AN APPROACH TO ENACTING BUSINESS PROCESS MODELS IN SUPPORT OF THE LIFE CYCLE OF INTEGRATED MANUFACTURING SYSTEMS

by

Marcos Wilson Costa de Aguiar

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy

of Loughborough University of Technology

Department of Manufacturing Engineering

February 1995

Acknowledgements

I wish to thank Prof. R. H. Weston for supervising and Prof. N. D. Burns for co-supervising the work of this thesis; Dr. N. Schofield and Dr. J. M. Edwards for the valuable comments made in my oral examination; and all colleagues of the Manufacturing Systems Integration Research Institute, particularly I. A. Coutts and I. S. Murgatroyd for their contribution and support to this research.

The assistance of D2D's skilled personnel, in particular S. Roberts should also be duly acknowledged.

Thanks to CAPES in the Ministry of Education of Brazil are also due for sponsoring my Ph.D. research.

Finally, my deepest acknowledgements are to my wife and my son for their patience and support throughout the work which has taken many of the hours that should have been dedicated exclusively to them.

This thesis is dedicated to

Daniel and Lucia, my partners in life.

Certificate of Originality

This is to certify that I am responsible for the work submitted in this thesis, that the original work is my own except as specified in acknowledgements or in footnotes, and that neither the thesis nor the original work contained therein has been submitted to this or any other institution for a higher degree.

Synopsis

The complexity of enterprise engineering processes requires the application of reference architectures as means of guiding the achievement of an adequate level of business integration. This research aims to address important aspects of this requirement by associating the formalism of reference architectures to various life cycle phases of integrating manufacturing systems (IMS) and enabling their use in addressing contemporary system engineering issues.

In pursuit of this aim, the following research activities were carried out: (1) to devise a framework which supports key phases of the IMS life cycle and (2) to populate part of this framework with an initial combination of architectures which can be encapsulated into a computer-aided systems engineering environment. This has led to the creation of a workbench capable of providing support for modelling, analysis, simulation, rapid-prototyping, configuration and run-time operation of an IMS, based on a consistent set of models associated with the engineering processes involved. The research effort concentrated on selecting and investigating the use of appropriate formalisms which underpin a selection of architectures and tools (i.e. CIM-OSA, Petri-nets, object-oriented methods and CIM-BIOSYS), this by designing, implementing, applying and testing the workbench.

The main contribution of this research is to demonstrate that it is possible to retain an adequate level of formalism, via computational structures and models, which extend through the IMS life cycle from a conceptual description of the system through to actions that the system performs when operating. The underlying methodology which supported this contribution is based on enacting models of system behaviour which encode important coordination aspects of manufacturing systems. The strategy for demonstrating the incorporation of formalism to the IMS life cycle was to enable the aggregation into a workbench of knowledge of 'what' the system is expected to achieve (i.e. 'problems' to be addressed) and 'how' the system can achieve it (i.e possible 'solutions'). Within the workbench, such a knowledge is represented through an amalgamation of business process modelling and object-oriented modelling approaches which, when adequately manipulated, can lead to business integration.

Table of Content

List of Figures

List of Tables

Chapter 1 - Introduction

The ISO TC 184/SC5 "Framework for Enterprise Modelling" [ISO 1993] is based on the postulate that **systems theory** and **enterprise modelling** can facilitate the process of **enterprise integration**. The following sections discuss this notion as a means of identifying the overall goals of this research.

1.1. Systems Theory

Systems theory emerged as a field of study from the biological and engineering sciences after the second world war. Its application to other fields of study was pioneered by Norbert Wiener in his early work on cybernetics [Wiener 1948]. Wiener proposed that the same principles used to control engineering and biological systems may be applied to the market mechanisms of economic systems, the decision-making mechanisms of political systems, and the cognitive mechanisms of psychological systems. He argued that (from [Roberts 1984]):

> "all aspects of human behaviour, ranging from the economic to the political to social and psychological, may be governed by a single set of governing principles." [i.e. systems thinking]

Although cybernetics has yet to prove such a claim, considerable progress has been made in applying systems theory to organisations, notably in Beer's "Viable System Model", Ashby's "law of requisite variety" (from [Espejo 1989]), Forester's "system dynamics" (from [Roberts 1984]) and Checkland's "Soft-Systems Methodology "[Checkland 1981].

The main features of general systems thinking on organisational dynamics, as stated by Stacey [Stacey 1993], are that: organisations are open systems comprised of interconnected parts which interact with one another and with their environment. The system imports energy and information from its environment and exports the transformed results. Imports and exports occur across the organisation's boundary.

According to Roberts [Roberts 1984],

> "Systems thinking is concerned with connectedness and wholeness. By its nature, a systems view of a problem cuts across disciplinary boundaries as defined in many traditional sciences, in a search to **understand a problem from an integrated vantage point.**"

Such an integrated view of organisations represents a shift in the basis on which organisations have traditionally been constructed. Wheatley [Wheatley 1992] contrasts the systems approach with mechanistic thinking which served as the basis for the design of traditional organisations, by stating that:

> "In the machine model [by Isaac Newton], one must understand parts. Things can be taken apart, dissected literally or representationally (as we have done with business functions and academic disciplines), and then put back together without any significant loss. The assumption is that by comprehending the workings of each piece, the whole can be understood. The Newtonian model of the world is characterized by materialism and reductionism - a focus on **things** rather than **relationships** and a search, in physics, for the basic building blocks of matter.
>
> In new sciences [i.e. quantum physics, chaos theory, dissipative structures and self-organising systems], the underlying currents are a movement toward holism, toward *understanding the system as a system and giving primary value to the* **relationships** *that exist among seemingly discrete parts*[1]."

This view is reinforced by Waldrop [Waldrop 1994] when he argues that

> "one should look at systems in terms of **how they behave instead of how they are made**".

Thus, complexity in the behaviour of a system emerges in the form in which the relationships among system components are manifest. In Stephen Wolfram's view (from [Waldrop 1994]):

> "The complexity is actually in the organization - the myriad possible ways that the components of the system can interact."

Systems theory provides an approach to unfolding the inherent complexity of a system described as a web of relationships among its parts (i.e. sub-systems and sub-systems components), as well as between the whole system and any other elements of the environment with which the system interacts.

---

1. All emphasis and italics in the citations were added by the author (when the term "author" is used in this document it always refers to the author of the thesis).

1.2. Enterprise Integration

Based on systems thinking [Checkland 1981] [ISO 1994] [Wu 1992], a manufacturing enterprise can be represented as a system, according to the representation depicted in Figure 1.a. Such a system interacts with its environment by means of a series of relationships with competitors, customers, suppliers and partners, all subject to technological, social, political, economical and other factors inherent to the environment where they operate.

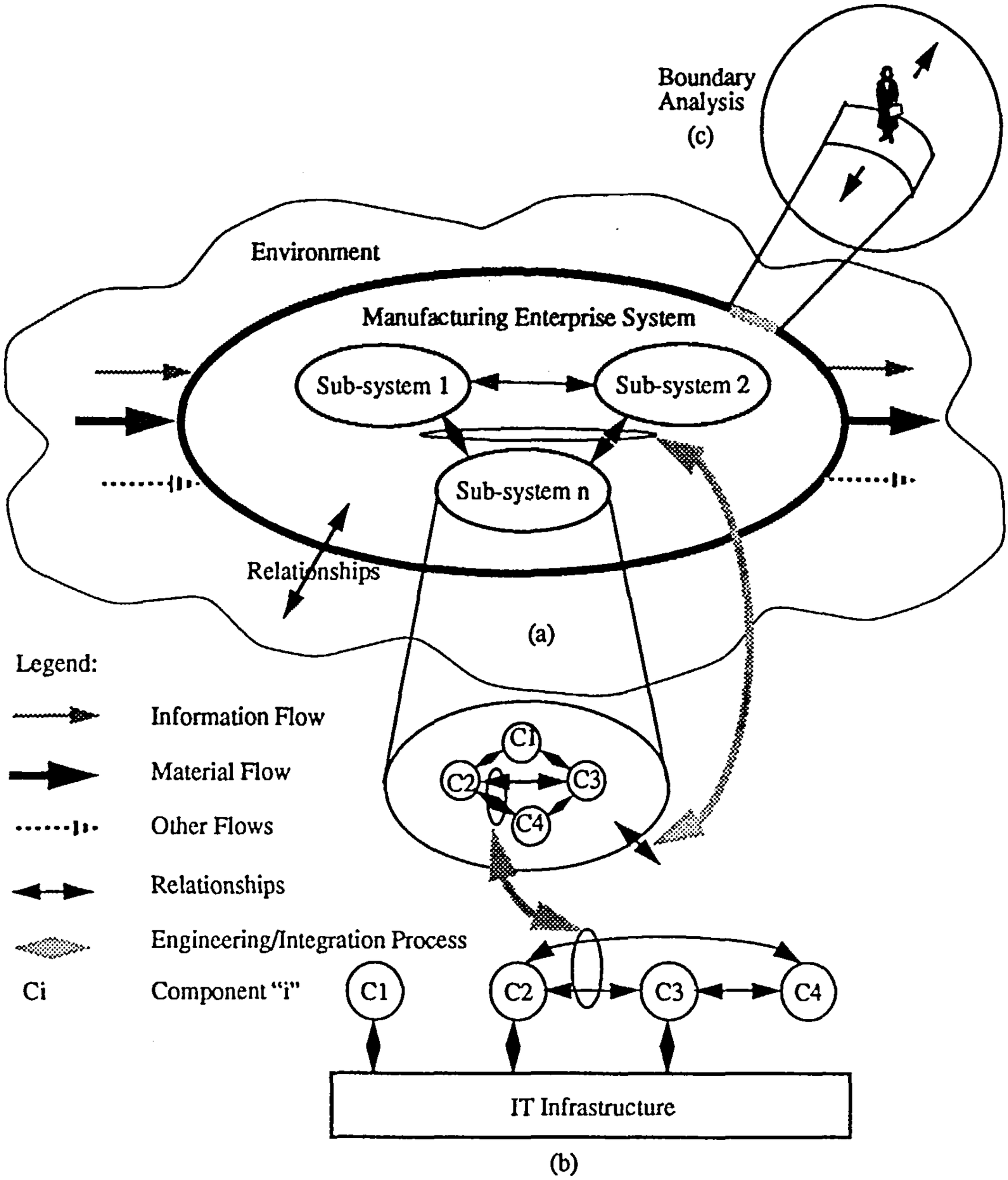


Figure 1 - Systems view of the enterprise engineering/integration problem

Coping with the complexity involved in engineering an enterprise system[1] which is able to thrive in its environment embraces the task commonly referred to as "enterprise integration"[2].

Essentially, enterprise integration is the task of **engineering** enterprises as complex systems which operate in an even more complex environment. Wheatley [Wheatley 1992] summarises the complexity involved in such a task in a few questions:

> "How do we create organizational coherence, where activities correspond to purpose? How do we create structures that move with change, that are flexible and adaptive, even boundaryless, that enable rather than constraint? How do we simplify things without losing both control and differentiation? How do we resolve personal needs for freedom and autonomy with organizational needs for prediction and control?"

The ISO TC 184/SC5 "Framework for Manufacturing Enterprise Modelling" [ISO 1993] refers to this task as "Integrated Manufacturing", whereby

> "... enterprises align their structure, processes, information, resources and responsibilities to the greatest extend possible towards their common goals of manufacturing products. When computers and information technology are central to the realisation of this organisation, the process and result are called **Computer Integrated Manufacturing** [i.e. CIM][3]".

In this context, one aspect of enterprise integration (or CIM) is its focus on

---

1. According to IFIP/IFAC Task Force [Bernus 1994], such an engineering process is: "a highly sophisticated, multi-disciplinary management, design and implementation exercise during which various forms of descriptions and models of the target enterprise need to be created". In this thesis, terms such as enterprise system, enterprise, integrated enterprise and integrated system are referred to as "integrated manufacturing system".
2. According to the Enterprise Integration Program (EIP) funded by ManTech [Petri 1992a]: "Enterprise Integration is the task of improving the performance of such large complex processes by managing the interactions among the participants". According to Fox [Fox 1993], "Enterprise Integration is concerned with how to improve the performance of distributed organizations and markets. It focuses on the communication of information and the coordination and optimization of enterprise decisions and processes in order to achieve higher levels of productivity, flexibility and quality."
3. The terms IMS, CIM system and integrated enterprise are used interchangeably in this thesis, as are CIM, system integration and enterprise integration. The difference between the use of these terms often lies in the scope or focus of their application.

understanding the behaviour of systems, sub-systems and components, emerging from their inter-relationships (which can be represented at varying levels of abstraction, as illustrated in Figure 1). The impact of information technology (i.e. IT) in improving the performance of an enterprise can be understood as a means of achieving an adequate design of the "parts" and "relationships" in the enterprise system. IT can function as both an enabler and a mechanism for establishing these relationships (i.e. to realise **integration in a broad sense).**

IT-based integration implies that a system, such as that represented in abstract form in Figure 1.a, ends-up by having certain of its components integrated total or partially by means of an IT infrastructure (as depicted in Figure 1.b). System components are the actual resources of the system and will be manifest in the form of human beings, machines, application programs, etc.

