International Journal of Production Research

Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713696255

An enterprise modelling CASE tool and data schema requirements for the selection of software support

K. T. K. Toh; J. A. Harding

To cite this Article Toh, K. T. K. and Harding, J. A.(1999) 'An enterprise modelling CASE tool and data schema requirements for the selection of software support', International Journal of Production Research, 37: 18, 4079 — 4104
To link to this Article: DOI: 10.1080/002075499189673
URL: http://dx.doi.org/10.1080/002075499189673

An enterprise modelling CASE tool and data schema requirements for the selection of software support

K. T. K. TOH† and J. A. HARDING†*

This paper describes a prototype CASE tool which has been implemented to support the selection of software applications. It is argued that information structures are invariably linked to the operation of the enterprise and can, therefore, be used as a powerful basis both for the suitability assessment and selection of candidate software applications. A structured methodology for the modelling of the enterprise operation and development of information requirements is first outlined to establish the application domain of the CASE tool. The functionality of the CASE tool is subsequently described, showing how aspects of the enterprise are captured in terms of organization, functionality, resource and information. It will be demonstrated how the information structures, captured by the CASE tool, are subsequently developed into a schema and used for the evaluation of a suitable software application.

1. Introduction

Advances in technological solutions coupled with the volatility of economic and social circumstances have encouraged enterprises to employ a multiplicity of commercially viable computer-based support systems to sustain their necessary business and manufacturing operations in an efficient and integrated manner. The reduction in processing time, physical storage and handling of paperwork, and easy access to information are some of the many justifications for the introduction of computer-based information systems. Examples of this are when sales order processing or material planning or inventory software support, etc. are introduced into the company. Therefore, a major aspect of enterprise engineering involves the design of the supporting enterprise information system.

Information systems are increasingly heterogeneous in nature; indeed, it is very rare to find industries with a monolithic homogeneous computing base where all functions operate from the same platform. Indeed, additional requirements of emerging manufacturing paradigms, e.g. virtual manufacturing, require an organization's interoperability to, eventually, also span its customer's and supplier's network interfaces. Furthermore, it is difficult for any single software supplier to support and provide the expertize necessary to cater for the full suite of software modules, needed to accommodate the diverse and specialized functions which exist within an enterprise. These different functions could range from order entry, shop floor data collection to inventory control. There may also be the requirement for modules

Revision received May 1999.

† Computer Aided Engineering (CAE) Research Group, Department of Manufacturing Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK.

* To whom correspondence should be addressed. e-mail: J.A.Harding@lboro.ac.uk

of custom-built components for highly specialized needs, e.g. organization-specific user interfaces.

The combination of several systems which individually satisfy particular requirements may not provide the best overall solution. The focus on software acquisition has traditionally been on pre-defined user interface requirements, and the selection of software configurations in enterprises is not based on sufficient assessment of the company's requirements in terms of operation and information structures. This paper shows how enterprise modelling architectures and CASE tools can be used to capture enterprise requirements and provide a powerful means of specifying software requirements. The emphasis on the design of CIM systems in the 1980s led to the development of a number of modelling architectures for the design and execution of CIM systems. Examples of these architectures are GRAI, CIMOSA and the PERA architectures. The emphasis of these architectures was that consideration of organizational, information and functionality aspects was important in the design of integrated systems. As such, the majority of the design architectures for CIM was based on developing and using integrated models of functionality, information and organization.

The argument of this paper is that methodologies for enterprise modelling can be used to generate 'neutral' information specifications (schema). This is a powerful asset when sourcing software, as applications may be selected by finding the best fit between the software data structures and the schema specifications. This paper outlines a methodology, and describes the role and functionality of a supporting prototype CASE tool which is aimed at providing smaller industries with the capability of generating specifications, from which decisions relating to the choice of application support can be made. It is important to note that the CASE tool is based on the principles of an underlying enterprise modelling architecture; this will demonstrate how modelling architectures can be powerfully deployed, as opposed to being perceived to be of solely theoretical benefit.

2. Enterprise integration and the role of IT

The design and implementation of an information system is a protracted task which involves diverse areas of expertize throughout all stages of the project. The requirements of the individual company have to be considered, along with the various enabling technologies and the highly competitive range of contemporary products available. To ensure the implementation of a system which is appropriate, a methodology should be used where individual phases of the design to implementation process are precisely defined (Uppington and Bernus 1998, Murgatroyd *et al.* 1998). Methodologies include guidelines, techniques and procedures which the end-user can follow in order to carry out the project. This may involve the use of structured approaches, reference architectures, and their associated modelling formalisms and graphical tools (Doumeingts and Chen 1992).

Structured approaches cover all aspects of the project which is typically divided into phases, e.g. analysis, design, implementation and operation phases. The reason for using structured approaches, as opposed to ad hoc methods, is so that a complex project may be organized into small, well-defined activities. By specifying the sequence and interaction of these activities, project planning and control becomes more effective. Exemplars include the structured approaches of the GRAI (Chen *et al.* 1997), Purdue (Williams 1994, Li and Williams 1997) and SSADM (Ashworth and Goodland 1990) methodologies.

Reusable modelling constructs may be identified by using CASE tools in the modelling process (Brough 1992, Williams *et al.* 1993, Hashemipour *et al.* 1997). The process of requirements definition (or modelling) then transforms the available knowledge into more formal descriptions of the information network specifications which can be used for implementation. The basic idea behind CASE is to support each phase of the life cycle with a set of labour-saving tools. Some CASE tools directly assist the design and support of system development, and also provide management information, documentation and control of the project as it develops, ensuring consistency, completeness and conformance to standards. The major task, however, is to select a suitable architecture from which the CASE tool can be developed. The CIMOSA and ARIS architectures are readily supported by CASE tools as each has a declared goal of formality in their definition and description of all aspects of their architectures (AMICE 1991, Scheer and Kruse 1994).

Formal systems design methodologies, e.g. CIMOSA, Purdue, etc. typically address the higher level subsystem issues that are relevant to the design of single (monolithic) custom-built applications. Indeed, some methodologies do not take account of the issues of integrating independently designed components, and the maintenance of systems through long life cycles entails requirements changes and extensions to multiple systems. This is gradually changing with the emergence of standards, e.g. the RM-ODP (ISO/OSI 1995), CORBA (Thomas *et al.* 1995) and DCOM (Li and Economopoulos 1997) for the development and deployment of applications in distributed heterogeneous environments. The impact of these IT infrastructure technologies on the majority of smaller industries is still unclear, especially their impact on the integration of existing software applications.

Very few organizations currently operate with monolithic custom-written software applications, and the trend is to opt for vendor or off-the-shelf solutions. Bespoke or custom systems are costly to implement, and the timescales required for realizing an operational system are frequently too long. Furthermore, it is unlikely that bespoke applications will match both the functionality and flexibility of the available vendor solutions. Powerful, market-tested vendor solutions, generally, provide a limited capability to be customized to meet specific end-user requirements.

The growing availability of pre-existing software, which is both diverse and comprehensive, is displacing the need for custom software. Specialist software houses operate in niche markets and benefit from a wide customer base, and as a consequence, specialized, off-the-shelf applications are developed with considerable expertize and knowledge of customer requirements. The end-user also benefits from software upgrades and peer support, e.g. technology transfer via user groups. Therefore, the task of systems design and implementation very often focuses on the integration of existing and new bought-in solutions.

The following sections describe a methodology and CASE tool which supports the generation of company information requirements. These form a basis for the selection of appropriate software applications to provide the best support to enterprise operations and the best satisfaction of enterprise information requirements.

3. Methodology for software specification based on the SSADM structure

The task of realizing an information or IT system, from specification to implementation, is well supported by existing structured approaches. The SSADM methodology (Ashworth and Goodland 1990), leads step by step from an existing

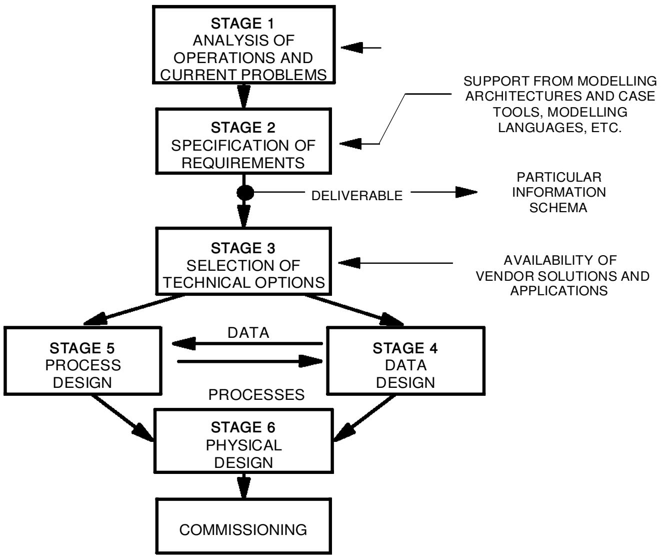

STAGE 1
ANALYSIS OF OPERATIONS AND CURRENT PROBLEMS
STAGE 2
SPECIFICATION OF REQUIREMENTS
SUPPORT FROM MODELLING ARCHITECTURES AND CASE TOOLS, MODELLING LANGUAGES, ETC.
DELIVERABLE
PARTICULAR INFORMATION SCHEMA
STAGE 3
SELECTION OF TECHNICAL OPTIONS
AVAILABILITY OF VENDOR SOLUTIONS AND APPLICATIONS
STAGE 5
PROCESS DESIGN
DATA
PROCESSES
STAGE 4
DATA DESIGN
STAGE 6
PHYSICAL DESIGN
COMMISSIONING

Figure 1. The structured approach of the SSADM methodology (Ashworth and Goodland 1990).

system to a future system taking into account evolution objectives and specific constraints, as illustrated in figure 1. Each stage consists of a set of tasks to be performed; the tasks are defined in terms of required inputs and outputs or deliverables. SSADM is accompanied by techniques which define how the individual stages of the structured approach are performed, these are supported by modelling formalisms, e.g. data flow diagrams, logical data structures, entity life histories and logical dialogue design.

Six stages of the SSADM realization process are fully supported by techniques and modelling formalisms, and these are highlighted in figure 1. These stages are user dominated and involve analysis of a current system and identification of the requirements. The result of this phase of activities is the requirements definition in terms of a set of specifications for the IT system. The subsequent stages of the SSADM methodology are vendor dominated and involve the activities which realize the IT system based on the specifications produced. Analysis of the organization in terms of its existing system is carried out to understand the operations and data requirements. This is an important aspect of the technique as it provides a firm basis for the design of the future system.

