB	*ldb_name*
NAME	*db_name_at_host*
TYPE	*db_type*
HOST	*host_name*
[USER	*user_name*
PASSWORD	*password*
DIRECTORY	*alternate_driver_directory*
OBEJCT_ID	{INTRINSIC | USER DEFINED }] [;]

Figure G.1 Syntax for registering a local database (UniSQL/M 1996)

```
REGISTER LDB   rProcDB
        NAME            'ProcDB'
        TYPE            'oracle'
        HOST            'fha219'
        USER            'Alan_L'
        PASSWORD        'Mount*('
        DIRECTORY       'd:\unisqlx'
```

Figure G.2 An example for registering an ORACLE relational database

```
REGISTER LDB    orProdDB
        NAME            'ProdDB'
        TYPE            'unisqlx'
        HOST            'fha212'
        USER            ' Olive_J'
        PASSWORD        'Flag97'
```

Figure G.3 An example for registering an UniSQL/X object-relational database

G.3 VIRTUAL CLASSES

A virtual class can unify information from local databases by naming other proxies or virtual classes in its query specifications. If classes make up part of the global database schema, virtual classes can access real data stored in those entities as well. Virtual classes can be defined with attributes having domains specified as proxies, classes, or other virtual classes in the global database. This composition hierarchy concept extends the relational DBMS data extraction capabilities and enables the global database to support complex data types (which are supported by UniSQL/X). However, neither a proxy nor a class in the global database can contain an attribute whose domain is defined as a virtual class (UniSQL/M 1996).

G.3.1 Query specification in a virtual class

A virtual class may contain none, one, or several query specifications. Query specifications are numbered within the virtual class according to the order in which they were defined. To verify the number of a particular query specification, a request ***;SCHEMA vclass_name*** can be sent to SQL/M to view the schema of the virtual class in question. The results of one **SELECT** statement can be unioned, differenced, or intersected against the results of another **SELECT** statement using the *table_operators* **UNION, DIFFERENCE,** or **INTERSECTION** in the query specification. The attributes that are selected must match the attributes that are defined in the virtual class in both number and data type. The **NA** designator (representing 'No Attribute' for missing attribute(s)) can be given as an item in the *select_list* when there is no data that corresponds to the virtual class attribute. **NA** can also be specified to resolve certain schema conflicts that arise from accessing different local databases.

G.3.2 Updatability of a virtual class or virtual class attribute

The following conditions must be satisfied in each query specification if an updatable virtual class is to be created:

1. The **FROM** clause must refer to only one proxy, class, or virtual class specification, which must also be updatable. This allows **FROM(vclass_x, vclass_y)** as a valid expression, since the two virtual classes named in parentheses represent a single virtual class specification.
2. The keywords **DISTINCT** or **UNIQUE** cannot be included in the query specification.

3. The **GROUP BY.....HAVING** clause cannot be used in the query specification.
4. Aggregate functions are not allowed in the **SELECT** clause.
5. If the **UNION** table operator is given in a query specification, the **UNION ALL** syntax must be used and updatable entities must be given on both sides of the **UNION**. The entities referenced under this format can appear in only one **FROM** clause.

G.4 SPECIAL ISSUES IN AUTHORISATION ON GLOBAL DATABASE ENTITIES

The relationships that exist between proxies and virtual classes in the global database, and between proxies and entities in the local database create some special considerations, which are summarised here:

1. To **SELECT, UPDATE, DELETE** or **INSERT** data associated with an entity in the global database, the user-id registered with the local database must have the corresponding authorisations in the table or class in the local database to be accessed.
2. To perform **SELECT, UPDATE, DELETE**, and **INSERT** operations on a virtual class, the owner of the virtual class must have **SELECT** and any corresponding authorisation on every entity (which could be either a class, proxy, or another virtual class) named in the query specification of the virtual class. If the appropriate authorisations are not granted on all entities, the query or requested operation is rejected.

The owner of a virtual class can grant authorisation privileges on that virtual class to other users of the global database. These users do not need explicit privileges on the proxies referenced by the virtual class; they automatically assume the virtual class owner's privileges on the proxy when they are granted authorisation on the virtual class. The syntax and examples for granting or revoking user access authorisation on a class are given in Figure G.4. The privileges include either ALL [PRIVILEGES] or a list of privileges separated by comma; the possible privileges are ALTER, DELETE, EXECUTE, INDEX, INSERT, SELECT and UPDATE.

Syntax:

GRANT *privileges* **ON** *class_specification_comma_list* **TO** *user_name_comma_list*
[WITH GRANT OPTION]

REVOKE *privileges* **ON** *class_specification_comma_list* **FROM** *user_name_comma_list*

Examples:

GRANT SELECT, INSERT, UPDATE ON *project* TO *jones;*
GRANT ALL PRIVILEGES ON *blade, input_to_stage2* TO *smith, brown;*

REVOKE INSERT, UPDATE ON *project* FROM *smith;*
REVOKE SELECT ON *input_to_stage2* FROM *jones, brown;*

Figure G.4 Syntax and examples for granting and revoking user access authorisation

G.5 ACCESS SCOPE OF THE GLOBAL DATABASE

The access scope is a list of the local database names used in the translation of virtual class queries and statements. Initially, when the global database is accessed, the default access scope includes all of the local databases that are registered with the global

database, as well as the classes that are defined in the global database. Access to a specified list of local databases can be obtained with the USE statement. The example shown in Figure G.5 limits the current access scope to local database *orProdDB* to create a new class *Stage2*. An **EVALUATE ON LDB** statement passes a non-query SQL statement from the global database to a named, registered local database for execution. An alternate way of performing the example given in Figure G.5 using EVALUATE ON LDB statement is shown in Figure G.6. The limitations on EVALUATE ON LDB statement are those imposed by the registered local database.

```
USE   orProdDB;

CREATE CLASS Stage2
(           part_number  char(10),
            ..............
);
```

Figure G.5 An example of access scope

```
EVALUATE  ON LDB  orProdDB

'CREATE CLASS  Stage2
 (          part_number   char(10),
            ..............
 );
```

Figure G.6 An example for EVALUATE statement

G.6 CHANGING DATA FROM THE GLOBAL DATABASE

Data in local databases can be changed through virtual classes defined in the global database. The INSERT, UPDATE, and DELETE statements can be issued against proxies and virtual classes to change data in the local database entities and global database classes that they access (UniSQL/M 1996). Figure G.7 shows an example that

inserts new data into the virtual class *vblade*. This new instance is targeted for the UniSQL/X local database *ProdDB*, so *orProdDB* is specified with the USE statement.

```
USE orProdDB;

Insert into vblade(part_number, , )

Values ('P001', , );
```

Figure G.7 An example for changing local database data from global database