The ESPRIT/AMICE[1] consortium [ESPRIT/AMICE 1993b] defined three levels of IT-based integration, namely: physical integration, application integration and business integration (see Figure 2).

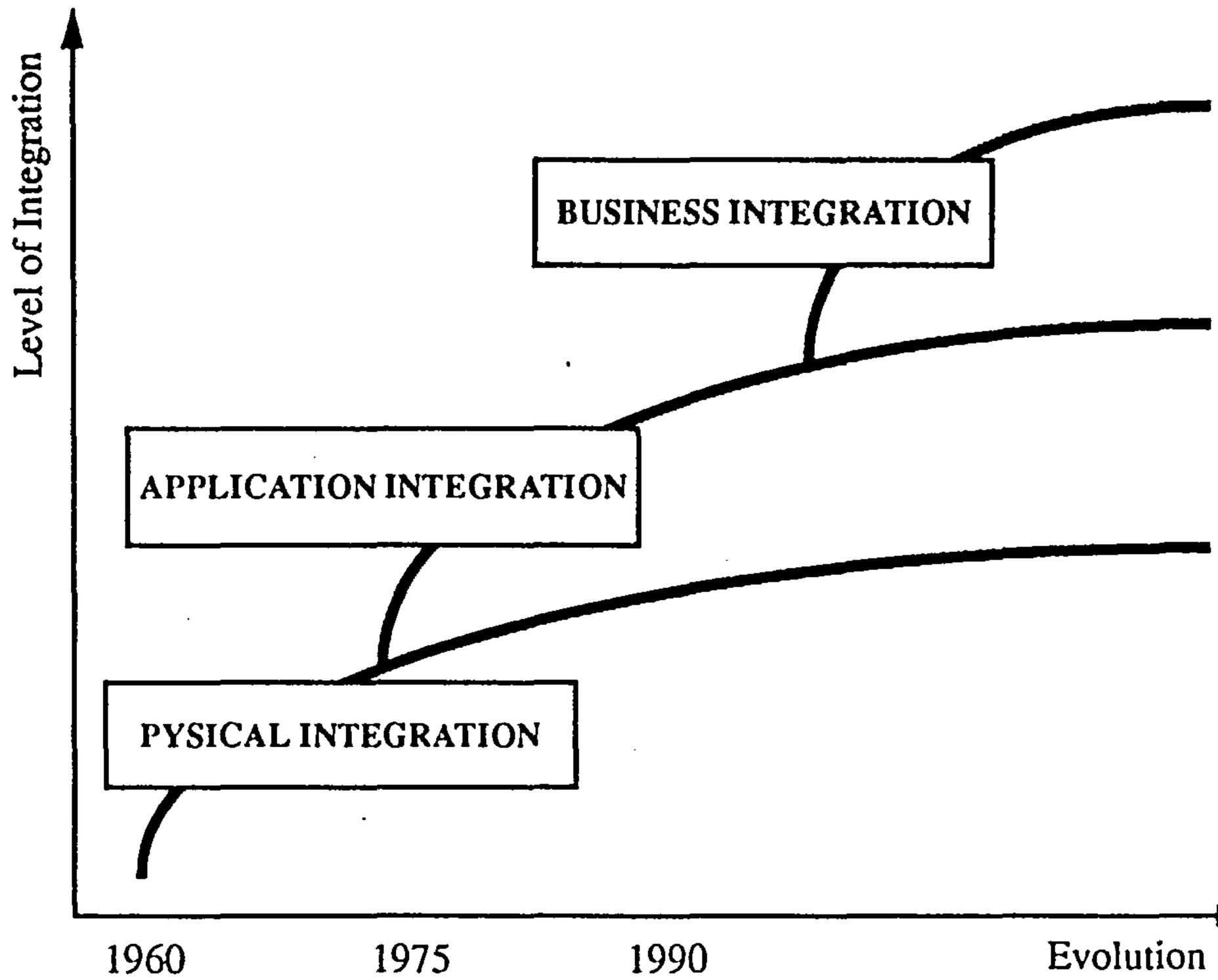


Figure 2 - Integration levels in the manufacturing enterprise

1. ESPRIT/AMICE stands for European Strategic Program of Research in Information Technology/European CIM Architecture.

a. **Physical integration** [ESPRIT/AMICE 1993b] "is concerned with the interconnection of manufacturing automation and data processing facilities to permit interchange of information between the so called 'islands of automation'".

b. **Application integration** [ESPRIT/AMICE 1993b] is concerned with the integration of software applications, thus considering issues of inter-operation between software applications, human beings and machines and the provision of IT support for transparent access to information.

c. **Business integration** [ESPRIT/AMICE 1993b] "is concerned with the integration of those functions which manage, control and monitor business processes [...] and in turn coordinate the day-to-day execution of activities at the application level. **Modelling business processes and their interrelations and [using it] for decisions and operation support is key to business integration".**

These levels of integration can be summarised in their concerns with, respectively, **physical integration of parts, interoperation of parts** and **coordination of parts**. The issues of business process integration (or simply 'business integration') are the major interest of this thesis.

According to Davenport and Short, a business process is (from [Childe 1993])

> "the logical organisation of people, materials, energy, equipment and procedures into **work activities** designed to produce a specified result".

Hickman defines a business process as (from [Childe 1993])

> "a logical series of dependent activities which use the resources of the organisation to create, or result in, an observable or measurable outcome, such as a product or service".

Therefore, business processes are abstractions which encapsulate high-level descriptions of an enterprise system in terms of work activities. This, in turn, provides the link between 'what' the overall system is expected to do and 'how' that translates into the definition and coordination of the activities performed by the acting parts of the system[1]. In this context, the importance of achieving business integration is emphasized by Petrie [Petrie 1992a] in his statement that

---

1. "To reduce or at least hide complexity [...] requires a better understanding of the system through a structured representation of its contents. Business process modelling [offers a] solution to the structured representation of the operation" [Petrie 1992a].

> "the enterprise integration hypothesis is that it is our ability to manage the complex interactions among organizations and people that currently limits large industrial endeavours."

The importance of enterprise modelling in achieving IT supported business integration is emphasised by Petrie [Petrie 1992a] when he argues that

> "... just exactly what support computers can offer for such coordination and control is still a matter for research and field experiments. But **modelling the interactions among elements of an enterprise is the first step to providing a system that improves enterprise integration.**"

1.3. Enterprise Modelling

Part of the enterprise modelling challenge is that of enabling business integration (through adequate application of information technology) in order to construct highly dynamic systems capable of coping with their ever-changing environment, which characterises today's manufacturing market. The ISO TC 184/SC5 [ISO 1994], "Framework for Enterprise Modelling", advocates the **need for formalism**[1] in dealing with the inherent complexity of such a process, by employing **enterprise modelling**[2] and **system theory** [ISO 1993] [Petrie 1992a] [Williams 1993] [CEN 1994a] [ESPRIT/AMICE 1993b].

According to the Enterprise Integration Program (EIP) [Petrie 1992a], enterprise modelling is:

> "used to clarify the interactions of enterprise components so that these interactions can be rationalized and improved."

---

1. According to ISO TC 184/SC5 [ISO 1993], formalism is "the formalised mode of representation used for the elaboration of models. This representation is said to be formal because it allows a common[ly] understood and non-ambiguous representation of all aspects of an enterprise model. It is achieved by a formal language made of a syntax and a dictionary."
2. A model, according to Jorgenson [Jorgenson 1992], is "a structured representation of physical objects, concepts, or a system that helps organise, clarify and unify knowledge; containing a system of rules, data, and inferences presented as a formal logical description of a system of objects and their states of affairs, or **interactive behaviour; that will facilitate analysis, experimentation, simulation, or comprehension**". The ICEIMT's Working Group 1 [Petrie 1992c] identifies "two central functions of models as they are currently used: to filter out irrelevant detail and thereby display only information that is essential to the task at hand, and to represent that information in a useful way."

One of their recommendations is that [Petrie 1992a]

> "Models have to provide answers to particular business questions and have to be constructed with these questions in mind. Models that are not used in running the business become 'shelfware' and quickly become inaccurate and completely useless for any purpose. The consensus [in the EIP] was that **models should be used to control and monitor the business itself, rather than only being used for analysis and decision support.**"

Thus, whilst systems theory may offer an adequate weltaschuung[1], enterprise modelling may provide the means (such as by providing a basis for creating tools) to determine how enterprise integration should be achieved. However, for a combination of systems theory and enterprise modelling to be generally applied to manufacturing enterprise systems, there is a need to identify appropriate formalism.

1.3.1. Life cycle

As enterprise integration is a staged process, a central concept in understanding the type of formalism associated with such a process is its life cycle. In the view of ISO TC 184/SC5, the scope of application of enterprise modelling comprises three stages, namely: Business Modelling (i.e. conceptual analysis), Technical System Design (i.e. design) and Realisation (i.e. implementation). These stages take as input the definition of business issues and deliver as output a usable system. The resulting five stages (this including inputs and outputs to the scope covered by the ISO initiative, as depicted in Figure 3)[2] are described in the following [ISO 1994]:

- identifying **business issues** (i.e. strategy definition): this stage focuses on the "boundary analysis"[3] illustrated in Figure 1.c, whereby key strategic issues that drive the performance of the enterprise in its environment are identified.
- defining **what to do** about business issues (i.e. conceptual analysis): this stage embraces conceptual analysis of how the enterprise system should be organised in order to address the business issues (this is illustrated by the sub-system models in

---

1. The German term for a 'view of the world'.
2. Essentially, there is a consensus view in the modelling community that these five life cycle phases commonly occur, although variations in their scope and denomination exist [ESPRIT/AMICE 1993b] [Williams 1993] [Bernus 1994]. These stages will be referred to henceforth in this thesis as the 'integrated manufacturing system (IMS) life cycle'.