Specification of the requirements involves extracting the logical view of the system, from the understanding obtained during the previous stage, in terms of what the new system is supposed to achieve (Ashworth and Goodland 1990). This will involve aspects of both current operations and decisions about what must be included in the new system. The first two stages can be supported by modelling architectures, modelling languages and associated CASE tools. It is necessary to highlight the important role of modelling architectures and their associated CASE tools in structured approaches, e.g. SSADM. Further explanation of the different roles of modelling architectures in structured approaches is beyond the scope of this paper, but a detailed discussion may be found in Toh (1999a).

The selection of technical options involves the choice of the hardware and operating system. This will invariably require the careful consideration of various enabling technologies and configurations. These range, e.g. from the choice of computer network topology, physical transmission medium, communication protocol, and network access to the performance of the information network. The purchase of new systems and applications requires input from the IT vendors to compile the different implementation options. The final selection is based on the most appropriate solution which is economically viable.

Logical or conceptual data design involves formally documenting the information requirements for the system. SSADM is a data-driven method, which means that there is an assumption that systems have an underlying unchanging data structure, although processing requirements may vary (Ashworth and Goodland 1990). Within SSADM, this underlying data structure is modelled at an early stage of the methodology and forms a major part of the systems design process.

Logical process design involves specifying the processing requirements of the system or operations that the system will perform in response to events, data enquiries or updates. This includes a description of the data content of every input and output from the system, which will form the basis for the detailed design of graphical display formats, reports and form layouts; these specifications are subsequently given to application programmers who create the user interfaces. Physical design involves converting the conceptual specifications for data and the processing requirements into a design which will run on the target environment (Ashworth and Goodland 1990). At the end of this phase, the documentation required for the subsequent construction phases is produced.

One disadvantage of these top-down approaches is that they are based on deriving specifications for the design of a single (monolithic) system where the integration issues are addressed during the physical design stages. It should be noted from the figure that one of the major deliverables of the specification stage is the information schema specification which, in this paper, is not used as a specification for the realization of a bespoke system. The emphasis, however, is on the use of specifications as a basis for the selection of vendor-sourced software applications.

4. Proposed application for modelling the enterprise and generating the information schema

Invariably, a major decision to implement a new information system (either paper or computer based) presents a new opportunity to evaluate existing company business processes and initiate improvement measures either through the introduction of new business processes or the improvement of existing ones. The detailed analysis of the functional structures of the organization will invariably result in the growth of the supporting data structures, or changes to ones already in existence in the information system.

The major challenge is to ensure that the changes made to the underpinning information structure of the enterprise are reflected and documented. A major feature of enterprise modelling architectures, e.g. CIMOSA and ARIS is that they contain both function and information views to ensure that the data schemas are not developed in isolation of the enterprise functionality. These architectures are supported by CASE tools which provide an identification of those reusable modelling constructs already established in the system (Brough 1992, Williams *et al.* 1993). The following sections outline the application of a CASE tool, which is an extension

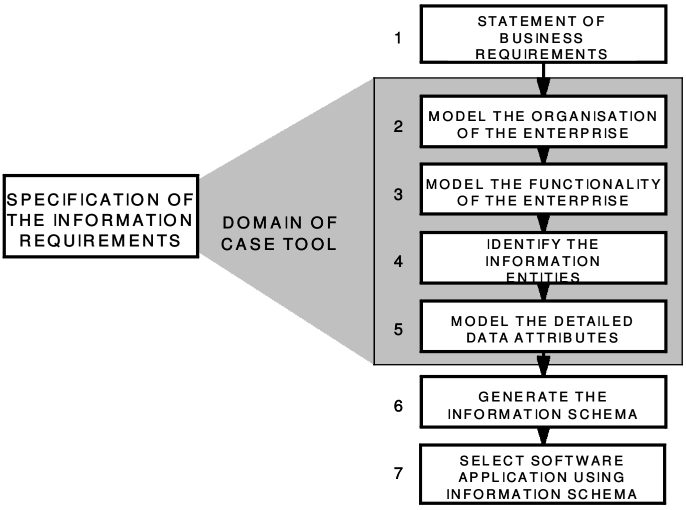

1
STATEMENT OF BUSINESS REQUIREMENTS
2
MODEL THE ORGANISATION OF THE ENTERPRISE
3
MODEL THE FUNCTIONALITY OF THE ENTERPRISE
4
IDENTIFY THE INFORMATION ENTITIES
5
MODEL THE DETAILED DATA ATTRIBUTES
6
GENERATE THE INFORMATION SCHEMA
7
SELECT SOFTWARE APPLICATION USING INFORMATION SCHEMA
SPECIFICATION OF THE INFORMATION REQUIREMENTS
DOMAIN OF CASE TOOL

Figure 2. Application domain of the CASE tool: modelling of the information requirements.

of the CIMOSA modelling language and which allows the information schema of an enterprise to be developed to support the task of software selection. (The CIMOSA modelling methodology was adapted to allow for the modelling of small human-dominated industries.) The reason for using this modelling approach is that the information structures captured by the CASE tool are inextricably linked to the functional structures of the enterprise and therefore, the schemas closely reflect the information requirements of the enterprise. Off-the-shelf software applications are selected on the basis that their underlying data schemas offer the most appropriate match to the enterprise data schema. Further development of the arguments for the development of modelling architectures may be found in Toh (1999a, b).

The application domain for the CASE tool is shown in figure 2, which shows a number of detailed steps leading to the specification of information requirements, expressed in the form of the information schema. The CASE tool is a realization of the fundamental CIMOSA concepts, where the enterprise functionality is modelled in terms of the operation of business processes; additional functionality has been added to the CASE tool to capture interactions between employees during the course of their work. This approach, which is an addition to solely modelling the business process, provides a means of identifying enterprise activities which are performed as a consequence of human-dominated interactions in enterprises. This enhances the applicability of the CIMOSA modelling language as it provides additional constructs for capturing enterprises which are dominated by employee collaboration.

The process begins with a high-level statement of the business requirements for the information system to be installed. The second step in the process is to establish the core business processes in the enterprise, this can be readily captured by using, e.g. the existing CIMOSA constructs, and using domain processes, business processes and enterprise activities. However, in this paper, an alternative approach is used which is centred on the modelling of functional entities derived from Koestler's (1967) concept of holons, explained in section 5.1.

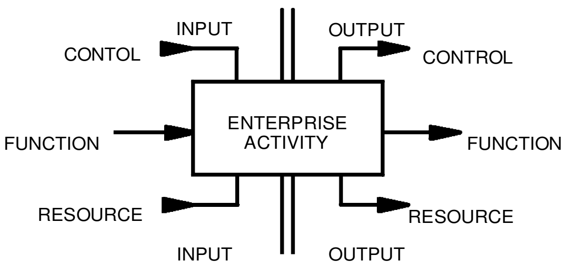


Figure 3. The modelling constructs for activity analysis (AMICE 1991).

The third stage involves the modelling of the enterprise functionality using holonic concepts described in section 5.2. Enterprise activities are captured using the language constructs for activity analysis shown in figure 3. The constructs are, by design, linked to the information view of the architecture through constructs, e.g. enterprise objects (sales orders, purchase orders, cutting tools, etc.) and corresponding object views (an object view is the collection of information entities associated to an enterprise object) which are modelled in conjunction with activity analysis. The identification of enterprise objects and object views becomes the starting point for the further analysis and development of information schemas.

The modelling of object views, in the fourth stage, results in the identification of information elements associated with individual enterprise objects. For example, a works order is the object view of a component part, which is an information entity. The information entities consist of a cluster of information elements associated with the component, e.g. start date for manufacture, due-date for delivery, etc. These information attributes are captured in the fifth stage.

The sixth stage involves the generation of the information schema which is compliant to the conceptual schema of the ANSI/X3 SPARC DBMS three-schema architecture. This involves identifying the relationships between the information entities, and is an important step as the information schema represents a neutral representation of the enterprise information structure independent of the implementation technology, e.g. databases, operating systems, distribution, etc.

During the sixth stage of the methodology, the enterprise information schema will be used as the basis for the selection of the most appropriate application software. The major reason for this focus is that an enterprise operates around an underlying, unchanging data structure and the conceptual information schema, therefore, provides a consistent, common and unambiguous view which can be referenced by the users of the IT system.

The authors believe that this is a powerful approach, as software applications can be shown to also have a corresponding information structure (section 6), which can be projected against the enterprise information schema specifications and the functional structures of the enterprise in order to identify the degree of compatibility. Methods for documenting the information schema against the software application will also be discussed in section 6. Other factors have to be taken into account in the ultimate choice of software, e.g. cost, user interface, security, etc. but it should be noted that this paper proposes an approach which is information centred, and that the focus of this research is to seek compatibility in terms of data structures as one basis for software selection.

5. Structure of the CASE tool

The design of the CASE tool has been heavily influenced by the architectural concepts of the CIMOSA architecture, where viewpoints provide the means to populate different aspects of the same unified model rather than dealing with the complexity of the whole model at the outset. This allows the modelling of the functional and behavioural aspects of the company using constructs belonging to one architectural view and subsequently proceeding to modelling the information structures using constructs in the information view. The advantage is that the modelling of the enterprise using one view of the architecture provides information which supports the analysis and population of other related views.

The CASE tool supports the modelling of function, information and organization views, whilst the modelling of enterprise functionality and behaviour is based on constructs developed from holonic concepts. The underlying schema of the CASE tool, based on the holonic constructs, will first be outlined using Booch class diagrams (Booch 1994). The holonic concepts have been described using the object-oriented model because the use of classes and inheritance provides a simple and expressive model for the definition of relationship in various parts of the system. The resulting model closely parallels the application domain, thus assisting in the design and understanding of a complex system (Monsef and Teggar 1998). The user interfaces of the prototype CASE tool have been implemented and tested in an industrial study. This will be explained in detail to illustrate the functionality of the CASE tool and demonstrate how data from a company can be captured using the modelling constructs.

The CASE tool is a windows-based application which has been implemented using the Microsoft development system, Visual C++ (Microsoft 1997). The development system is object oriented, where the visual user-interface objects, e.g. windows, dialogue boxes, dialogue controls and menus are encapsulated as object classes and provided in the Microsoft Foundation Class Library (Microsoft 1997). The object classes provide the functionality common to most applications written for windows-based operating environments. It will be seen that the most commonly used object class is the dialogue class, which encapsulates the functionality of the Windows dialogue box and controls. This class is invariably used in conjunction with the control classes, e.g. buttons, list boxes, edit boxes and check boxes, etc. An important feature of the object classes derived from the Foundation Library is the support for object persistence, which is the ability to write or read an object to or from a persistent storage medium, e.g. a disk file. The basic idea is that an object should be able to write its current state, usually indicated by the value of its member variables, to persistent storage. Later, the object can be recreated by reading the object's state from the storage. This handles all the details of object pointers and circular references to objects that are used when the state of an object is stored. This allows a repository of models representing different companies to be created and stored.