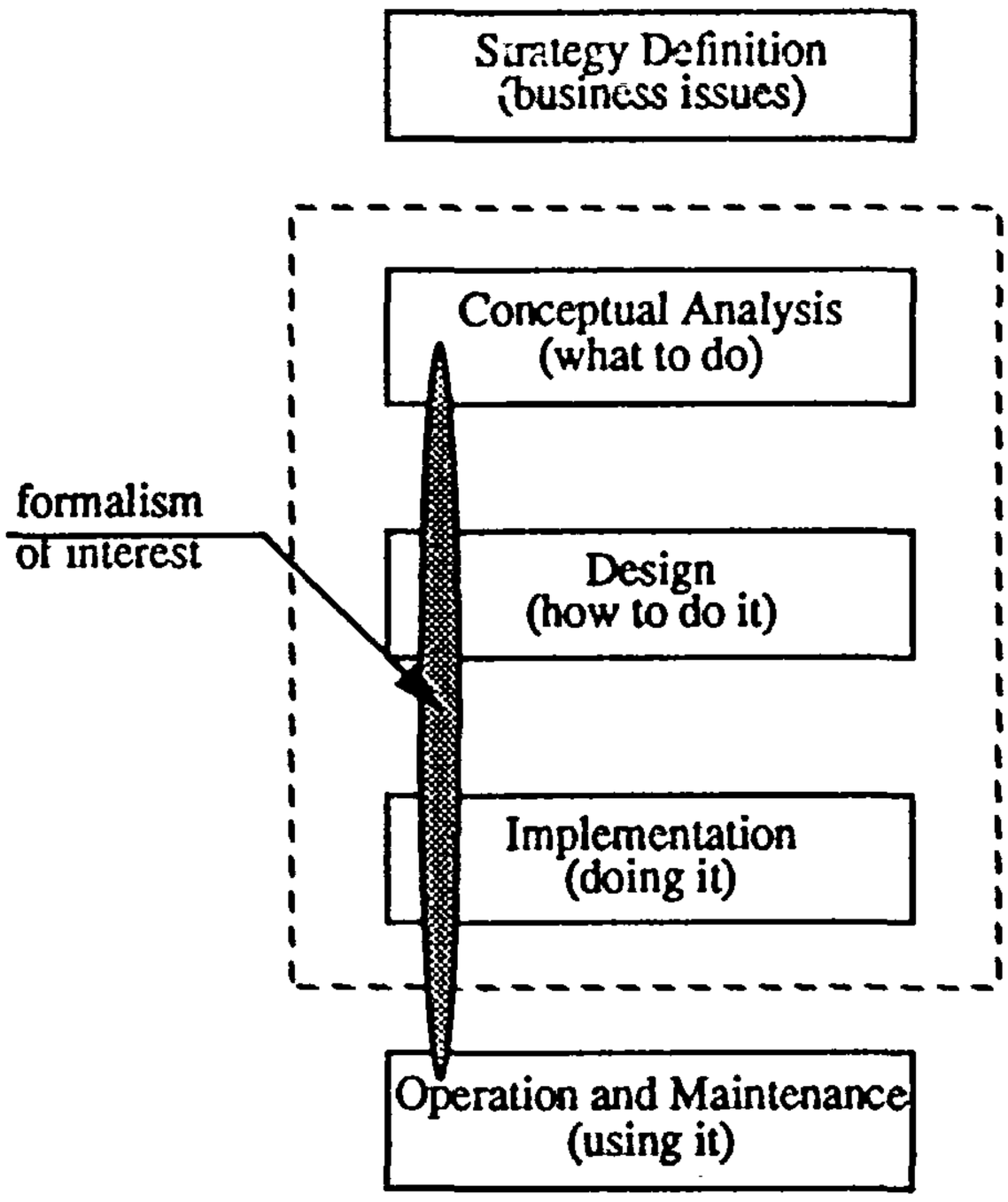


Figure 3 - Stages in the integrated manufacturing system life cycle

Figure 1.a).

- defining **how to do it** (i.e. design): at this stage, the identification of system components (which can either be of a manufacturing technology or information technology nature) and the definition of how they inter-relate (i.e. how these components are integrated) are addressed (this is illustrated by the components models in Figure 1.b);
- **doing it** (i.e. implementation): at this stage, components and integration mechanisms are physically realised to construct the system (this is illustrated by the components integration in Figure 1.b).
- and **using** the resulting enterprise system (i.e. operation and maintenance).

---

3. The process of strategy definition is represented in Figure 1.c by the boundary analysis or static analysis [Stacey 1993], whereby the posture and the position of an organisation are analysed. According to Stacey [Stacey 1993], "the posture of an organisation is a picture of what it looks like, its shape and its capabilities. The posture of an organisation is what you see when you stand at the organisational boundary and look inwards. Position, then, is a picture of how an organisation relates to the people that are its environment. The position is what you see when you stand at the organisational boundary and look outwards. The posture and the position of the organisation are the result of the strategy [...] it has pursued. That posture and position determine its performance [...]".

The role of modelling in the life cycle is illustrated by the reference model for open distributed processing (RM-ODP) when referring to enterprise integration as a [ISO/IEC 1993] "process of creating and using an IT system to serve the needs of an enterprise". This process involves three main elements (see Figure 4): (1) the enterprise, constrained by its environment and, indirectly, by technological constraints; (2) the models, constrained by modelling methods; and (3) the IT system constrained by technology.

"Models developed in [the] idealisation process describe the IT requirements of the enterprise", whereas "through the realisation process, models are used to implement an IT system" [ISO/IEC 1993]. These models (which characterise the prime formalism of interest in this research) can be related to the models manipulated at the "conceptual analysis" and "design" stages which in the ISO framework.

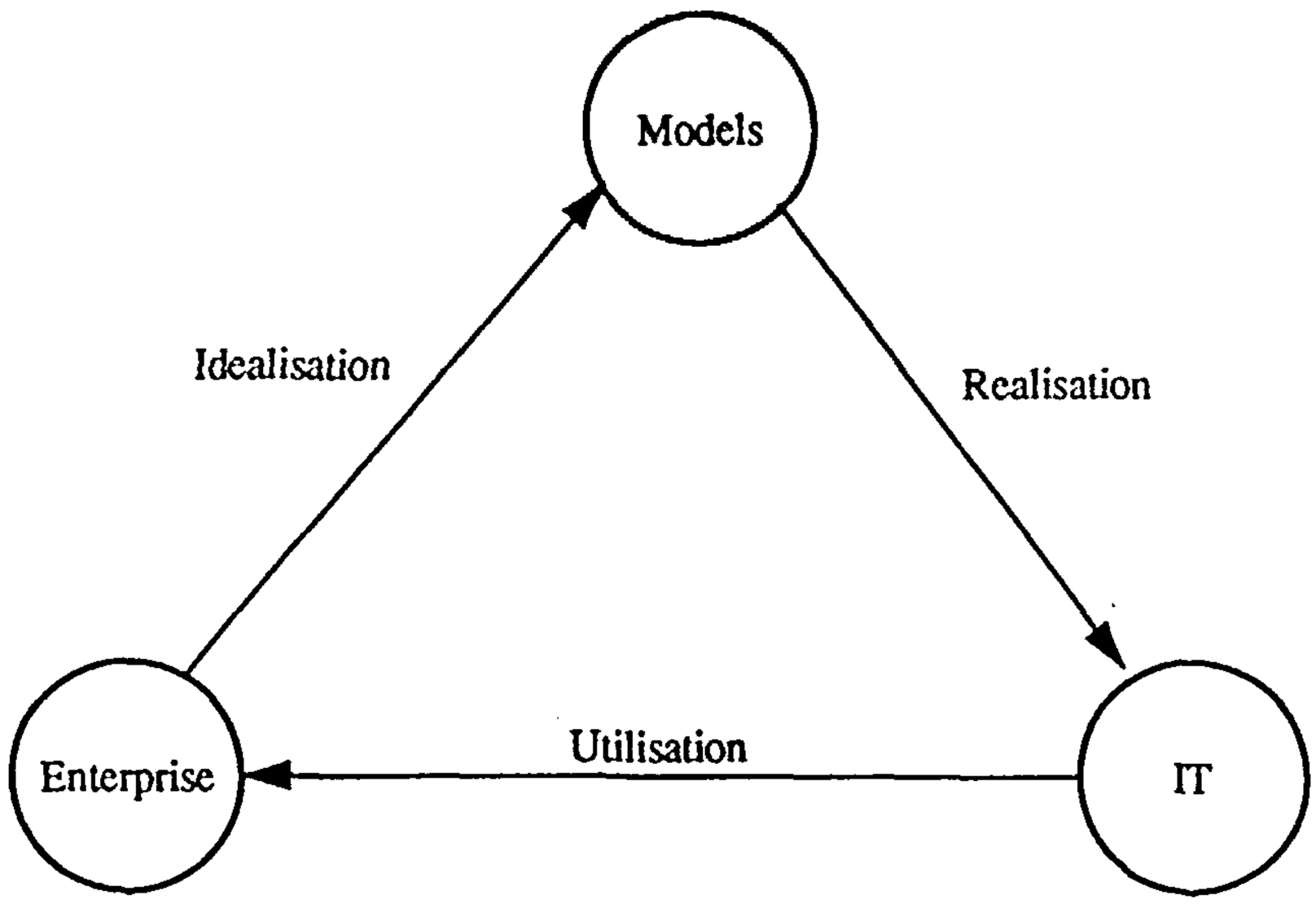


Figure 4 - Modelling process and enterprise integration [ISO/IEC 1993]

1.3.2. Reference architectures

Formalism to support the IMS life cycle is viewed by the IFIP/IFAC "Task force on architectures for integrating manufacturing activities and enterprise"[1] [Williams 1993], as being provided by an enterprise reference architecture which:

> "...models **the whole life history** of an enterprise integration project [i.e. the IMS life cycle] from its initial concept in the

1. Henceforth referred to as the "IFIP/IFAC Task Force"

eyes of the entrepreneurs who initially developed it, through its definition, functional design or specification, detailed design, physical implementation or construction, and finally operation to obsolescence [i.e. the mapping across life cycle phases]. The architecture becomes a [...] framework upon which all of the functions and activities involved in the aforementioned phases of the life of the enterprise integration project can be mapped."

Hence, enterprise modelling and, more specifically, reference architectures are viewed as a means of supporting the mapping between models across life cycle phases, by encapsulating a level of formalism into enterprise integration processes. Such a formalism is required to enable capturing knowledge about 'problems' and 'solutions' (i.e. domain specific knowledge) and manipulate it adequately in the light of an enterprise integration problem to which the formalism is being applied (as represented in Figure 5).

As part of such a formalism, the IFIP/IFAC Task Force [Williams 1993]

> "determined that the potential user of architectures vitally needed instruction on "how" to design, develop, implement and use manufacturing and enterprise integration [which should go beyond] descriptions or merely designs for the related computer system. [They should describe] both the integrated system and its full life history of development and use."

The IFIP/IFAC Task Force encapsulates the multitude of issues associated with achieving enterprise integration by listing the main requirements of a reference architecture, namely [Williams 1993]:

- "the best treatment of the enterprise scope from the system theoretic point of view" (i.e. all activities which are "involved directly or indirectly in designing, operating or improving the enterprise should be covered by the architecture");
- "the provision of a consistent modelling environment leading to executable code" (i.e. a computer-aided systems engineering environment - henceforth in this thesis referred to as CASSE[1] environment or workbench);
- "the existence of a detailed methodology which enterprises can follow" which can support the complete IMS life cycle;

---

1. The acronym CASSE is used to differentiate from CASE for software engineering.

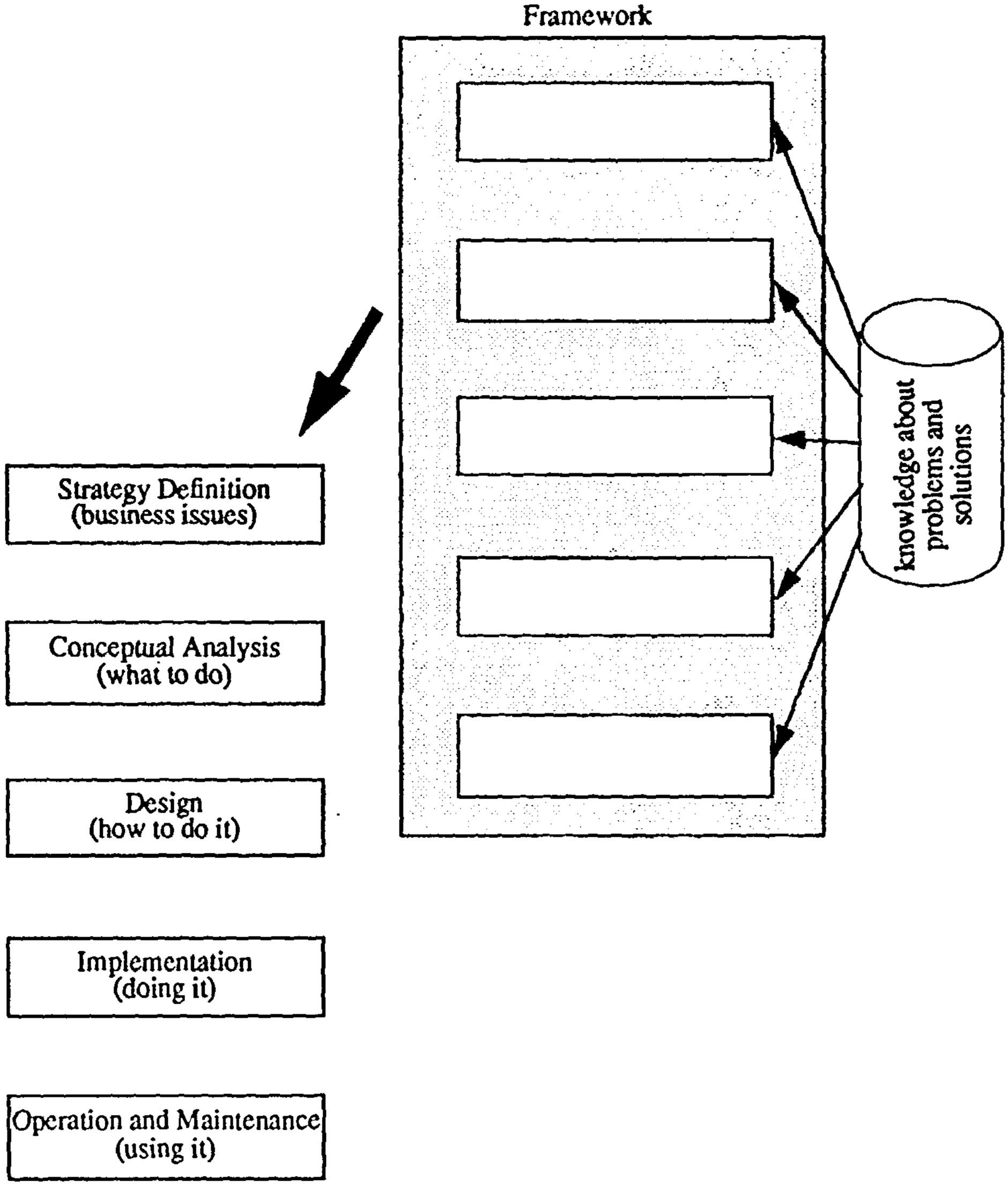


Figure 5 - Model-based enterprise engineering process

- "the adoption of good engineering practice for building reusable, tested, and standard models";
- "the provision of a unifying perspective for products, processes, management, enterprise development and strategic management" (i.e. it should provide means of relating the enterprise integration initiative with the remaining operational activities of the enterprise).

However, incorporating the formalism of architectures into the IMS life cycle involves a complex process of bringing all these requirements together into an organised whole, so that adequate support for the enterprise engineering process can be achieved.

As part of such complexity, two issues are of particular interest in this research:

a. **What level of formalism should be used in the framework of Figure 5 in order to support and link the various IMS life cycle phases and hence the activities performed by the designers and builders of such systems?**

   This embraces the need to adopt a 'satisfactory' level of formalism from specifications of reference architectures which seek to support the IMS life cycle. Embodied in this study of the use of formalism of currently available reference architectures is the question of devising what needs to be captured by the formalism found in a reference architecture and what should be captured as domain specific knowledge.

b. **How effective can such a formalism be used to realise integration of real manufacturing systems?**

   This implies the need to define a means of combining formalism with knowledge of 'problems' and 'solutions' to enable their application to contemporary integration problems. Embodied in such a study is the need to establish a compromise between the requirement of general applicability of the adopted and the degree of prescription with which methods and solutions should be imposed on system designers and builders.

Concerning the capability embodied in such a formalism, the IFIP/IFAC Task Force has identified some of the desirable features of an architecture, namely [Williams 1993]:

> "**Verification of completeness and consistency** for all described functions and objects at any detailing level; **simulation of the enterprise model** at any detailing level; easy and fast **change of the model in case of changing business processes**, methods and tools; the **use of the model to initiate, monitor and control the execution of the enterprise's daily operation.**"

In regard to this embracing view of the application of modelling technology, Mize [Mize 1992] points out that

> "Robust modelling methods should be able to **accommodate both analytical and simulation approaches based on the same descriptive model.** Finally, any chosen modelling method should accommodate mixed-level modelling,

> permitting different system elements to be modelled at varying degrees of detail."

He also highlights that in order to attain such a condition, the following key areas of research need to be addressed:

> "(1) system design and modelling methodologies for organisations experiencing continual change, (2) model representation schemes and methodologies which support all phases of a model's life cycle, (3) modelling approaches which support hybrid modelling as well as multiple- and mixed-level modelling".

Petrie [Petrie 1992a] illustrates a use of enterprise modelling within the context of business integration, in the following example

> "... eliminating unnecessary connections and paths among organisations and processes is one way to improve coordination. This usually requires a static model of the enterprise that can be used to analyse the efficiency of the existing structure or sequencing. Once a change has been decided upon, one may **need to use an executable model that can simulate the enterprise to the extent that the effect or proposed changes can be seen.**"

Concerning the required means of enabling the application of modelling technology to enterprise integration, Jorgenson [Jorgenson 1992] argues for the need for formal modelling environments[1], when he states that

> "Formal modelling [...] technology environments provide improved capabilities for modelling and for integration of models as a means [of identifying] problems, simulate changes, and give direction to managing change and integration."

Based on these views (i.e. [Williams 1993] [Mize 1992] [Petrie 1992a] [Jorgenson 1992]) one may argue that a CASSE environment which encapsulates the

---

1. A formal modelling environment is a model-repository-based computing system (e.g. CASE technology) for retaining and managing enterprise process definitions and computing automation design definitions and associated design history, being the enterprise definitions and designs in model form [Jorgenson 1992].

formalism of architectures should provide facilities for modelling and analysis of an IMS (thus, leading to model creation), simulation of such a model, as well as design, implementation and operation of the IMS based on the same models. Testing such an assumption is one of the motivations of this research.

Additionally, concerning the application of modelling concepts embodied in frameworks for IT standards (and indeed in such a CASSE environment), the DISC "Framework for User Requirements" [Millis 1992] argues that

> "IT standards are a means to an end, and frameworks for IT standards are a means to a means to an end [...] It would be possible to do endless theoretical work without immediate benefit. What we need is some practical testing of value and even better some exploitation yielding useful results."

In summary, there is a need to apply architectures in support of the construction of integrated manufacturing systems.

1.4. Research Requirements

The key issues raised in this chapter provide the background rationale to the definition of the need for the research reported in this thesis. Such a background can be summarised in the following statements:

- systems theory provides a basis for studying an enterprise system as a web of relationships among its sub-systems and components;
- enterprise integration is a means of engineering an enterprise system by providing adequate support for the interactions (i.e. the relationships) among its sub-systems and components;
- information technology can function as an enabler for attaining adequate interactions so that enterprise integration can be achieved;
- three levels of integration can be identified in the support for such interactions (as depicted in Figure 2), namely: physical (integration of parts), application (i.e. interoperation of parts) and business (i.e. coordination of parts), whereby business integration is of particular interest.
- The life cycle stages (shown in Figure 3) represent a view into the multi-faceted complexity of the enterprise integration process.
- Reference architectures for enterprise modelling can provide the formalism required for coping with the complexity involved in such a life cycle.

- Benefits can be gained by using the formalism encapsualted into **CASSE** environments to engineer contemporary enterprises with a view to enabling IT-based business integration of its processes and components.

When incorporating the formalism of models and architectures into the IMS life cycle, the framework shown in Figure 5 needs to be populated in a way which **enables system modelling, analysis and simulation at different levels of abstraction to be performed on the same set of models which are used to generate the system to be created or transformed in such a way that they can also help formally structure and drive the system operation.**

The issue then becomes one of defining **how the above requirements can be achieved** through incorporating the formalism of currently available reference architectures in a formal modelling environment (i.e. software tools that cover a broad span of the life cycle). Thence, questions "a" and "b" on page 13 can be re-stated respectively as:

- How can the formalism of models and architectures be amalgamated into a usable whole?
- How can such an amalgamated formalism be organised so that it can be applied to support the IMS life cycle?

Therefore, based on the assumption that the formalism of reference architectures incorporated in a CASSE environment can provide a basis for supporting the activities along a broad span of the IMS life cycle, this research aims to investigate the issues involved in the realisation, application and evaluation of such an environment, thereby testing such an assumption.

1.5. Expected Benefits

The provision of support for life cycle activities (via the formalism of models and architectures encapsulated within a computer-aided systems engineering environment) can bring about the following business benefits[1]:

- **improved consistency** with respect to the link between *what the manufacturing system is required to perform* and the *way in which it is configured and the components that it requires*. This should be the result of the environment's capability to: (1) describe the requirements of the IMS in a form that can be easily

---

1. This is not meant to be an exhaustive list of benefits. Additionally, the benefits are stated as compared to current practice (as discussed later in this thesis).

communicated among the people involved in realising such requirements (i.e. via conceptual analysis models and by means of adequate modelling languages); (2) map between *conceptual analysis models*, which will be manipulated when the enterprise is being modelled, and *actions that are performed by the system components*, once the enterprise system is being used; (3) support the specification of adequate components, which will provide the system functionality identified in the conceptual models; and (4) describe the interactions (required by the system functionality) amongst system components (i.e. shaded area in Figure 3, which illustrates formalisms of interest in this research).

- **reduced system failure** (resulting from design errors) and **improved traceability of design decisions**. These benefits should result from the capability of the environment to facilitate analysis and simulation on the same models at various levels of abstractions, thereby enabling decisions to be made on alternative design issues (e.g. alternative system configurations), before the system is deployed.
- **shorter system design-to-build lead-times**, due to the environment's capability to rapidly generate prototypes of the system, this including prototypes of system components, which can provide means of rapidly realising alternative solutions to system requirements.
- **lower costs**, due to the environment's capability of re-use knowledge and experience acquired when engineering previous systems, through re-use of models, and system components and configurations.
- **improved flexibility**, due to the environment's capability to separate integration issues from issues connected with specifics of system functionality (i.e. the actual work that the system performs), this by capturing integration issues within models that can be easily changed and specifying system functionality as modular components that can be supplied by the market on an 'off-the-shelf' basis.

Chapter 2 - Literature Review

Chapter 1 identified the need to associate formalism to the IMS life-cycle, as well as to enable the application of such a formalism to manufacturing system design through adequate CASSE environments. The state-of-the-art associated with research activities aimed at addressing this need is related to two main areas:

- efforts on the definition of models and architectures which can lend formalism to the IMS life cycle;
- initiatives on the application of such formalism to contemporary manufacturing systems integration problems, through case studies applications of CASSE environments.

The study presented in this chapter covers the evolution of research effort within these two main areas, with particular emphasis upon their evolution within the period of this research (i.e. over the last three years).

2.1. Concepts and Definitions

Before introducing the architectures, it is important to clarify the definitions of the terms adopted in this thesis. Where possible the recommendation of the DISC "Framework for User Requirements" [Millis 1992] will be followed, namely:

> "It is fruitless to redefine terms in common use, but it is helpful to say what the usage is in a particular document."

The need to adopt a selection of definitions stems from the fact that nearly every architecture (referred to below) uses a different term to designate similar concepts or the same terms for different concepts. One shall find terms such as: architectures and reference architectures, models and reference models, frameworks and modelling frameworks, modelling methodologies, modelling methods and modelling techniques. Rather than trying to use every author's view of what his model is, this thesis has adopted those definitions that appear to be more widely accepted.

As presented in Chapter 1, according the IFIP/IFAC Task Force [Williams 1993], an enterprise **reference architecture**

> "...**models the whole life history of an enterprise integration project** [i.e. the IMS life cycle] from its initial concept in the eyes of the entrepreneurs who initially developed it, through its definition, functional design or specification, detailed design, physical implementation or construction, and finally operation

> to obsolescence [i.e. the mapping across life cycle phases]. The architecture becomes a [...] framework upon which all of the functions and activities involved in the aforementioned phases of the life of the enterprise integration project can be mapped."

Where a **framework** is commonly used to mean [Fowler 1964] a

> "structure upon or into which casing or contents can be put".

The DISC "Framework for User Requirements" [Millis 1992] extends such a definition by stating that a **framework** is

> "a **structured collection of concepts and their relationships** which scopes a subject and enables the partitioning and relationship of the topics relevant to that subject to be expressed by a common means of description".

According to the ISO/IEC reference model for open distributed processing (ODP) [ISO/IEC 1993], a **modelling technique** (which in this thesis will also be used as a synonym for **modelling method**)

> "is any **technique used to construct the supporting model for a body of specification**. As such, each modelling technique corresponds to a language".

whereby a **modelling language** is

> "**a graphical or textual formalism** which includes both a semantic model and a syntax".

or, more generally [Ostler 1991],

> "a method or style of expression; [a] system of symbols and rules [...]".

A National Science Foundation study on design theory and methodology (from [Boldyreff 1994]) defined (design) **methodology** as

> "the **collection of procedures**, tools and techniques that the designer can use in applying theory to design".

whereby a (design) **theory** consist of

> "systematic statements of **principles and experimentally verified relationships** that explain the [modelling] process and provide the fundamental understanding necessary to create useful methodology [for design]".

ODP defines **theory** as

> "axioms to express the **constraints** of a particular **logical system**".

A **reference model** or **enterprise model** is viewed by IFIP/IFAC Task Force [Bernus 1994] as

> "**prototypical models** which can subsequently modified to fit a particular case; generic (abstract) models **capturing commonalities** but leaving out specific details (i.e. 'fill-in-the-blank')".