The CASE tool is built around two principal categories of dialogue interfaces. The first category is designed around a list-box which displays a list of objects which have been created and stored. The second category is based on a modelling template which enables instances of classes (objects) to be created. Examples of these two interface categories are shown throughout the following sections.

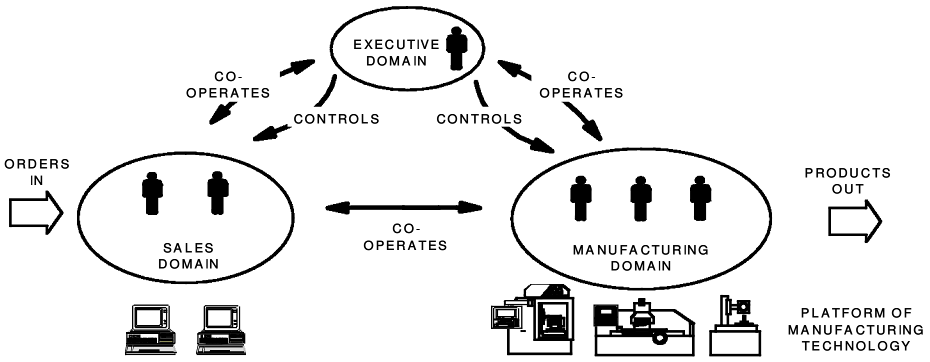
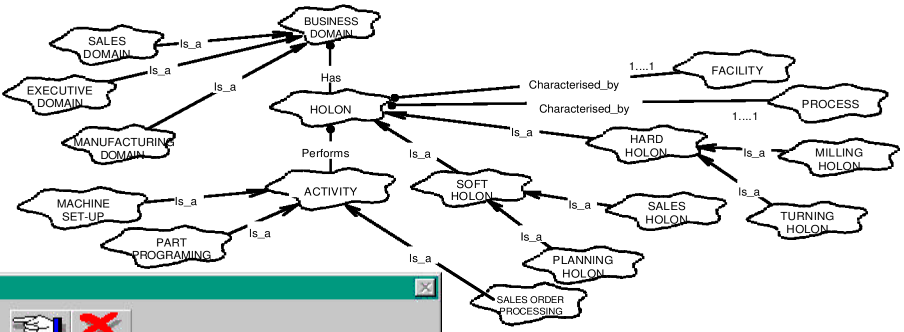


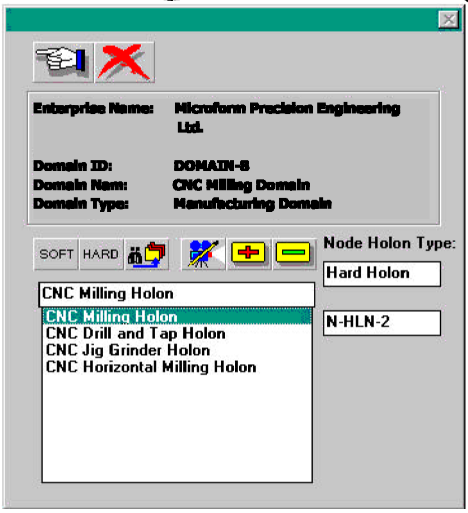
Figure 4. A model of the organization of a small company in terms of activity domains.

5.1. *Modelling the organization*

The CASE tool is based on the modelling of functional entities in the enterprise organized into functional entities, called holons, which exist within individual organizational groupings. It has been found that this is a more appropriate approach than relying solely on business processes for the modelling of small companies which are not highly structured (Toh *et al.* 1997). The CASE tool has been developed and the prototype has been tested for the modelling of small organizational structures. The model of a small company is depicted in figure 4, where the organizational structure is described in terms of activity domains. Three domains are indicated, these are: an executive domain; a business support domain; and a manufacturing domain. As a particular example, the executive domain can include activities, e.g. the planning and supervision of daily operations, and liaison with customers and suppliers. The business support domain may include the processing of orders in two parallel avenues. The first avenue relates to the financial aspects in terms of costing, invoicing, etc. The second avenue involves production planning, routing or scheduling, and the purchasing of material, etc. The manufacturing domain involves the implementation of production plans, monitoring the progress and ensuring that the production orders are completed on time. The representation of the business as interacting domains is not imposed on the company as a pre-ordained configuration, rather a particular representation is derived from a study of the business and its preferred view of the operation. This provides the starting point for the subsequent derivation of more appropriate information support requirements.

The concept of domains provides the association between top-down modelling and the complementary bottom-up modelling using holonic concepts, where an individual domain, shown in figure 5, acts as the starting point for the modelling of the individual company holons. The relation is such that an individual domain in a company *has* holons. (The semantic relations will now be shown in italics throughout the rest of the paper.) There is a hierarchy of domains, such that a business support domain *Is_a* domain. Furthermore, in accordance to Suda's (1990) definitions, holons which perform information-processing tasks are categorized as soft holons, and those which perform manufacturing tasks are categorized as hard holons. These are described using the class diagram such that a hard holon and a soft holon *is_a* holon.





Enterprise Name: **Microform Precision Engineering Ltd.**

Domain ID: **DOMAIN-8**
Domain Nam: **CNC Milling Domain**
Domain Type: **Manufacturing Domain**

Node Holon Type:
Hard Holon
N-HLN-2

CNC Milling Holon

CNC Milling Holon
CNC Drill and Tap Holon
CNC Jig Grinder Holon
CNC Horizontal Milling Holon

DIALOGUE CONTROL BUTTONS

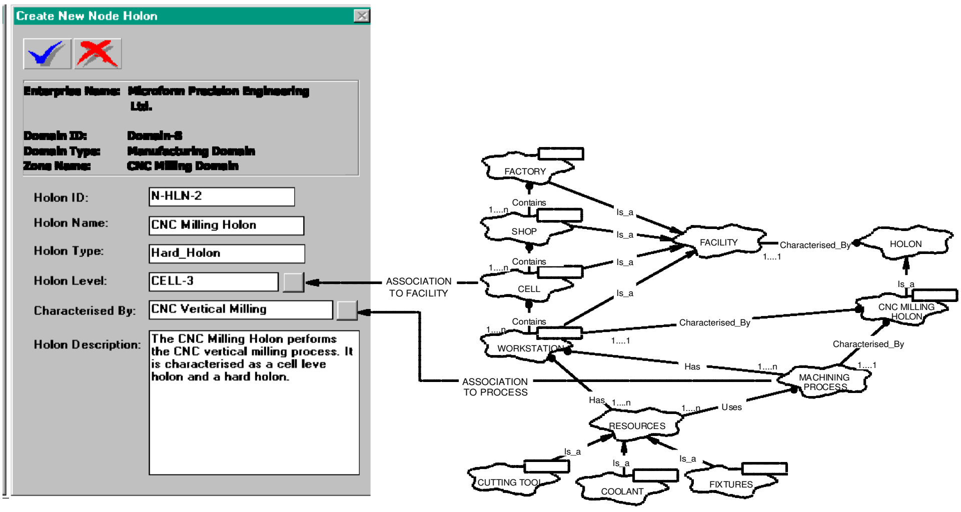
✗	CLOSE DIALOGUE AND CANCEL OPERATION
BUS	LIST THE BUSINESS SUPPORT ZONES
MFG	LIST THE MANUFACTURING ZONES
	SHOW OBJECT DETAILS
	EDIT OBJECT
＋	ADD OBJECT
−	DELETE OBJECT

Figure 5. The CASE tool dialogue which captures the organization of the business.

The interface which has been designed to model the organization structure is also shown in figure 5, it captures the holons (functional entities) in the CNC machining domain. The functionality is captured in the class diagram such that holon *performs* one or more $(1, \ldots, n)$ activities. Examples of the machining holons are the CNC jig grinder, CNC machinist, etc. In this instance, the machining (hard) holons are captured by the CASE tool, but it is possible to model information processing (soft) holons as there may be non-machining activities, e.g. operations planning, performed by employees within the CNC machining domain.

Holons are created using a modelling template shown in figure 6. The feature of this modelling template is the access it provides to process and facility browsers, which enable the process and facility aspects of the holon to be defined. Through this dialogue interface, a holon is characterized in terms of its principal functionality and by the facility level, e.g. factory, shop, cell or workstation level to which it can be associated.

In addition to characterizing the holon in terms of its principal functionality, the flexibility is built into the CASE tool for the holon to be *characterized_by* the facility level to which it is associated. An executive holon may be characterized as a factory level holon, or a machining holon as a workstation level holon, where the ability to associate any holon to a particular facility level enhances its description and gives more information about an entity which is captured by the CASE tool.



Create New Node Holon

Enterprise Name: Microform Precision Engineering Ltd.

Domain ID: Domain-8
Domain Type: Manufacturing Domain
Zone Name: CNC Milling Domain

Holon ID: N-HLN-2
Holon Name: CNC Milling Holon
Holon Type: Hard_Holon
Holon Level: CELL-3
Characterised By: CNC Vertical Milling
Holon Description: The CNC Milling Holon performs the CNC vertical milling process. It is characterised as a cell leve holon and a hard holon.

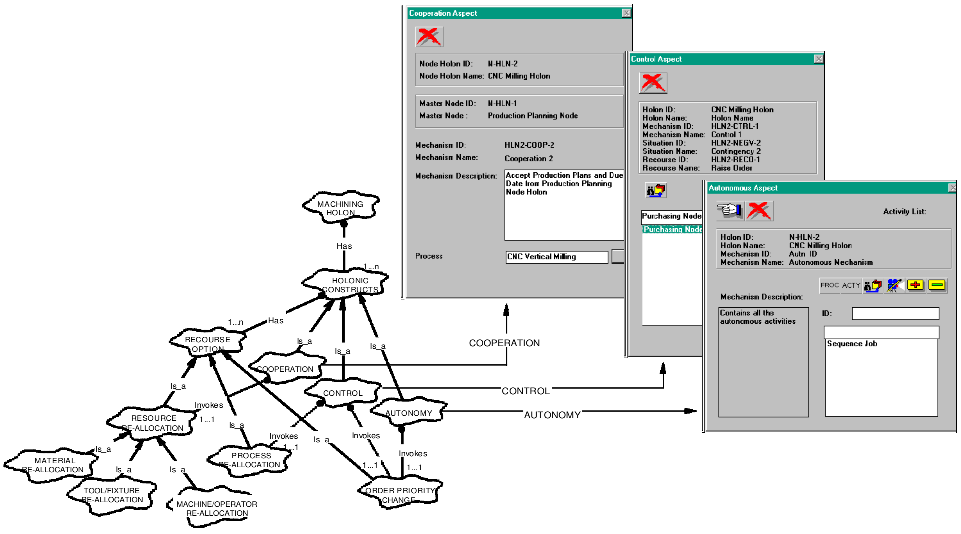
Figure 6. The CASE tool template for the modelling of holons.