This definition of enterprise model maps into what ESPRIT/AMICE refers to as partial models [ESPRIT/AMICE 1991b] (i.e. partially instantiated descriptions of types of enterprises).

This research will seek to use these terms according to definitions presented here. Where a generic designation embracing more than one of these definitions is required, the terms **architecture** or **reference architecture** will be used as a more general concept.

2.2. The Formalism of Architectures

Reference models, architectures, frameworks, methods, techniques, tools, models or any other modelling artifices are commonly used to reduce the complexity of the **engineering** of the integrated enterprise [Bernus 1994].

The multi-disciplinary nature of such an engineering process accounts for the variety of perspectives based on which architectures stem, involving, for example: the perspective of a systems analyst, designer and integrator; the perspective of the enterprise management; the perspective of IT suppliers, the perspective of researchers investigating the issues of formalism from a variety of viewpoints; and the perspective of standardisation bodies (such as ISO, IEC and CEN) and professional societies and groups (such as IFIP/IFAC and OMG) which attempt to analyse, harmonise and organise proposals flowing from the other perspectives.

The description presented in this section attempts to classify works developed in this area largely based on their perspectives. However, such a classification should not be viewed as definitive, but rather as an attempt to structure the information presented in this chapter.

2.2.1. Early architectures

Early work on the definition of models and architectures for CIM started in the 1970's. Initiatives stemming from distinct perspectives of what such formalism should embrace are manifest in frameworks which proposed models based on: (1) the organisation of production control functions of an enterprise as a hierarchy of controllers; (2) attempts to define models describing the main functions of an enterprise system and the inter-relationships among such functions; and (3) methodologies to guide the design of a particular enterprise system.

Exemplars of such architectures are[1]: the **NIST/MSI** architecture [Jones 1989] and **DEC/Philips** model [DEC 1987] [Biemans 1990] for production management; models produced within one of the first ESPRIT project in CIM methodology in Europe: "Design Rules for CIM Systems" [Yeomans 1984] (**Yeomans**); an early **DIN** initiative on the definition of interfaces for CIM [DIN 1988]; **Purdue CIM reference model** [Williams 1989]; early **CAM-I** (Computer Aided Manufacturing International) [Boykin 1990] work on reference models for CIM; early work within the **ISO TC184 SC5 WG1** on "reference models for manufacturing standards" [Graefe 1989] focusing on the interactions among major flows of information, material and control within a CIM system; the CIM model proposed by **Thacker** [Thacker 1989], as an enhancement of the **SME** (Society of Manufacturing Engineers) wheel; and early methodologies adopted by consulting firms and systems integrators (e.g. **CIM-Plan** [Hales 1989] [Hales 1990]).

An integral part of early efforts on the definition of architectures are those produced by major IT suppliers as a means of structuring the services and functions of a CIM system which are encapsulated by the products supplied by them. These architectures usually define how a system can be constructed with single-vendor IT solutions. Examples of suppliers that provided some definition of architecture are IBM, DEC and SIEMENS [Rembold 1991] [DEC 1991].

These early architectures focus on identifying (or defining) generic elements of functionality, interfaces and structure of a CIM system. They usually provide a descriptive model of how CIM functions should be organised. This model can be viewed as a collection of static descriptions for reference in CIM systems engineering.

---

1. For the sake of conciseness, later references to these architectures will use the term that appears in bold font in the text.

They are useful in as much as they provide an insight into components of technology required to build a complete CIM system.

Among these early architectures, the Purdue CIM reference model (which evolved into the Purdue enterprise reference architecture, or **PERA** [Williams 1994]) is still one of the most prominent. It provides useful guidance for the activities along a greater span of the life cycle. Arguably, PERA is one of the first instances of a large scope methodology for CIM implementation.

2.2.2. Modelling languages and architectures for CIM

Another perspective started to emerge with projects which focused on defining modelling languages and architectures for guiding the CIM systems engineering processes. Important representatives of such a perspective are:

- the **IDEF** architecture (U. S. Air Force ICAM - Integrated Computer Aided Manufacturing DEFinition) which provides methods for describing a number of modelling aspects (or views) of a CIM system, namely: function (IDEF0 [Bravoco 1985a]), information (IDEF1 [Bravoco 1985b]), data (IDEF1X), and dynamics (or simulation - IDEF2 [Bravoco 1985c]). IDEF has recently been extended with the addition of a super-set of the **Zachmann** framework[1] [Zachmann 1986] [Zachmann 1987], and a second set of methods, which are under development, namely: process description capture (IDEF3), object-oriented design (IDEF4), ontology description capture (IDEF5), design rationale capture (IDEF6), user interface modelling (IDEF8), scenario-driven information system design specification (IDEF9), implementation architecture modelling (IDEF10), information artifact modelling (IDEF11), organisation modelling (IDEF12), three schema mapping design (IDEF13) and network design (IDEF14) [Mayer 1991] [Mayer 1992].

- a method and framework proposed by the **GRAI** Laboratory of Bordeaux University [Maloubier 1985] [GRAI 1984] [Akif 1991] to analyse and design production management systems (i.e. GIM - GRAI integration methodology). GRAI includes GRAI grid which relates planning horizons within a company with the company's functions involved, their intersections being referred to as decision centres; and GRAI nets which can be used to describe the flow of information/ material and the flow of decisions within a decision centre. GIM focuses on

---

1. The Zachman framework [Sowa 1992] relates (on a grid) the levels of description against different description perspectives based on which the system is modelled. The levels of description (which resemble the life cycle stages) include: scope, enterprise model, system model, technology model, component and functioning system. The description perspectives (which relate to modelling views) include: data, function, network, people, time and motivation.

decision flow and adopts IDEF to model the remaining aspects of a system [Doumeingts 1992].

- the **CIM-OSA** (Open Systems Architecture for CIM) architecture proposed by the ESPRIT/AMICE consortium [ESPRIT/AMICE 1991b], which is now being considered as a candidate to a European standard [CEN 1994a]. CIM-OSA proposed a modelling framework for model-building [Kosanke 1992] which embraces the definition of the modelling constructs required for modelling four views (i.e. function, information, resource and organisation) [Jorysz 1990a] [Jorysz 1990b] [Russel 1991] [Vernadat 1992], along three modelling levels or stages (i.e. requirements definition, design specification and implementation description) based on three levels of generality or detail (i.e. generic, partial and particular). CIM-OSA also provides the specification of an integrating infrastructure for model execution [Querenet 1991].

- the combination of a process-oriented description akin to IDEF0 (but modelling flows of orders, information and resources) and object-oriented design (i.e. MOOD), proposed by the IPK institute in Germany [Mertins 1992], which has evolved into a candidate to DIN standard for enterprise modelling, as IEM (i.e. Integrated Enterprise Modelling) [CEN 1994b], under the umbrella of QCIM [Pirron 1994].

- the **ARIS** (Reference Architecture for Information Systems) proposed by Scheer [Scheer 1992] which evolved from his work on information systems design [Scheer 1991]. ARIS defines the constructs of a modelling language and a modelling framework akin to CIM-OSA, which embraces three modelling levels and four modelling aspects, namely: information, organisation, resource and control.

A basic underlying thrust of the perspective represented by IDEF, GRAI and CIM-OSA is that no unique archetype description or model (i.e. reference model) can be obtained which can be generally applied to any enterprise. The complexity involved in the integration process requires rather the application of languages which should provide adequate constructs to describe the relevant aspects of a CIM system. These descriptions can then be generalised to construct reference models. The result of the application of such constructs (in order to create reference models) can then be used as a reference for later designs (i.e. for modelling a particular enterprise through a process called instantiation).

Among these architectures, CIM-OSA is, arguably, one of the first to propose a framework for integrating (into an organised whole) the perspectives presented early in the section, particularly those of users and suppliers of IT components.

2.2.3. Methods and techniques from software analysis and design

One of the basis upon which modelling languages were proposed is the software analysis and design methods, which were themselves applied as languages for designing CIM systems, notably:

- process oriented methods, such as: **Yourdon**/De Marco and **SSADM** [Longworth 1992] [Yourdon 1989] [Vervoort 1988] [Weymont 1987] [Maji 1988].
- Object-oriented design methods [Booch 1991] [Adiga 1993] [Bailin 1989] [Mize 1992] [Cox 1986] [Jochem 1989] [Schiel 1990] (A review of the object-oriented methods from the viewpoint of their application to manufacturing is presented by Nof [Nof 1994].).
- **Petri-nets**-based methods and their various extensions [Peterson 1981] [Boucher 1991] [Devapriya 1991] [Farines 1992] [Garnousset 1989] [Harhalakis 1989] [Hatono 1989] [Tonshoff 1989] [Tzafestas 1989] [Bonney 1993] (A review of Petri-nets and its extensions is presented by David [David 1994].).
- Data engineering methodologies [Verheijen 1982] [Leva 1987] [Hsu 1990] [Jain 1990] [Fritisch 1989] [Carswell 1987] [Boyle 1991] [Blinn 1991] which adopted and extended the pioneering work of Chen [Chen 1976] on a language for data modelling (A comprehensive review of data engineering methods is developed by Hars [Hars 1991] within the scope of the ESPRIT/CODE project.).

2.2.4. Artificial intelligence and federated architectures

A contrasting view of enterprise modelling to that embodied in architectures such as CIM-OSA is expressed by the artificial intelligence community view of CIM, represented by federated architectures, such as: PACT/SHADE [Pan 1991] [Tenembaum 1992], SIRIUS-BETA [Goranson 1992] and architectures for semantic unification, such as: CARNOT [Huhns 1992], SUMM [Fulton 1992] and TOVE [Fox 1992].

PACT/SHADE is one the most prominent initiatives which seeks to define an environment where intelligent agents cooperate through sharing knowledge, thus enabling coordination of interaction among people as well as software interoperability [Tenembaum 1992] (with particular emphasis to supporting product design environments).

CIM-OSA, IDEF, GRAI as well as early reference models are mostly descriptions or means of describing systems via the application of a homogeneous modelling language. The federated architectures embody a 'bottom-up' view of integration. In such a view, no attempt is made to construct a system from models created through the application of a unique language. Components, described in a

variety of heterogeneous modelling languages, are added to the system incrementally. Each component is a new agent which brings in knowledge and uses knowledge already available in the system. It is generally trusted in these architectures that [Petrie 1992a] "...models play a unique role in the enterprise integration equation [in such a way] that the technical leverage for enterprise integration will be found primarily through considering issues of model integration...". Thus, architectures such as CARNOT and SUMM provide alternative means of integrating models (mostly information models) described in different modelling languages, in order to achieve integration across heterogeneous applications. SUMM is based on the semantic unification of models and CARNOT on a global ontology[1] shared by the models being integrated.

2.2.5. Frameworks for integrating infrastructures

An additional perspective into the issues of integration is the definition of frameworks for the organisation of IT infrastructural services and functions which can be generally applied to any manufacturing system or indeed any system. In simple terms, such an infrastructure should provide a level of IT services, so as to enable the integration of system components (see Figure 1.b) in such a way that the details of physical integration (see Figure 1) are encapsulated by the integrating infrastructure, thereby hidden from the system components. Hence, components can be developed independently from integration issues, thus enabling the configuration of a system based on 'off-the-shelf' components.

Pioneering academic initiatives in this area are: **CONIC** [Kramer 1990a] [Kramer 1990b] [Magee 1989] and the **CIM-BIOSYS** integrating infrastructure [Weston 1993] (the latter being particularly oriented to manufacturing systems integration). A review of research works in the area of distributed computing is presented in the conference proceedings edited by Kramer [Kramer 1992] and Meer [Meer 1992]. A review of the main features of some proprietary integrating infrastructures is presented by AMR [AMR 1991], which includes: DEC NAS and BASEstar [DEC 1992], HP-OSF/DCE [Boswell 1992], IBM-DAE/Plantworks, ITP MainStream and Savoir FLEXIS. Goodwin [Goodwin 1994] lists a number of products that are emerging in the market which he classifies as 'middleware', which provide an operating system independent environment for applications to inter-operate and to

---

1. A global ontology is used to yield the appearance and effect of homogeneity among existing models. Here, an ontology is viewed as consisting basically of a kind of knowledge base which maps the semantic differences between models [Huhns 1992]. According to Fox [Fox 1994], ontology is a formal description of entities and their properties; it forms a shared terminology for the objects of interest in a domain, along with definitions for the meaning of each of the terms.

access data transparently (e.g. ICL-DAIS [ICL 1994]). Initiatives aiming at defining commonly agreed services for an integrating infrastructure which may impact the definition of such services are the Open Software Foundation (OSF) [Johnson 1991], the ESPRIT CCE-CNMA project [Pleinevaux 1994], and Object Management Group (OMG) [OMG 1991]. According to Brenner [Brenner 1987], the leading distributed computing architectures are: OSF-DCE, UI-ATLAS framework, The X/Open XDCS framework, OMG-CORBA (Common Object Request Broker Architecture), and ISO-ODP.