Another association which gives the holon its identity is created by identifying the principal process which is performed by the holon. This is achieved by choosing the appropriate process classes from a process taxonomy which are discussed in Molina (1995). Furthermore, there is a relationship between the process and the resource where, e.g. the CNC milling process *uses* resources, e.g. cutting tools, material etc. Aspects of this are shown later in figure 9, which further describes how resource taxonomies can be created using the CASE tool. Other details which are required for the definition of a holon are a unique identification (ID), name and description. Once these parameters have been completely specified, an instance of a holon is created and included in the relevant domain. When a holon is defined, the CASE tool automatically progresses to the modelling of the enterprise functionality and behaviour which will subsequently lead to the modelling of information structures.

5.2. *Modelling the functionality and behaviour*

A major aspect of the CIMOSA architecture is the function view. To capture function and behaviour, CIMOSA models the enterprise in terms of a collection of concurrent business processes executed by a set of functional entities or resources (Kosanke 1992, Vernadat 1996). The language relies on formally defined domain processes, where enterprise events are used to identify the boundaries of a process (Kosanke 1995). The central and starting viewpoint for building a particular model in CIMOSA is the function view, its constructs capture in a top-down fashion the enterprise domain processes, business processes and enterprise activities (Kosanke *et al.* 1996).

The incorporation of holonic concepts in the CASE tool (figure 6) provides an alternative approach to modelling enterprise functionality which is centred on the holon as a functional entity. The holon is introduced as an abstract representation of both the human employee in the enterprise and the resource, e.g. machine tool, computer workstation, etc. The characteristics of the holon have been influenced by Koestler's fundamental definition of the holon and from



Cooperation Aspect

Node Holon ID: N-HLN-2
Node Holon Name: CNC Milling Holon

Master Node ID: N-HLN-1
Master Node : Production Planning Node

Mechanism ID: HLN2-COOP-2
Mechanism Name: Cooperation 2
Mechanism Description: Accept Production Plans and Due Date from Production Planning Node Holon

Process CNC Vertical Milling

Control Aspect

Holon ID: CNC Milling Holon
Holon Name: Holon Name
Mechanism ID: HLN2-CTRL-1
Mechanism Name: Control 1
Situation ID: HLN2-NEGV-2
Situation Name: Contingency 2
Recourse ID: HLN2-RECO-1
Recourse Name: Raise Order

Purchasing Node
Purchasing Node

Autonomous Aspect

Activity List:

Holon ID: N-HLN-2
Holon Name: CNC Milling Holon
Mechanism ID: Autn ID
Mechanism Name: Autonomous Mechanism

FROC ACTY

Mechanism Description: Contains all the autonomous activities

ID:

Sequence Job

Figure 7. Modelling the holonic aspects using the constructs for autonomy, cooperation and control in the function view.

Suda's application of the holon for highly automated manufacturing (Suda 1990). This provides a bottom-up modelling approach to capture the human interactions and facilitate the development of the information schema. Holons are characterized by their cooperation in an enterprise, where the holon is obliged to cooperate by performing its principal functionality, i.e. if the holon is identified as a machining holon, it cooperates by producing machined components. In turn, a contingency situation is immediately introduced, this is used to represent the circumstance where the holon is unable to fulfil its obligation. These ideas behind the holon have provided powerful concepts for developing a bottom-up modelling methodology, which can be used to capture interactions within and between domains.

The main thrust of modelling the enterprise behaviour in terms of employee interactions, is to identify the activities which are performed. The CASE tool therefore models a holon by providing constructs to capture its autonomy, cooperation and control aspects. The autonomous construct captures the self-assertive tendency of a holon, hence, it has the autonomy to perform a defined set of activities. The cooperative dimension captures the integrative tendency; the primary tendency of the holon is to cooperate, e.g. a turning holon cooperates by accepting jobs from the production planning holon and produces completed components.

The holon *has* autonomy, control and cooperation constructs, as illustrated in figure 7. The holon is also expected to have a degree of autonomy in performing supporting activities. The assignment of these activities is carried out during the modelling of an individual holon and provides an approach which is centred around the functional entity (employee and equipment) rather than in terms of business processes within an enterprise.

A third dimension of control is used, where at a particular instance, a holon may assert itself and impose constraints on another holon. When one peer holon controls another, a corresponding cooperation construct for the controlled holon to meet the demand must be created. When interactions occur between two holons, activities must be identified and analysed.

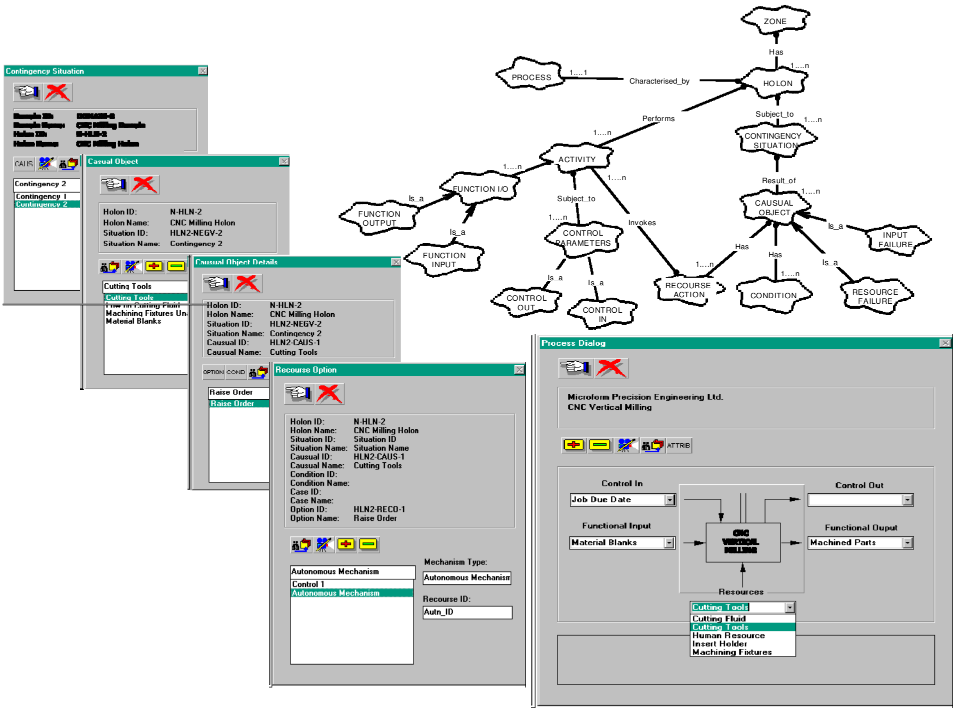
By registering both the cooperative and autonomous activities of holons during the modelling process, the 'portrait' of the holon is built up in the function view. Modelling the activities enables the modelling of the information requirements to be specified that support the activities, i.e. specification of the parameters in the information environment. In order to identify the primary functionality of the holon, the semantic relation *characterized_by* is used to associate the holon with a process which is defined in the manufacturing model. Hence, a holon with a vertical CNC machining centre would be *characterized_by* vertical CNC machining and *is_a* hard holon which performs vertical CNC machining tasks. The principal functionality of the holon is modelled using the relation *performs*, it therefore *performs* information processing or manufacturing activities. The holon also *performs* a set of defined supporting activities. The activities are, e.g. the setting up of a work piece or cutting tool, the purchase or acquisition of raw material, or the preparation of an NC part program.

Many of the interactions between holons occur, out of necessity, when contingencies are encountered. In order to further capture the holonic behaviour, any node holon is *subject_to* contingency situations. For each cooperation mechanism, an associated contingency is created, these occur when the obligations are not met, e.g. failure to fulfil the due date. A set of causal classes can be modelled with each contingency; a machining node holon requires cutting tools, raw material or machining fixtures, etc. as inputs to the process. A contingency can occur when the machining fixtures are absent, or there is a shortage of cutting tools or raw material.

Each causal class *has* one or a selection of recourse options. The node holon can *invoke* its autonomous mechanism, or summon (control) other nodes into taking recourse actions. In addition, the machining node holon may *invoke* its control mechanism over the production planning node holon to purchase new cutting tools. The production planning node is obliged to cooperate and a new activity is generated which is to purchase new cutting tools. Furthermore, the machining node holon *has* the autonomy to perform additional activities, e.g. generating an alternative job sequence or reallocating material from another job.

A number of recourse options, shown in figure 7, can be governed by some conditions which the enterprise may choose to impose. If the contingency was the *result_of* a shortage of cutting tools, a condition may be stipulated as the cost of the cutting tool. A number of scenarios can subsequently be specified which *determines* the choice of recourse option, e.g. if the cost of the cutting tool was less than a stipulated amount, the recourse would be to buy the cutting tool without the need for prior approval. Otherwise, the recourse option could be to *invoke* the control mechanism so that the production planner will raise a purchase requisition for the tool, or to compel a peer node holon to take over the machining task.