Among these initiatives, OMG, which stems from the application of object-oriented paradigm in distributed computing, has defined CORBA [OMG 1991], embracing a language for integration of distributed software objects (i.e. IDL) which interact with one another in order to perform a certain task. The basic underlying objective behind CORBA is to define the specification of an Object Request Broker which can serve as an enabling mechanism for integration of 'off-the-shelf', multi-vendor software components. Such components are envisaged to be marketed as CORBA compliant software objects built regardless of the details of the platforms, operating systems and networking infrastructures on which they run.

Distributed computing in itself is an area in which a large diversity of work exists which this research is far from mastering. Thence, apart from the overview given above, the interest of this research in distributed computing stems basically from the impact that integrating infrastructures can have on the definition of architectures for enterprise integration. This is strongly emphasised on the standardisation front with the reference model for open distributed processing [Brenner 1987] [Kourie 1989] [Kobayashi 1990] [Linigton 1991] [Hutchison 1991] (RM-ODP).

The RM-ODP is an on-going initiative to define a framework for the standardisation of open distributed processing [ISO/IEC 1993], by

> "creating an architecture within which support [for] distribution inter-working and portability can be integrated".

The primary objective of the RM-ODP [ISO/IEC 1993]

> "is to provide a framework which enables programmers to construct distributed applications, without having to take [into] account [...] the potential diversity of hardware, operating systems and communications mechanisms in the underlying computer network. The ISO work on ODP is about the standardisation of ODP architecture and interfaces, so that

systems can be constructed from heterogeneous and re-usable components."

An ODP system is a system which can be "specified by using a viewpoint language. To deal with the complexity of an ODP system, the system is considered from different viewpoints" [ISO/IEC 1993] each of which represents a different abstraction of the original system. The viewpoints defined by ODP[1] are:

- **enterprise viewpoint**, for the "expression of purpose, policy and boundary";
- **information viewpoint**, for the "expression of information and information processing functions in a distributed system";
- **computational viewpoint**, for the "expression of the functional decomposition of an ODP system and of the inter-working and portability of ODP";
- **engineering viewpoint**, for the "expression of the infrastructure required to support distributed processing";
- **technology viewpoint**, for the "expression of the suitability of technologies to support aspects of ODP".

According to ODP [ISO/IEC 1993], "any existing language can, in principle be used for specification of a system from a particular viewpoint, provided that those specifications can be interpreted in terms of relevant viewpoint concepts". For that matter, ODP specifies general requirements for each viewpoint language and strongly recommends the use of an object-oriented basis.

By providing such a framework, the RM-ODP aims to support the generation of the following categories of standards through the application of the viewpoint languages (see Figure 6 [ISO/IEC 1993]): specific reference models for certain types of enterprises, and standards for the realisation of common functions (such as generic applications) and specific functions (such as specific applications).

The framework shown in Figure 6 is similar to the key concept embodied in the CIM-OSA cube which relates its views, modelling levels and levels of generality (see discussion later in this thesis). In either case the fundamental aim is to enable the convergence of interests of users and suppliers of system components around a common architectural understanding of the problem domain that the models are

---

1. According to ODP [ISO/IEC 1993], "the most significant of the ODP viewpoints from an OSI [i.e. Open Systems Interconnection - ISO 1990] perspective have been identified as the information, computational and engineering viewpoints".

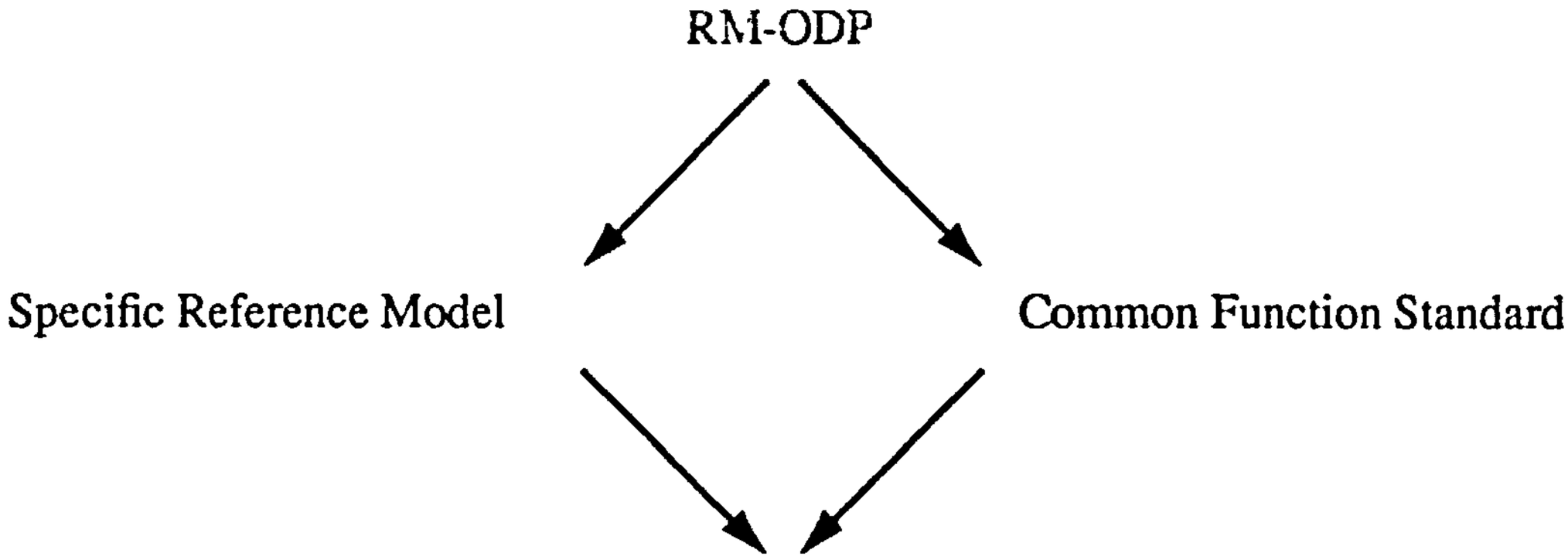


Figure 6 - ODP support for standardisation

representing.

The viewpoints can also be mapped into the traditional phases of system development (assuming that the ODP system is implemented from 'scratch'), namely [ISO/IEC 1993]:

- Requirements analysis (i.e. part of the strategy definition stage in Figure 3): enterprise viewpoint;
- Functional specification (i.e. part of the conceptual analysis stage in Figure 3): information and computational viewpoints;
- Design: engineering viewpoint;
- Implementation: technology viewpoint.

2.2.6. Consulting practitioners methods and OPENframework

The main difficulty with analysing methods used by consulting practitioners and systems integrators is to gain access to information. Information on methods is kept in confidence and is usually made accessible only through a training process. During the course of this research, the only method to which the author had access (through a training course) was the OPENframework architecture [ICL/OFD 1994]

OPENframework emerged as a result of an effort initiated by a supplier of computers and software (i.e. International Computers Limited - ICL) [Brunt 1992] and is now becoming a method which is being applied on a world-wide scale. OPENframework started as a vendor-independent technical architecture (i.e. a specification of how the IT systems of an enterprise should be organised), embracing the definition of:

- four **perspectives** associated with the description of an information system, namely user, service provider, application developer and enterprise manager [Brunt 1992];

- five **qualities** which serve as metrics for qualifying technical architectures developed for particular industries, namely: availability [Smethurst 1993], usability [Hutt 1993a], performance [Sutcliffe 1993], security [Faithorne 1993] and potential for change [Pratten 1993];
- seven **building blocks** which encapsulate the concerns that are manifest in any technical architecture, namely: user interface [Hutt 1993b], application architecture, distributed application services [Brenner 1993], information management [Kay 1993], application development [Brown 1993], systems management [Gale 1993], networking services [Deignan 1993] and platforms [McVitie 1993].

Over the last two years, OPENframework evolved a business architecture which embraces the definition of [ICL/OFD 1994]:

- four **perspectives** associated with the description of a business enterprise, namely: customer, employees, trading partner and enterprise management;
- five **qualities** named as in the technical architecture, but applied as qualifiers for the business architectures;
- seven **building blocks** which encapsulate the concerns that are manifest in any business architecture, namely: strategy, organisation, process, information, systems, assets and appraisal.

In addition to defining the content of perspectives, qualities and building blocks as major elements for building business and technical architectures, OPENframework has also introduced a set of methods [ICL/OFD 1993] for organising the information related to these elements for a particular enterprise. Such methods support the description of the three major systems of an enterprise, which OPENframework defines as being: business system, social system and technical system. These descriptions aim to cover the complete IMS life cycle leading to the definition of a system configuration in terms of system components adequate to the enterprise under consideration.

A positive point of this architecture is its orientation towards aggregating most of the perspectives previously mentioned in an organised whole. However, as in other methods adopted by consultants and systems integrators, OPENframework is limited to provide general guidelines as to how the process of developing architectures should be performed (i.e. a methodology), without actually defining a modelling language or an actual architecture (in the sense used in this thesis). Much of the knowledge required to apply the methods as a coherent whole must be supplied by the practitioners who use them.

2.2.7. Methods and tools for business analysis

From the perspective of strategic management, a manufacturing enterprise is essentially a business.

With respect to the IMS life cycle, this perspective includes methods and techniques which can be used to support enterprise engineering at the upper most level of abstraction (i.e. the strategy definition stage shown in Figure 3). This includes the definition of a business strategy which can serve as the basis upon which the activities of the remaining life cycle stages can be performed.

Examples of methods and techniques which comprise *current practice* in business analysis (as defined by Stacey [Stacey 1993])[1] include: PEST analysis [Johnson 1993], Boston Consulting Group (BCG) matrix, Shell directional policy matrix, Porter's five forces, value chain and generic strategies [Porter 1985], strategic group analysis [Johnson 1993], SWOT analysis, costing methods [Shararoun 1994], Miller's configurations [Miller 1983] [Davis 1991], Beer's Viable System Model [Espejo 1989] and the 7S framework [Stacey 1993]. Another well-known methodology which is usually associated with business analysis, but not limited to, is the soft-systems methodology proposed by Checkland [Checkland 1981] to address ill-defined problems.

Stacey [Stacey 1993] analyses the current practice, its underlying assumptions and postulates that much of what it advocates cannot be applied to organisations working in a state of "bounded instability" (i.e. chaos). This a state on the boundary between complete disorder and order, in which organisations are very creative and innovative thereby becoming models of success. States of complete disorder or order lead, respectively, to disintegration or stagnation ultimately leading to failure. According to Stacey [Stacey 1993],

> "organisations must operate in states of chaos if they are to transform themselves and that the process of transformation is a spontaneous self-organising one. These properties lead us to see that the conventional wisdom on strategic management is a limited special case applicable only to the short-term control of an organisation or to the strategic development of organisations required simply to repeat their past. In all other cases the conventional wisdom cannot apply and thinking in those terms is a harmful fantasy escape from reality. Instead we need to think of an organisation as a learning community out of which new strategic directions may emerge [spontaneously]."

---

1. Stacey refers to "current practice" as the "conventional wisdom" in strategic management.

In association with this line of thought, business process analysis and re-engineering (BPR) have been the focus of research initiatives aiming at improving business performance from a process perspective. Business re-engineering as defined by Hammer [Hammer 1994] is a step change in the way the business is currently working. This involves a view to completely re-defining business processes as if the organisation were starting from 'scratch'. One of the primary focuses in BPR is upon improving the enterprise performance through re-organising it from a process perspective, one of its primary focus being the role of human beings in an IT context [Davenport 1994].

From a modelling perspective, BPR can be understood as the first level of analysis shown in Figure 1. At this level, the complete enterprise system is analysed at a high level of abstraction with a focus on the way in which its sub-systems inter-operate.

An extensive review and comparison of methods and techniques for computer-aided production management, which includes methods for business analysis, is presented by Frizelle [Frizelle 1991]. A review of the methods and techniques mentioned in this section was also developed by the author outside the scope of this thesis, as part of a proposal for further research projects [Aguiar 1994m].