A holon is a functional entity which has the capability to interact, and by modelling these aspects of an enterprise, the operation of a set of activities is captured. Activities are modelled using a form of the IDEF0 formalism (Colquhoun *et al.* 1993, Jorgensen 1995), shown in figure 8, characterizing the inputs, outputs, controls and constructs (known as ICOMs). This has been selected because it is easily understood and has been widely applied for activity modelling. The extended IDEF0 notation for activity modelling is introduced in this research, with the addition of the 'control output' construct. This is necessary in order to model the information



Contingency Situation

Contingency 2
Contingency 1
Contingency 2

Casual Object

Holon ID: N-HLN-2
Holon Name: CNC Milling Holon
Situation ID: HLN2-NEGV-2
Situation Name: Contingency 2

Cutting Tools
Cutting Tools
Machining Fixtures Un
Material Blanks

Causual Object Details

Holon ID: N-HLN-2
Holon Name: CNC Milling Holon
Situation ID: HLN2-NEGV-2
Situation Name: Contingency 2
Causual ID: HLN2-CAUS-1
Causual Name: Cutting Tools

Raise Order
Raise Order

Recourse Option

Holon ID: N-HLN-2
Holon Name: CNC Milling Holon
Situation ID: Situation ID
Situation Name: Situation Name
Causual ID: HLN2-CAUS-1
Causual Name: Cutting Tools
Condition ID:
Condition Name:
Case ID:
Case Name:
Option ID: HLN2-RECO-1
Option Name: Raise Order

Autonomous Mechanism
Control 1
Autonomous Mechanism

Mechanism Type: Autonomous Mechanism
Recourse ID: Autn_ID

Process Dialog

Microform Precision Engineering Ltd.
CNC Vertical Milling

Control In: Job Due Date
Control Out:
Functional Input: Material Blanks
Functional Ouput: Machined Parts
Resources: Cutting Tools

Cutting Fluid
Cutting Tools
Human Resource
Insert Holder
Machining Fixtures

Figure 8. Modelling contingencies and recourse options.

parameters which are altered during the performance of an activity. A similar approach is also used by CIMOSA (AMICE 1991) for analysis of enterprise activities (which has a further resource output ICOM). It should be noted that only the 'resource input' has been used for activity analysis. Although usage of the resource output would enrich the description of the activity, the construct has not been used for the prototype CASE tool as it has been found that the resource input is adequate for the capture of the resource objects. The modelling of these resource objects during activity analysis will in turn allow the identification of associated information elements.

The objects which constitute the functional inputs and outputs for a machining activity are, e.g. the raw material, bought-in components, component drawings. The task of object identification involves the specification of enterprise objects as the inputs and outputs of the enterprise activities. The specification of the enterprise objects in terms of functional inputs and outputs, and resource objects are linked to the specification of information where the identification of the objects constitutes the first step in identifying entities for the information schema, where the enterprise objects are in turn composed of information elements manifest as object views (Jorysz and Vernadat 1990, AMICE 1991). This will be further discussed in section 5.4.

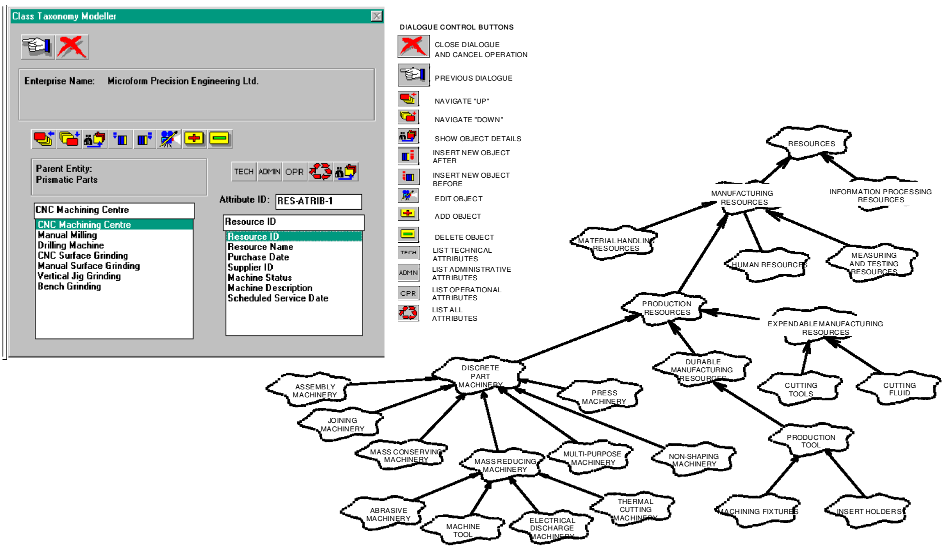
5.3. *Modelling of company resources*

The modelling of company resources is linked to the task of activity analysis, which is carried out using the extended IDEF0 notation shown in figure 8. The type of objects which constitute the resource inputs and outputs are determined by the activity being modelled. For a machining activity, the functional inputs are the raw material, bought-in components, component drawings, etc. The functional outputs are the completed components. In the case of information processing activities, the functional inputs may be sales order forms, purchase requisition forms, etc. The functional input and output of an activity are specified with modelling templates, and the addition of any new item will be reflected in the combo style list-box.

The resource inputs are the machine tool, the human resource (operators), machining fixtures, cutting tools, inspection gauges, etc. A prerequisite in specifying the resource inputs is that the source (or library) of objects have to be obtained from the resources identified within the facility. The modelling of resource structures can be considerably enhanced through the development of resource taxonomies; these have been used as a means to establish common glossaries for the successful communication of concepts and ideas amongst multi-disciplinary groups involved in the design of CIM systems (Camarinha-Matos *et al*. 1995, Perakath *et al*. 1995).

This gives a major advantage as it allows the generic structure of process and resource classes to be developed independently and reused. The resource taxonomy can be also defined using the generalization–specialization mechanism. The creation of taxonomies allows the different types of manufacturing resources to be organized consistently in hierarchies. The taxonomy can be continually enhanced by the addition of new classes of resources or processes, or the creation of greater levels of abstraction. Maintaining the independence between the taxonomies and facility description promotes the reusability of the generic resource structures.

A feature of the CASE tool is the class taxonomy modeller which enables the process and resource taxonomies to be created, the interface is illustrated in figure 9, where the taxonomy structures can be created independently of any specific instance of manufacturing facility. The taxonomy modeller consists of two list-boxes, the first displays a list of the resource classes which have been modelled. The second list box displays the attributes which are associated to the resource class. Control buttons


Class Taxonomy Modeller

Enterprise Name: Microform Precision Engineering Ltd.

Parent Entity:
Prismatic Parts

CNC Machining Centre

CNC Machining Centre
Manual Milling
Drilling Machine
CNC Surface Grinding
Manual Surface Grinding
Vertical Jig Grinding
Bench Grinding

Attribute ID: RES-ATRIB-1

Resource ID

Resource ID
Resource Name
Purchase Date
Supplier ID
Machine Status
Machine Description
Scheduled Service Date

Figure 9. The class taxonomy modeller which captures information classes.

have been built in to display the technical, administrative and operational attributes of the resource, respectively. An important feature of the taxonomy modeller is that any attribute which is added to a higher level class in the taxonomy is automatically inherited by the child class. Therefore, if the attribute 'process cost' is specified for a machining process, all derived classes, e.g. CNC machining processes and turning processes also inherit the attribute 'process cost'. In addition, specific attributes may also be added to an information class at any particular level.

The class taxonomy modeller includes the 'navigate-up' and 'navigate-down' controls which provide the means of viewing individual levels of the entire taxonomy. Dialogue controls are also provided to enable the navigation into lower or higher levels of the class structure. New information classes can be added to, or deleted from the list at any particular level of the taxonomy. The taxonomy modeller has to maintain the integrity of the taxonomy when any information class is removed or added at the intermediate levels of the structure, this enables the correct inheritance of information attributes to be maintained as the taxonomy evolves. There are two scenarios which have to be considered, the first is when an information class is deleted from the hierarchy. In this instance, the taxonomy tree of derived classes below the deleted class has to be moved onto the parent of the deleted class. The second scenario involves the addition of a class to the taxonomy, where in this instance, the taxonomy tree of derived classes has to inherit from a new parent class. The taxonomy structure is maintained as each information class is incorporated with a record of its parent classes, i.e. each information class has a record of its genealogy. Any modifications to the taxonomy will result in updates to the genealogy record of the classes affected by the changes.

5.4. *The specification of information elements*

The specification of the information elements and generation of the information schema corresponds to the information view of the CIMOSA architecture. The work is typically carried out in three stages, i.e. specification, design and implementation. The CASE tool supports the specification stage, where the results of the activity analysis are the identification of data parameters which constitute the elements of the extended IDEF0 control input, output and resource ICOMs.

In accordance with the CIMOSA modelling language, the support for information specification (in the information view) at the specification stage is centred around object identification. At the design stage, the relationships between the identified objects have to be modelled within the information view to produce the specifications for the conceptual information schema. Should decisions be influenced by the implementation-specific issues in the latter stages of the design phase, further data engineering techniques (e.g. relational analysis) may be applied to derive detailed internal and external schema specifications, in accordance with the ANSI/X3 SPARC DBMS (Tsichritzis and Klug 1972) three-schema architecture.

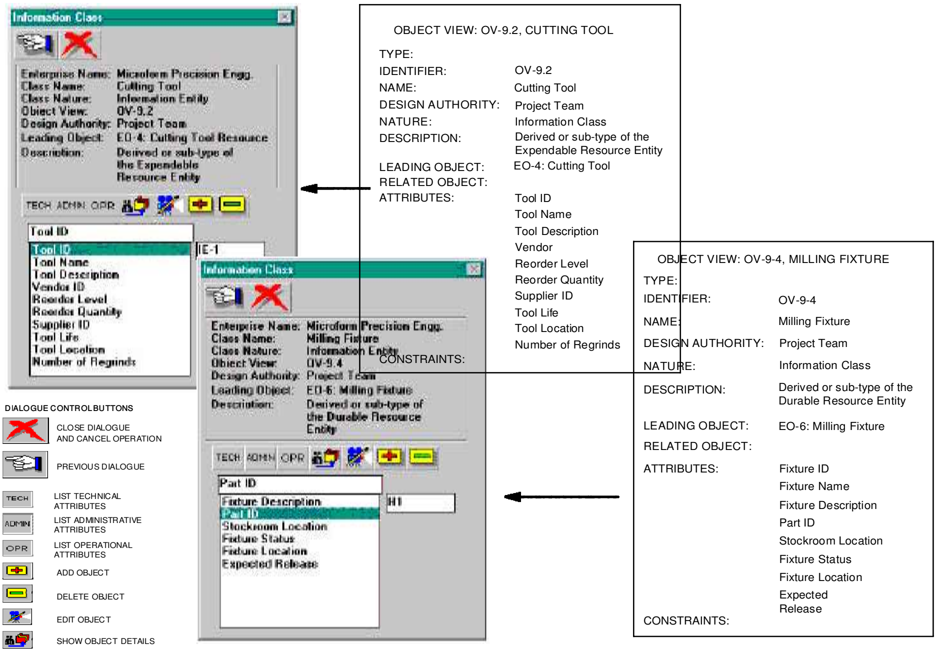
During functional analysis, e.g. a production planning holon will invoke the cooperation of a machining holon. Correspondingly, the machining holon cooperates with the production planning holon by producing the required components. The interaction between the two holons thus identifies a due date for the order, which is a data parameter taken from the information models, as a control input ICOM for the machining activity.