Finally, the DISC "Framework for User Requirements" [Millis 1992] (in the context of the "framework of frameworks") is one of the few initiatives which, in its review of existing architectures, lists some of the methods and techniques for business analysis as part of a wide scope framework for enterprise engineering. Most reviews of existing architectures do not include these methods and techniques.

2.2.8. Other models and architectures

Nearly every CIM project involves some level of definition of architectural concepts which has led to the appearance of a variety of other models. A review of some of these models, some of which related to the architectures discussed previously, was developed by Doumeingts [Doumeingts 1992] chiefly based on work presented at the CIM-CON conference [Jones 1990] and on projects developed within the ESPRIT program. This review included the following additional architectures: (1) models for shop and cell control levels, such as: MMCS (Manufacturing Management Control System) developed within the ESPRIT Project 418 - Open CAM System; the PAC (Production Activity Control) model developed within the ESPRIT Project 477 - COMISA (Control system for Integrated Manufacturing); (2) the model for factory supervision being developed in the ESPRIT Project 2434 - Real-Time Controllers for

Distributed Factory Supervision; (3) the model of IMPACS (Integrated Manufacturing Planning And Control Systems) - ESPRIT Project 2338; (4) models developed by the ESPRIT Project on factories of the future (one-of-a-kind production techniques); (5) the methodology developed by the ESPRIT Project 2706 - MICIM (Methodology for the Introduction of CIM); (6) the methodology for economic and technical evaluation of various options of CIM solutions introduced in ESPRIT Project CIMSIM; (7) the CIM architecture of the Sandia National Laboratories [Yoder 1990]; and (8) the RAMP (Rapid Acquisition of Manufactured Parts) architecture [Litt 1990].

2.3. Literature Surveys of Architectures

A number of reviews of architectures has been developed over the last ten years by various authors, notably: Parunak [Parunak 1987], Rembold [Rembold 1991], Maji [Maji 1988], Mayer [Mayer 1991], Wiendahl [Wiendahl 1991], Rogers [Rogers 1989], Frizelle [Frizelle 1991], Bohms [Bohms 1990], Mertins [Mertins 1992], Doumeingts [Doumeingts 1992]; the IFIP/IFAC Task Force [Williams 1993] and the ManTech-funded Enterprise Integration Program (EIP) [Petrie 1992a].

Most of the above comparisons focus on the descriptive power of architectures. They are mainly interested on how complete an architecture is in order to be used to represent a system. The four latter reviews analyse architectures for their capability to support the IMS life cycle, as defined in Chapter 1. These reviews vary in the methodology that they adopt for their analysis as well as in the final outcome of the analysis. However, in most of these studies three main observations stand-out:

- no single architecture can provide the required support for all the issues involved in the IMS life cycle;
- CIM-OSA stands out as one of the most comprehensive;
- an architecture resulting from a combination of the best features of the most outstanding architectures is desirable.

The results of these reviews were confirmed by a study developed by the author, as part of an early review of the literature on architectures undertaken within the scope of the Model-Driven CIM project [Aguiar 1992d] [Aguiar 1993e] [Aguiar 1993c] [Aguiar 1994m]. This review considered the architectures known to the author at the time of the survey and compared them against a number of parameters.

Among these surveys, one of the most important studies was conducted within the Enterprise Integration Program, funded by ManTech. This study started in 1990 and

resulted in a consensus on a conceptual framework for enterprise integration. The framework was used as the basis for discussion within a number of workshops that culminated with the realization of the First ICEIMT (International Conference on Enterprise Integration Modelling Technology) [Petrie 1992a] in which various more recent architectures were discussed. A number of architectural definitions were proposed from the ICEIMT's workshops.

One of the most important definitions was a meta-model for integration, based on five components, namely: application architecture (i.e. the components of an enterprise system), execution environment (i.e. integrating infrastructure), enterprise characterization (i.e. models of an enterprise system), formal mechanisms (for binding the three other components) and integration domain (where these four components are actually integrated). Such a meta-model was then applied to evaluate what existing architectures provide. One important outcome of such an application was the fact that CIM-OSA could be directly associated with the separation of concerns implied by the five components of this meta-model.

The EIP also proposed three types of approaches to the problem of syntactic and semantic model integration, namely [Petrie 1992a]: master model (e.g. ESPRIT/CODE Project [Hars 1991]); unified model (e.g. CIM-OSA [ESPRIT/AMICE 1991b]) and federated model (e.g. CARNOT [Huhns 1992]). An analysis of these approaches resulted that the two latter ones stand out as the most promising. However, between them a number of trade-offs stemming from their mutual advantages and disadvantages were encountered which resulted in the recommendation that an amalgamation of these two would be desirable.

2.4. Amalgamation Efforts

Exemplars of efforts on the combination of some of the architectures previously presented are:

- the manufacturing systems integration methodology (MSI-UK) proposed by an association of research groups in the UK [Carrie 1993], which combines Beer's Viable System Model [Espejo1989], Miller's configurations [Miller 1989], GRAI grid [Doumeingts 1992], and data-flow diagrams [Yourdon 1989].
- a number of combinations of IDEF0 and Petri-nets [Boucher 1990] [Meta 1990] [Meta 1989]. An outstanding example of such combinations is Design/IDEF and Design/CPN [Meta 1990], tools developed by MetaCASE Technology which combine IDEF0 for functional modelling associated with coloured Petri-nets for behavioural modelling.

2.5. Standardisation Efforts

The main standardisation efforts are:

- the ISO TC184 SC5 WG1 - CD 14258, [ISO 1994] "Framework for Enterprise Modelling", recently distributed for ballot. This document basically issues general guidelines for enterprise modelling.
- the CEN TC 310/WG1 - ENV 40003, "Framework for Enterprise Modelling", which is defining requirements for enterprise model execution and integration services (i.e. an integrating infrastructure); and a suite of constructs to be used for enterprise modelling. These two fronts of work make strong reference to the CIM-OSA specifications, although they do not adopt CIM-OSA in its entirety.
- the DISC "Framework for User Requirements", developed by the Frameworks group of the DISC Business Strategy Forum, which is now a BSI (i.e. British Standard) draft for development, and is attracting interest and support worldwide. This work was the main starting point for the framework of frameworks [Millis 1992].
- the momentum that the architecture for business process modelling proposed by QCIM (i.e. the integrated enterprise modelling method, associated with QCIM [Mertins 1994]) is gaining within a German standardisation initiative (i.e. DIN).

2.6. Harmonisation Efforts

Two main harmonisation efforts worth noting are:

- the mapping between the constructs used in QCIM and CIM-OSA [CEN 1994b].
- the effort invested by the IFIP/IFAC Task Force in devising a framework to classify architectures (i.e. GERAM: Generic Enterprise Reference Architecture and Methodology [Bernus 1994]) and applying it to major architectures.

In regard to the latter effort, the IFIP/IFAC Task Force have found

> "that only three of the many architectures known to [them] were suitable [...]. These were CIMOSA, the GRAI-GIM Methodology, and the Purdue Enterprise Reference Architecture and its associated Purdue Methodology."

The IFIP/IFAC Task Force has also recently included TOVE [Fox 1992] as another suitable architecture. TOVE (TOronto Virtual Enterprise) aims to create generic representations of enterprise knowledge (i.e. ontology) which can be reused

across a variety of enterprises [Fox 1993].

2.7. Tool Development and Application

Three years ago (when this research started) most architectures were at a state of specification with few architectures implemented as usable tools. Some of the most commonly available tools were: CASE tools based on methods for software development; IDEF-based modelling tools and Petri-net-based tools.

Examples of early attempts to encapsulate the formalism of some of the architectures are:

- Computer-Aided GRAI [Akif 1991], a tool to support the creation of GRAI grids and GRAI nets for a particular enterprise;
- CIM-OSA Demonstration [Emond 1988], an early attempt to build a tool to partially support the creation of function view models at the requirements definition modelling level, which evolved into the CIM-OSA Computer-Aided Enterprise Engineering (CAEE) tool [ESPRIT/AMICE 1991f]. This was the result of an attempt to build a CASE tool to fully support the creation of function view models at the requirements definition modelling level.

Most of these tools covered a narrow scope of the IMS life cycle, few of which extended further from supporting model building. Outstanding examples of tools that supported a level of model-enactment[1] (usually related to a simulation capability), are: Design/IDEF and Design/CPN [META 1990]; UNISSON [Bonney 1992] which supported simulation and control of a system modelled on a Petri-net-based language; and tools based on traditional discrete-event simulation languages [Pidd 1992].

Examples of early attempts to apply these architectures for modelling manufacturing domains (some of which without the benefit of modelling tools) are:

- a case study on the application of CIM-OSA to a shop-floor domain within Aerospatiale [ESPRIT/AMICE 1991d]. This was chiefly limited to applying the constructs of function view at the requirements definition modelling level, and later extended to fragments of models for the three other views of CIM-OSA. Based on the models created in this case study, a number of fragments of models have been extracted as examples used throughout the CIM-OSA specification.
- IDEF models of a number of industrial applications [Colquhounn 1991]

1. Model-enactment is used here to denote the process of executing a model description in a computer.

[Jayaraman 1990] [Malhotra 1990] [Marechal 1987] [Ranky 1991a] [Ranky 1991b] [Sarkis 1994] (a review of the state-of-the-art of IDEF0 is presented by Colquhoun [Colquhoun 1994]).

- Petri-net models of a number of applications, but particularly shop-floor control applications [Boucher 1991] [Devapriya 1991] [Farines 1992] [Garnousset 1989] [Harhalakis 1989] [Hatono 1989] [Tonshoff 1989] [Tzafestas 1989].

When this research started, no tool or application known to the author supported the IMS life cycle so that the same models could be used for modelling, analysis, simulation, rapid-prototyping of code for the system structure and its components, configuration and operation of the deployed system. However, efforts in this direction have been invested in a number of research projects, notably:

a. In the ESPRIT/AMICE - CIM-OSA project which ended in February 1994 with a workshop [Kosanke 1994a] whereby the main deliverables of the project were demonstrated. Two outstanding deliverables were:

- the pilot implementation of CIM-OSA developed by WZL-Aachen [Katzy 1993] which consisted of: (1) using TeamWork (a CASE tool supplied by HP) to model a mini-factory built in their laboratories. The modelling exercise supported by this CASE tool was formalised in data-flow diagrams, state-transition diagrams and entity-relationship-attribute diagrams as a means of partially emulating the constructs of CIM-OSA; (2) applying a simulation tool developed 'in-house' to execute simulation on the models; (3) using code generated by the CASE tool to drive the operation of the mini-factory.

Although this pilot-implementation is certainly the most embracing demonstration of CIM-OSA, it presents the following limitations:

- One of the most outstanding features of CIM-OSA is its modelling constructs (i.e. its modelling language). Although the pilot-implementation makes reference to the CIM-OSA modelling framework, it does not adopt its modelling language. The exercise of model-building and model-enactment consists basically of generating a software application from a Yourdon-based model (re-interpreted as if it were a CIM-OSA model) which controls the interactions among the components in the mini-factory, thus playing the role of a cell controller.
- The tools, models and systems used in the implementation were built to demonstrate the concepts of CIM-OSA rather than to provide solutions in themselves in regard to support for the IMS life cycle.

- The pilot-implementation adopted HP-DCE as the integrating infrastructure which does not provide the complete functionality required in the integrating infrastructure of CIM-OSA.

- the case-study application of CIM-OSA to model a gear-box shop-floor at FIAT [FIAT 1994a] [FIAT 1994b] [Naccari 1994], which consists of using a Petri-net-based tools in association with an object-oriented tools (i.e. Protob, Artifex and Quid [Bruno 1994]) to model the main processes on the shop-floor.

Although this is arguably one of the most complete case studies on the application of CIM-OSA to a real industrial domain, it presents the following limitations:

- in a similar way to the pilot-implementation, the CIM-OSA modelling language was re-interpreted in the light of the languages supported by the modelling tools adopted;

- the case study covered only the modelling stage.

b. In the ESPRIT-VOICE project[1] [ESPRIT/VOICE 1992] [ESPRIT/VOICE 1993]) whose period of execution partially overlapped with that of this research. VOICE aimed to validate and apply CIM-OSA to a number of industrial domains. The main deliverables that resulted from this project were:

- the McCIM modelling tool [Didic 1992] [Didic 1993] which enables the creation of Petri-net models under the front-end of a CIM-OSA-based graphical representation. These models can be enacted for simulation purposes (using a SmallTalk-based environment) and to drive the interactions governed by a cell controller;

- an effort to develop an integrating infrastructure for CIM-OSA, which has not been completed yet [ESPRIT/VOICE 1992] [ESPRIT/VOICE 1993];

- case study applications of some of the tools in a car manufacturing shop-floor [Didic 1993].