The link between the information elements and the extended IDEF0 representation of the activity is established through the control input and output ICOMs. The operation of an activity, depicted previously in figure 8, is controlled by, or alters the values of the information parameters associated to the enterprise objects. The information elements which are altered during the course of the activity are derived from the information schema.

At the specification level, the process of information system specification is related to functional analysis, where the modelling of enterprise activities leads to the identification of enterprise objects, and in turn, to object views and information elements (AMICE 1991). These elements might be, e.g. related to the products to be created, inventory levels or parameters related to the resources employed. The relationship between the function view and information view is that the performance of any activity or task results in an alteration to the attribute values in the information system environment. (It should be clarified that here an 'alteration to the attribute value' refers to the changing of the actual values rather than the type of attribute. For example, the attribute 'Month' belonging to a sales-order may change in value from 'August' to 'September'. The type of attribute (string) is not changed.)

The specification of information system requirements may be illustrated using the analysis of the shop floor machining activity. The activity analysis for the machining of components using the CASE constructs allows the modelling of the shop floor activities in order to specify the information system requirements. The function inputs are the raw materials, tooling, coolant, part drawings, etc. The identification of the resource entities as enterprise objects, e.g. cutting tools, fixtures, etc. is part of the analysis of the machining process. Information elements associated to the cutting tools are shown in figure 10.

The cutting tool entity is a sub-type of the expendable resource entity, the object view which contains all the associated information elements (attributes). The information elements are, e.g. the Tool ID (unique), tool name, description,


Information Class

Enterprise Name: Microform Precision Engg.
Class Name: Cutting Tool
Class Nature: Information Entity
Object View: OV-9.2
Design Authority: Project Team
Leading Object: EO-4: Cutting Tool Resource
Description: Derived or sub-type of the Expendable Resource Entity

TECH ADMN OPR

Tool ID — IE-1

Tool ID
Tool Name
Tool Description
Vendor ID
Reorder Level
Reorder Quantity
Supplier ID
Tool Life
Tool Location
Number of Regrinds

OBJECT VIEW: OV-9.2, CUTTING TOOL

TYPE:	
IDENTIFIER:	OV-9.2
NAME:	Cutting Tool
DESIGN AUTHORITY:	Project Team
NATURE:	Information Class
DESCRIPTION:	Derived or sub-type of the Expendable Resource Entity
LEADING OBJECT:	EO-4: Cutting Tool
RELATED OBJECT:	
ATTRIBUTES:	Tool ID Tool Name Tool Description Vendor Reorder Level Reorder Quantity Supplier ID Tool Life Tool Location Number of Regrinds
CONSTRAINTS:	

Information Class

Enterprise Name: Microform Precision Engg.
Class Name: Milling Fixture
Class Nature: Information Entity
Object View: OV-9.4
Design Authority: Project Team
Leading Object: EO-6: Milling Fixture
Description: Derived or sub-type of the Durable Resource Entity

TECH ADMN OPR

Part ID — H1

Fixture Description
Part ID
Stockroom Location
Fixture Status
Fixture Location
Expected Release

OBJECT VIEW: OV-9-4, MILLING FIXTURE

TYPE:	
IDENTIFIER:	OV-9-4
NAME:	Milling Fixture
DESIGN AUTHORITY:	Project Team
NATURE:	Information Class
DESCRIPTION:	Derived or sub-type of the Durable Resource Entity
LEADING OBJECT:	EO-6: Milling Fixture
RELATED OBJECT:	
ATTRIBUTES:	Fixture ID Fixture Name Fixture Description Part ID Stockroom Location Fixture Status Fixture Location Expected Release
CONSTRAINTS:	

DIALOGUE CONTROL BUTTONS

✗	CLOSE DIALOGUE AND CANCEL OPERATION
☞	PREVIOUS DIALOGUE
TECH	LIST TECHNICAL ATTRIBUTES
ADMIN	LIST ADMINISTRATIVE ATTRIBUTES
OPR	LIST OPERATIONAL ATTRIBUTES
➕	ADD OBJECT
➖	DELETE OBJECT
✎	EDIT OBJECT
🔍	SHOW OBJECT DETAILS

Figure 10. Identification of attributes associated to cutting tools and milling fixtures.

etc. Furthermore, there is also usage information, which will maintain the information relating to the level of inventory, tool life, tool location, etc. If the information schemas are realized in the form of a company database, data obtained from the shop floor can be maintained giving useful information on the level of tooling inventory, reorder dates and tool location, etc.

In a similar manner to the cutting tool example, information relating to milling fixtures can be specified in terms of the object view for the fixture resource. The fixtures are derived from the durable resource class. The attributes, e.g. fixture status and location provide the data fields for monitoring the usage of the machining fixtures on the shop floor. The specification of data schemas which accommodate such 'on-line' usage of information provides the company with the capability to monitor the status of their resources for the sequencing of work at machine tools. In modelling the machining process, it may be decided that information, e.g. the expected release date (completion time of the machined component) will constrain other downstream activities which also require the same machining fixture.

The data schemas are developed during the design stage using the enterprise objects and information elements identified at the specification stage. A number of information entities are shown in figure 11, using the EXPRESS modelling language (Schenck and Wilson 1994). The EXPRESS language has been used to complement development of the schema using entity-relationship diagrams. The number of information entities shown in the figure have been necessarily restricted due to constraints on the length of this paper, but these have been chosen to illustrate how the EXPRESS schema can be related directly to the CIMOSA enterprise objects and associated object views. The EXPRESS language has been found to provide an appropriate means for information modelling in accordance with the SPARC/ANSI/X3 DBMS architecture for conceptual schema specifications. The data attributes within each entity type can be clearly identified and documented in the schema diagram, and are consistent with the guidelines of the RM-ODP for information modelling. There is a further advantage where the EXPRESS schema can be parsed and checked to ensure the correct usage of the syntax.

6. Mapping of the software applications against the schema

Software applications invariably have an underlying information structure which can also be documented in the form of a conceptual schema. Off-the-shelf software have a higher chance of both being accepted and successfully applied in the enterprise if they are selected on the basis that their underlying data schemas have the potential to offer the most appropriate match to the enterprise data schema in terms of data usage. The development of the schema for the candidate software is carried out by documenting the data fields used during information entry and export from the software, shown in figure 12. The information elements and attributes for the development of the schema may be obtained from sources, e.g. data fields on graphical displays, reports and form layouts or data files from the software.

The conceptual schema of the software is captured using an appropriate modelling language, and in this case, IDEF1x (Bruce 1992) has been used. To illustrate this mapping, the underlying structure of a sales processing package has been chosen. The initial assessment of the product was that it provided a very powerful interface for the handling of the task of order entry and order book maintenance. It has been developed around two principal categories of information. The first relates to the maintenance of a structured product or parts record, carried out through the bill of

```
SCHEMA Enterprise_Information_Model;

ENTITY Sales_Order;
Sales_Order_No  : STRING;
Order_Date      : Date;
Delivery_Date   : Date;
Customer        : Customer;
Contact         : STRING;
Placed_By       : Employee;
Item_List       : LIST[0:?] OF Job;
Order_Quantity  : INTEGER;
Order_Desc      : STRING;
Status          : Order_Status;
END_ENTITY;

ENTITY Job;
Job_ID          : STRING;
Job_Name        : STRING;
Part_Desc       : STRING;
Drawing_ID      : STRING;
Order_Quantity  : INTEGER;
Proc_Plan_List  : LIST[0:?] OF
Process_Plan;
INVERSE
In_Sales_Order  : Sales_Order FOR
Item_List;
END_ENTITY;

ENTITY Process_Plan;
Plan_ID         : STRING;
Plan_Date       : Date;
Date_Modified   : Date;
Generated_By    : Employee;
Op_List         : LIST[0:?] OF
Machine_Operation;
Plan_Desc       : STRING;
INVERSE
In_Job          : Job FOR Proc_Plan_List;
END_ENTITY;

ENTITY Resource;
Resource_ID     : STRING;
Resource_Name   : STRING;
Resource_Desc   : STRING;
Supplier : Supplier;
Resource_Cost   : INTEGER;
Location: STRING;
INVERSE
In_Process      : Process FOR
Resource_Used;
In_Facility     : Facility FOR
Resource_List;
END_ENTITY;
```

```
ENTITY Supplier;
Sup_Name        : STRING;
Sup_Address     : STRING;
Contact         : STRING;
Tel_Number      : INTEGER;
Fax_Number      : INTEGER;
END_ENTITY;

ENTITY Process;
Process_ID      : STRING;
Process_Name    : STRING;
Process_Desc    : STRING;
Process_Cost    : INTEGER;
Resource_Used   : LIST[0:?] OF Resource;
END_ENTITY;

ENTITY Machine_Operation;
Operation_ID    : STRING;
Operation_Name  : STRING;
Operation_Desc  : STRING;
Workstation     : Facility;
Setup_Time      : INTEGER;
Operation_Time  : INTEGER;
INVERSE
In_Op_List      : Process_Plan FOR
Op_List;
END_ENTITY;

ENTITY Employee;
Employee_ID     : STRING;
Employee_Name   : STRING;
NI_Number       : INTEGER;
Address         : STRING;
Emp_Function    : STRING;
Date_Joined     : Date;
END_ENTITY;

ENTITY Purchase_Order;
PO_Number       : STRING;
PO_Date                   : Date;
Required_Date   : Date;
Supplier : Supplier;
Raised_By       : Employee;
Item_Desc       : STRING;
END_ENTITY;
```

Figure 11. A selection of entities from the EXPRESS schema for the enterprise information requirements.

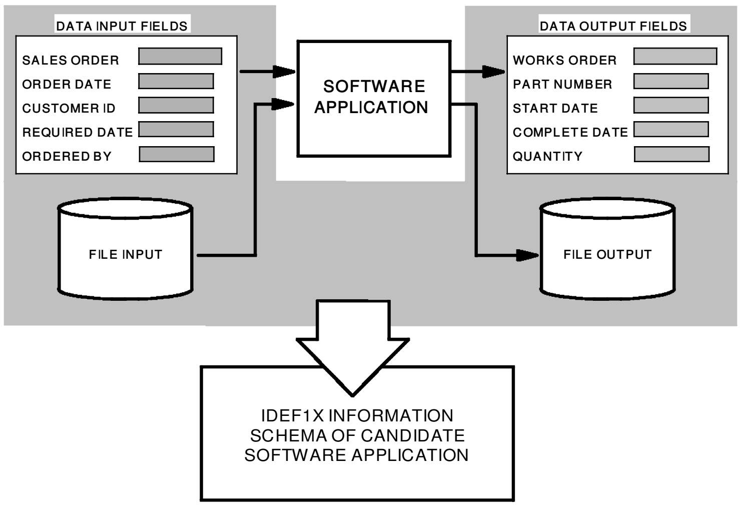


Figure 12. Documentation of the application schema.

material (BOM) list. This list captures the products or parts which the company manufactures or handles. The second category of information relates to the order book information, which links the parts to customer sales orders and works orders. Therefore, the parts list captures the static product structure, which is given 'live' status when parts on the structure are associated to a customer demand through the sales order.

From the functionality standpoint, the software package was therefore considered to be ideal to handle the business front-end tasks required of the company. This includes information relevant to the company itself, e.g. company delivery address, employees, shifts, working hours, etc. The software also meets the requirements for the order entry process, where customer orders are captured along with any relevant information relating to customers, e.g. customer name, invoice address, etc. This is an aspect of the software which is centred around the administrative requirements of the business. The underpinning structure of the software is illustrated in figure 13, using IDEF1x to show the information entities and individual data elements.

Established working procedures and company best practice must be considered when introducing software applications to automate the information handling and processing tasks in a company. Paper-based systems are invariably bound to working procedures (and their peculiarities) in terms of their data structures. Furthermore, the choice of data fields, e.g. job number, material identification numbers, etc. have particular meanings which tend to be company specific and reflect the characteristics of the company procedures they are designed to support.

It is therefore possible to ascertain if there is the potential for mismatches in data usage by documenting and making a comparison of the two sets of information structures. One of the benefits of this is that it enforces a detailed assessment of the vendor solution; availability of the enterprise information schemas provides a very powerful leverage for decision making. This is particularly important for small industries, which require systems that are most supportive of the enterprise operations. The availability of the enterprise information schemas provides the smaller companies with a potential leverage against the purchase of consultancy driven

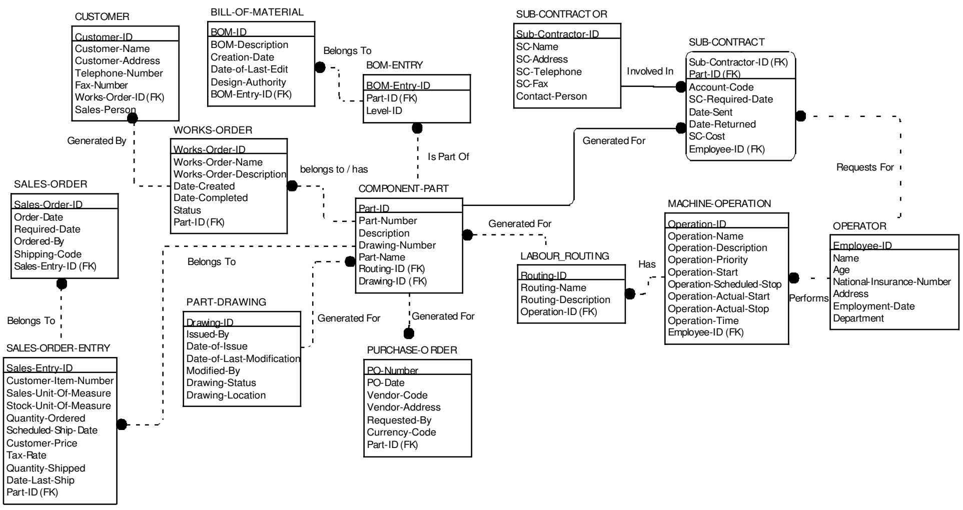


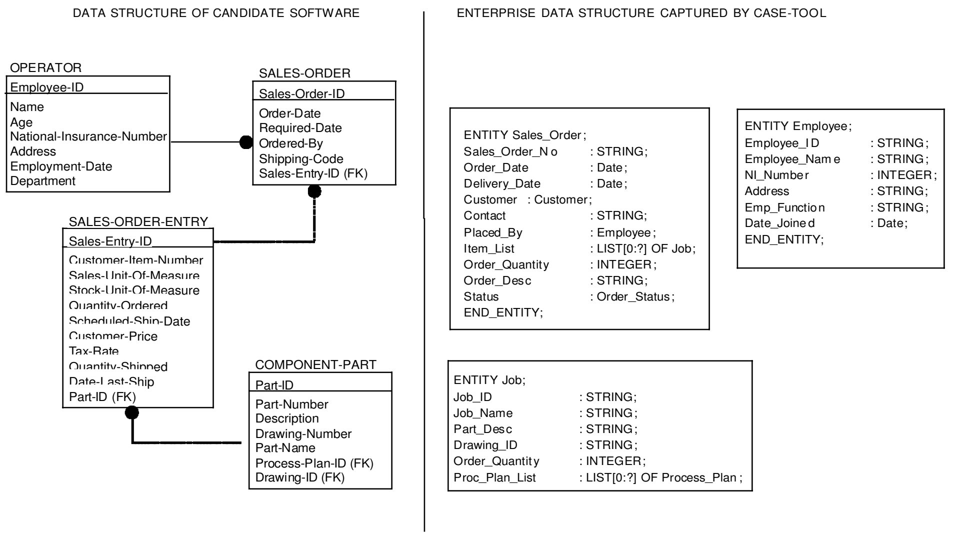
Figure 13. An example of the schema for an order handling application.

solutions, which may constrain the enterprise rather than provide the most effective support.

The information structures captured by the CASE tool are inextricably linked to the functional structures, and therefore, the schemas closely reflect the information requirements of the enterprise. The analysis of the candidate software application involves the comparison of the data structures of the software (shown in IDEF1x) with those of the specification set (captured in EXPRESS). Comparisons can be made at the individual entity level, where the aim is to identify information entities for which users within the enterprise are familiar in terms of usage. Invariably, there will be both similarity in usage (matches in understanding of the semantics) or instances where there will be differences in the understanding of the data semantics. This is illustrated in figure 14, where comparisons are made between sets of data entities which are centred around the sales order and the component part entities. The first set of entities underpins the data structure of the candidate application and the second set captures the data specifications.

It can be seen that there is a similar concept in usage between the two sales order entities. There will invariably be differences in the words chosen to represent the data attributes, e.g. between required_date (candidate software) and the delivery_date (enterprise specification) for the sales order. It is necessary to examine the terms of usage of these attributes to ascertain the understanding of their meaning. For example, the required_date (candidate software) may refer to the date of completion of the sales order and may therefore be different from the delivery_date. It is, therefore, important to ensure that the meaning of these two attributes captures the concept (understanding) in terms of usage in the enterprise, i.e. the specified delivery date for the sales order. In this instance, the individual attributes represented the same element and usage of information.

The other individual entries which constitute the sales order in the candidate application, shown in figure 14, are the component part item, which is the equivalent of a job in the enterprise specifications. The individual attributes, e.g. part-number (Job_ID), part-name (Job_Name), drawing-number

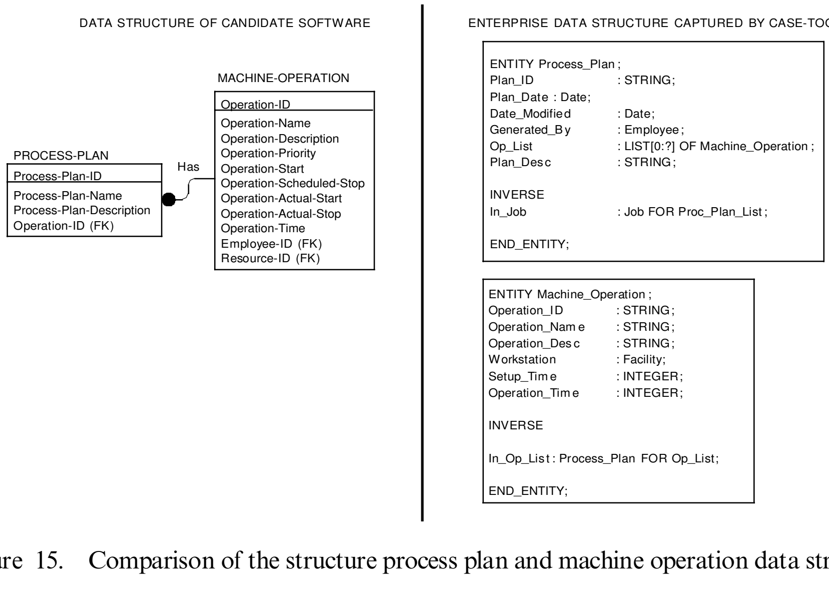


```
ENTITY Sales_Order;
Sales_Order_No      : STRING;
Order_Date          : Date;
Delivery_Date       : Date;
Customer  : Customer;
Contact             : STRING;
Placed_By           : Employee;
Item_List           : LIST[0:?] OF Job;
Order_Quantity      : INTEGER;
Order_Desc          : STRING;
Status              : Order_Status;
END_ENTITY;
```

```
ENTITY Employee;
Employee_ID         : STRING;
Employee_Name       : STRING;
NI_Number           : INTEGER;
Address             : STRING;
Emp_Function        : STRING;
Date_Joined         : Date;
END_ENTITY;
```

```
ENTITY Job;
Job_ID              : STRING;
Job_Name            : STRING;
Part_Desc           : STRING;
Drawing_ID          : STRING;
Order_Quantity      : INTEGER;
Proc_Plan_List      : LIST[0:?] OF Process_Plan;
```

Figure 14. A comparison of data structures between the candidate software and enterprise data specifications.



```
ENTITY Process_Plan;
Plan_ID             : STRING;
Plan_Date : Date;
Date_Modified       : Date;
Generated_By        : Employee;
Op_List             : LIST[0:?] OF Machine_Operation;
Plan_Desc           : STRING;

INVERSE
In_Job              : Job FOR Proc_Plan_List;

END_ENTITY;
```

```
ENTITY Machine_Operation;
Operation_ID        : STRING;
Operation_Name      : STRING;
Operation_Desc      : STRING;
Workstation         : Facility;
Setup_Time          : INTEGER;
Operation_Time      : INTEGER;

INVERSE

In_Op_List : Process_Plan FOR Op_List;

END_ENTITY;
```

Figure 15. Comparison of the structure process plan and machine operation data structures.

(Drawing_ID), etc. are again, similar in terms of the enterprise understanding of their meaning and in usage.

Individual component parts, in turn, consist of a process plan which is identified by the process-plan-ID in the candidate application, shown in figure 15. In relation to this, the job entity comprises a process plan list (Proc_Plan_List: LIST [0:?] of Process_Plan). Each process plan consists of a list of machine operations, which are identified in both instances of schemas. It can be seen that in this case, the software application offers more fields of data capture, where data, e.g. operation priority, operation-scheduled-stop, operation-actual-start, operation-actual-stop can be recorded. It is often common for off-the-shelf applications to offer greater functionality and features in catering to their established niche markets.

It can be seen that this instance represents a case where there is a good match, based on data matching, between the candidate application and the enterprise requirements. The differences in semantics, e.g. between component-part and job can be easily reconciled with usage of the software application. The use of the data structures has allowed such differences to be highlighted and identifies where users of the software will be required to adapt to the new system of usage. It should be remembered that enterprise requirements, captured by the CASE tool, represent both the structure of information and the working procedures of the enterprise. This is one of the primary considerations in this paper, when companies decide to migrate from a paper- or verbal-based system to one which is supported by computer automation. The mapping of the two schemas will highlight the degree of compatibility between the candidate software application and the company requirements.

However, it must be recognized that any software application will invariably result in some changes to the enterprise operation, which are needed to accommodate some of the idiosyncrasies of the application. Although these should be kept to a minimum, there are usually some changes which have to be accepted; indeed, if the software was new, business processes have to be established (if not already done so) and documented.

7. Concluding discussions

The challenges facing the specification of software involve a number of important considerations, e.g. the integration of functionality and integration in the transfer of information. The growing supply of off-the-shelf software, which is both diverse and comprehensive, is displacing the need for custom software. It is envisaged that this approach will gain in importance, as very few organizations have monolithic systems using custom-written software and the practice of building systems has changed from custom programming to systems integration using pre-existing components. It is argued that the purchase of any software application should cause the minimum disruption to the enterprise operation.

There are other considerations, e.g. design of the user interface (ease of use) and functionality which are important but these are out of the scope of this paper. This paper has addressed the use of a CASE tool as part of a methodology, which is to facilitate a greater chance of integrated usage of information across the various applications in a company. The focus of the CASE tool is on the modelling of the business operation and the capture of the enterprise information structures as a first step to the development of a unified information model. The CASE tool and its underlying principles are based on CIMOSA, which is an established modelling architecture and language. However, the constructs of the function viewpoint have been based on holonic concepts which are also suitable for the modelling of small businesses.

This paper has shown how the CASE tool can play a role in a structured approach to software selection by capturing the information requirement of an enterprise. The main functionality of the CASE tool has been described, which includes the modelling and specification of information elements and attributes. These specifications are subsequently used for the development of a unified information schema. It is argued that the information schema reflects the information requirements which support the enterprise operation and is a sound basis for the selection of potential software applications.

The realization of the CASE tool which supports the modelling of the enterprise, from organization, function, resource and information views is a realization of the fundamental CIMOSA concepts at specification level. The subsequent design and implementation stages of the modelling language were not applied. The design stage was only carried out for the information view in the generation of the unified schema which forms the reference model for the subsequent assessment of software applications.

The constructs of the CASE tool enable the information requirements of an enterprise to be captured in a way which allows the development of a formal, logical description of a particular system of information objects and their relationships. The definition and conceptual structure of this information schema becomes an important resource in its own right to the company. The CASE tool allows this, via interfaces, e.g. the resource and process taxonomy modeller, to be subsequently used as a reference data model which has the potential to support a larger number of companies.

The major advantage is that the resultant system, which is specified with reference to the uniform information model, has a greater potential to be integrated in terms of the exchange of information and support for the enterprise functionality. The semantic conflicts would have been resolved at the level of the conceptual schema. Potential redundancy of data and complexity arising over data semantics during information interchange are minimized as the applications are selected based on a reference conceptual schema specification.

Acknowledgements

The authors would like to thank the Department of Manufacturing Engineering, Loughborough University, and acknowledge the very useful interactions which have arisen out of EPSRC project (GR/L27077) 'IT tools to improve the manufacturing performance of metalworking SMEs'. Dr Toh is also grateful to the Committee of Vice-Chancellors and Principals of the Universities of the United Kingdom (CVCP) for the ORS Award and to the Department of Manufacturing Engineering for the research studentship. The support from Professor D. J. Williams and Dr S. T. Newman in the application for the awards is greatly appreciated.

References

AMICE, 1991, *CIMOSA: Open System Architecture for CIM*, 2nd revised and extended edn. (London: Springer).

Ashworth, C. and Goodland, M., 1990, *SSADM: A Practical Approach* (London: McGraw-Hill).

Booch, G., 1991, *Object-Oriented Design with Applications* (London: Benjamin/Cummings).

Brough, M., 1992, Methods for CASE: a generic framework. In *Proc. Fourth Int. Conf. on Advanced Information Systems Engineering*, edited by P. Loucopoulos (London: Springer), pp. 524–545.

Bruce, T., 1992, *Designing Quality Databases with IDEF1x Information Models* (New York: Dorset House Publishing).

Camarinha-Matos, L. M., Pinheiro-Pita, Rabelo, R. and Barata, J., 1995, Towards a taxonomy of CIM activities. *Int. J. Computer Integrated Manufacturing*, **8**, 160–176.

Chen, D., Vallespir, B. and Doumeingts, G., 1997, GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology. *Computers in Industry*, **33**, 387–394.

Colquhoun, G. J., Baines, R. W. and Crossley, R., 1993, A state of the art review of IDEF0. *Int. J. Computer Integrated Manufacturing*, **6**, 252–264.

Doumeingts, G. and Chen, D., 1992, State-of-the-art on models, architectures and methods for CIM systems design. In *Proc. of the IFIP TC5/WG 5.3, 8th Int. PROLAMAT Conference: Human Aspects in Computer Integrated Manufacturing*, edited by G. J. Olling and F. Kimura (North-Holland/Elsevier), pp. 27–40.

Hashemipour, M., Anlagan, O. and Kayaligil, S., 1997, A computer-supported methodology for requirements modelling in CIM for small and medium sized enterprises: a demonstration in apparel industry. *Int. J. Computer Integrated Manufacturing*, **10**, 199–211.

ISO/ OSI, 1995, ISO Basic Reference Model for Open Systems Interconnection (ISO/OSI). International Standards Organisation.

Jorgensen, F., 1995, Overview of functional modelling-IDEF0. In *Information Management in Computer Integrated Manufacturing: A Comprehensive Guide to State-of-the-Art CIM Solutions* (Springer), pp. 340–354.

Jorysz, H. R. and Vernadat, F. B., 1990, CIM-OSA Part 1: total enterprise modelling. *Int. J. Computer Integrated Manufacturing*, **3**, 144–156.

Koestler, A., 1967, *The Ghost in the Machine* (Arkana Books).

Kosanke, K., 1992, CIMOSA a European development for enterprise integration: Part 1: an overview. In *Proc. of the First Int. Conf. on Enterprise Integration Modelling*, edited by C. J. Petrie (The MIT Press), pp. 179–188.

Kosanke, K., 1995, CIMOSA overview and status. *Computers in Industry, Special Issue: Validation of CIMOSA*, **27**, 95–100.

Kosanke, K., Vernadat, F. B. and Zelm, M., 1996, Progress towards establishing CIMOSA models. *Proc. of the Workshop on Process Modelling for Enterprise Integration with CIMOSA*, Loughborough University, UK.

Li, H. and Williams, T. J., 1997, Some extensions to the Purdue enterprise reference architecture: explaining the Purdue architecture and the Purdue methodology using the axioms of engineering design. *Computers in Industry*, **34**, 247–259.

Li, S. and Economopoulos, P., 1997, *Visual C++ 5 ActiveX COM Control Programming* (USA: Wrox. Press).

Microsoft, 1997, *Microsoft Visual C++ 5.0: Development System for Windows* (USA: Microsoft).

Molina, G. A., 1995, A manufacturing model to support data-driven applications for design and manufacture. PhD thesis, Loughborough University, UK.

Monsef, Y. and Teggar, M., 1998, Unified object-oriented approach for process modelling in manufacturing. *International Journal of Modelling and Simulation*, **18**, 298–302.

Murgatroyd, I. S., Hodgson, A and Weston, R. H., 1998, Enterprise modelling in support of business process visualisation and improvement. In *Proceedings of the Institution of Mechanical Engineers* (*IMechE*), Vol. 212, pp. 621–633.

Perakath, C. B., Menzel, C. P., Richard, J. M. and Padmanaban, N., 1995, Toward a method for acquiring CIM ontologies. *Int. J. Computer Integrated Manufacturing*, **8**, 225–234.

Scheer, A. W. and Kruse, C., 1994, ARIS-framework and toolset: a comprehensive business process re-engineering methodology. In *Proc. of the Third International Conference on Automation, Robotics and Computer Vision* (*ICARCV '94*), Singapore, Vol. 1, pp. 327–337.

Schenck, D. A. and Wilson, P. R., 1994, *Information Modelling: The Express Way* (Oxford University Press).

Suda, H., 1990, Future factory system formulated in Japan. *Japanese Journal of Advanced Automation Technology*, **23**, 51–61.

Thomas, J., Mowbray, P. and Zahavi, R., 1995, *The Essential CORBA: Systems Integration Using Distributed Objects* (Chichester, UK: Wiley).

Toh, K. T. K., 1999a, The realisation of reference enterprise modelling architectures. *Int. J. Computer Integrated Manufacturing*, to be published.

Toh, K. T. K., 1999b, Modelling architectures a review of their application in structured methods to systems specification. *Int. J. Production Research*, **37**, 1439–1458.

Toh, K. T. K., Newman, S. T. and Bell, R., 1997, An information systems architecture for small metal working companies. *Journal of Engineering Manufacture* (*Part B*), *Proceedings of the Institution of Mechanical Engineers* (*IMechE*), **212**, 87–103.

Tsichritzis, D. and Klug, A., 1972, The ANSI/X3/SPARC DBMS framework: report of the study group on database management systems.

Uppington, G. and Bernus, P., 1998, Assessing the necessity of enterprise change: pre-feasibility and feasibility studies in enterprise integration. *Int. J. Computer Integrated Manufacturing*, **11**, 430–447.

Vernadat, F. B., 1996, *Enterprise Modelling and Integration: Principles and Applications* (London: Chapman & Hall).

Williams, T. J., 1994, The Purdue enterprise reference architecture. *Computers in Industry*, **24**, 141–158.

Williams, T. J, Bernus, P., Brosvic, J., Chen, D., Doumeingts, G., Nemes, L., Nevins, J. L., Vlietstra, J. and Zoetekouw, D., 1993, Architectures for integrating manufacturing activities and enterprises. *Proc. of the JSPE/IFIP TC5/WG5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing DIISM'93*, Tokyo, Japan, edited by H. Yoshikawa and J. Goossenaerts (Amsterdam: North-Holland/Elsevier), pp. 3–18.