The work of VOICE project is focusing on providing means of integrating and controlling machines on a shop-floor based on the application of CIM-OSA. Part of the orientation of this project is to use Petri-nets as the basic language for model-building and model-enactment, and interfacing with other tools available from the market in order to cover the remaining modelling aspects of CIM-OSA.

---

1. Partial results of this project were also demonstrated at the workshops of the CIM-OSA club [Kosanke 1994a] [Kosanke 1994b].

c. At IWI, on the provision of a modelling tool to support the application of the ARIS architecture to capture the requirements of a domain [Scheer 1992]. This tool is limited to support model-building for the four views of ARIS, without actually providing any capability for enacting models.

d. A few others tools and applications developed based on the CIM-OSA modelling framework which, however, support only a limited scope of such a framework [Siemens 1994] [Gaches 1994] [Naeger 1993] (The workshops of the CIM-OSA club congregated most of the information on such tools and applications [Kosanke 1994a] [Kosanke 1994b]).

e. In a number of case studies covering the modelling stage of the life cycle, namely: applications of IDEF0 in association with the emerging IDEF methods, notably IDEF3 [Colquhoun 1994]; the wide use of the GRAI method in France, as claimed by Doumeingts [Doumeingts 1992]; the application of PERA by the industrial members of its consortium, as noted by Williams [Williams 1993]; and the work under development by Naeger [Naeger 1993] on the specification of resource components based on the CIM-OSA constructs.

2.8. Contemporary Practice

In spite of the research effort illustrated in previous sections, contemporary industrial practice in this area is still that of applying the formalism of methods and tools to a narrow scope of activities within the IMS life cycle. A common feature of most of these applications is that they embrace the use of architectures for modelling, sometimes simulation, but rarely analysis, rapid-prototyping, system configuration and system operation in an integrated manner. With the exception of CIM-OSA, no application supports these stages based on the same models. Even the CIM-OSA-based applications share models by simply adopting a strategy of interfacing tools available in the market (in such a manner as to re-interpret the CIM-OSA concepts).

In most cases, tools and methods are used simply to facilitate communication or formalise requirements, or as an aid to decision-making based on simulation. In regard to evolving from modelling and simulation to system generation and execution, work-flow systems, such as ProcessWise [ICL 1993], have been used. These systems provide mechanisms for controlling a certain process in industry based on a model of its major tasks. Typical applications for such systems have been in the automation of office procedures. However, although such systems provide a level of connection between system modelling and system operation, they do not embrace the level of description required to support life cycle processes, as prescribed by the most well recognised architectures (e.g. CIM-OSA, PERA, GRAI, ODP, etc.).

2.9. Summary of the State-of-the-Art

Accurately describing the state-of-the-art in so dynamic an area as this is not an easy task. Devising a research need within this area of work to which a contribution can be made over a period of about three years is even more challenging.

Over the period in question, the scenario has changed. The situation three years ago can be briefly characterised by the following:

- major architectures were generally manifest in the form of partially defined specifications (on paper);
- applications of these architectures to manufacturing were limited to modelling (usually as a paper exercise), and modelling geared towards analysis and simulation. In most cases, a barrier to application was the general lack of understanding about what the architecture could provide or how it could be used;
- no integrating infrastructure was available which could readily be used for model execution, thus leading to a requirement of adapting an existing infrastructure to suit this purpose, this being a task whose viability depended upon the level of openness for change that the supplier of the integrating infrastructure was able to provide;

This scenario has evolved to a situation which can be briefly characterised by the following:

- specifications of the main architectures can be considered as nearly complete, although a number of issues mostly related to achieving implementation and extending their coverage to include the "strategy definition" stage (in some of them) are yet to be defined. This is particularly the case in CIM-OSA [ESPRIT/ AMICE 1993a];
- tools are available which can support part of the activities of the IMS life cycle which implement aspects of an architecture or refer to their main guidelines when 're-vamping' or 'amalgamating' other tools and methods to support such guidelines;
- additional case studies on the application of architectures have been developed which, although in most cases are limited to describing requirements or simply modelling, provide (if nothing else) guidance as to how architectures can be applied;
- progress within the area of integrating infrastructures (such as in OMG-CORBA) may cause a major impact in the way which integrating infrastructures are supplied and software components are marketed. Nevertheless, in the case of OMG for

instance, only recently is a working group being formed to address manufacturing software [Waskiewicz 1994];

- efforts on the amalgamation, harmonisation and standardisation of architectures are still very much under development. The work done so far has re-inforced the need for architectures, sought to use the best features of contemporary architectures, notably in initiatives of the IFIP/IFAC Task Force and CEN.

In spite of the changes incurred in this scenario, the requirements identified in Chapter 1 still persist as an important research need, namely: to encapsulate the formalism of selected architectures in a form that can render them applicable to contemporary IMS engineering processes. The scenario evolution may have even re-inforced such a need.

The state-of-the-art in the area, as discussed in this chapter, has shown that a considerable amount of work has been developed (especially over the last few years). However, no single initiative has provided support to the IMS life cycle by embracing an approach to the encapsulation of architectural concepts into a CASSE environment capable of supporting modelling, analysis, simulation, rapid-prototyping (of code), configuration and operation of an IMS, in an integrated manner.

The emphasis on scenario evolution presented here is important as a background against which some of the decisions which have been made during the course of this research can be judged. Essentially, the manner in which the research need was addressed reflects the evolution of the state-of-the-art over the period in question. Consequently, if some of the decisions on the research methodology had to be made now, the research direction may have been different (e.g. the tools used in the implementation effort may differ).

2.10. Concluding Remarks

The diversity of perspectives presented in this chapter may convey a message of chaos due to the complementary and, sometimes, conflicting perspectives that architectures represent. However, each and every architecture may have something to offer with regard to support to the IMS life cycle. Outstanding features of some of the architectures include:

- a mechanism for aggregating the perspectives of users and suppliers of IT components into a coherent whole, tackling integration from a standardisation viewpoint, offered by architectures, such as CIM-OSA and ODP;
- the comprehensiveness of the CIM-OSA framework, which offers a model for

using modelling technology consistently along a wide span of the IMS life cycle, covering activities from system modelling to system execution and operation;

- the wide-spread acceptance of IDEF methods in industry, as a consequence of their application to a number of case studies;
- the wide-spread availability of tools and applications based on Petri-nets and their extensions;
- mechanisms for enabling reusability of models offered by object-oriented modelling methods;
- support for describing the enterprise from a high-level of abstraction provided by methodologies such as the MSI-UK and GRAI-grid;
- the multi-dimensional view of enterprise modelling, as encapsulated by the Zachmann framework;
- the many forms in which descriptions of the main elements of the structure and functionality of a manufacturing enterprise can be viewed, as provided by early reference architectures;
- the structural description of IT services and functions provided by the OPENframework technical architecture;
- the methodologies for CIM introduction which seek to cover technical and managerial aspects of such a process, as expressed in: PERA; the SME wheel as extended by Thacker; the DISC framework of frameworks; and the OPENframework methods;
- the business strategy viewpoint associated with the enterprise engineering process, provided by a series of methods and tools which, if nothing else, serve as catalysers for the development of a strategic mind-set in the business modelling activity;
- the orientation towards enabling model integration embodied in the various federated architectures so far conceived, based on the assumption that a large diversity of legacy models and modelling languages will have to co-exist in an enterprise. This feature sets the stage for a scenario in which the complexity of enterprise integration is tackled by combining a variety of modelling tools and languages;
- the pioneering effort to provide a vendor-independent integrating infrastructure which can support flexible integration in manufacturing, as represented by CIM-BIOSYS (Here, a key aspect is that the use of CIM-BIOSYS at the MSI Research Institute has enabled its researchers to understand the key aspects and problems involved in building an IMS.);

- the possibilities that the application of the object-oriented paradigm offers to achieve inter-operation of multi-vendor software components marketed as software objects, opened up by initiatives such as: OMG-CORBA and ODP.

Hence, an issue of major importance is that of aggregating the features of such architectures into a coherently organised whole, so that their formalism can actually be used in (re-)engineering manufacturing enterprises. On-going efforts to compare, contrast, amalgamate and harmonise these architectures reflect this need to combine and use the most desirable features of outstanding architectures.

Chapter 3 - Research Objectives and Plan

The discussion presented in Chapter 2, has shown that a considerable body of work on formalism already exists, stemming from a variety of architectures, although most of them have not been tested or delivered in implemented solutions. Among these architectures, CIM-OSA stands out as the most comprehensive in terms of addressing the several perspectives from which architectures have been proposed. Furthermore, a desirable level of synergy can be obtained from the complementary perspectives of architectures, hence overcoming some of their individual limitations by combining them with one another in such a way that their best features can be used.

The complexity involved in realising an ideal combination of architectures is a challenge that is far beyond the scope of this research. Nonetheless, this research aims to make an incremental step towards such an objective and thereby in part meeting the requirements identified in Chapter 1, by:

- structuring the framework represented in Figure 5 in order to populate the IMS life cycle with formalisms of architectures. Such a structure is also meant to serve as a basis for delineating the author's long term research aim in this area[1];
- beginning to populate such a framework with an initial combination of formalisms of architectures (and added knowledge of 'problems' and 'solutions' as illustrated in Figure 5) which can be encapsulated into a CASSE environment capable of providing support for modelling, analysis, simulation, rapid-prototyping, configuration and operation of an IMS, in an integrated manner. Implementation of this environment will provide a basis for proving the concepts embodied in this research.
- evaluating the environment by applying it to (re-)engineer a manufacturing domain.

This chapter focuses on describing the strategy and the methodology adopted in order to achieve the above aims, by:

- introducing a preliminary framework for the IMS life cycle;
- detailing the methodology adopted to conduct this research;
- describing the main elements of the computer-aided systems engineering environment proposed for implementation;
- outlining the thread of decisions which have been made as part of the proposed plan

---

1. Investigating the formalism associated with architectures to support the complete IMS life cycle is essentially the author's broad research objective.

of research, as well as highlighting alternative courses of action which could have been taken.

3.1. A Preliminary Framework for the IMS Life-Cycle

Based on a simplified model of life cycle described by Figure 5, the life cycle model shown in Figure 7 was proposed as a means of structuring major elements of formalism with a view to achieving IT-based system integration. In each life cycle phase, certain design considerations will be dealt with, where deliverables and parameters of phases are inter-related. Figure 7 represents a decomposition of the four meta-phases[1], namely: strategy definition, conceptual analysis, design and implementation, and operation and maintenance. Each meta-phase and the constructs that are manipulated within it are defined below (the definition of each construct is presented in Appendix 1):

a. **Strategy definition.** This embraces the activities of: developing corporate, manufacturing and IT strategies, assessing the overall performance of an enterprise, defining goals for improvement, and identifying and prioritising enterprise domains to be tackled in order to achieve the required performance improvements.

b. **Conceptual analysis.** In this meta-phase, a formal system analysis is carried out with regard to one or more enterprise domains. Here, the focus is on analysing both present (i.e. 'as-is') and potential (i.e. 'should-be') scenarios in order to identify means of achieving the improvement goals defined for the domain(s) under consideration during the strategic definition stage. This process should lead to the selection of a 'to-be' scenario which provides means of achieving those goals.

c. **Design and implementation.** This meta-phase is concerned with realising a 'to-be' scenario. Hence, functional components (e.g. representations of people, machines, application programs, databases and infrastructural resources) modelled in the conceptual analysis meta-phase are implemented as physical entities that can be either purchased 'off-the-shelf', developed 'in-house' or fulfilled by the legacy components[2] of the enterprise.

---

1. In the model shown in Figure 7, the phases of "design" and "implementation" (defined by ISO TC 184/SC5 [ISO 1993] as shown in Figure 3) are combined into only one phase, namely: "design and implementation".
2. According to the ICEIMT [Petrie 1992a] legacy systems consist of any sub-systems or components which needs to be considered in an integration initiative, for it is already in place when the integration process is initiated. Once the integration initiative is completed it becomes the legacy for future integration initiatives and so forth.

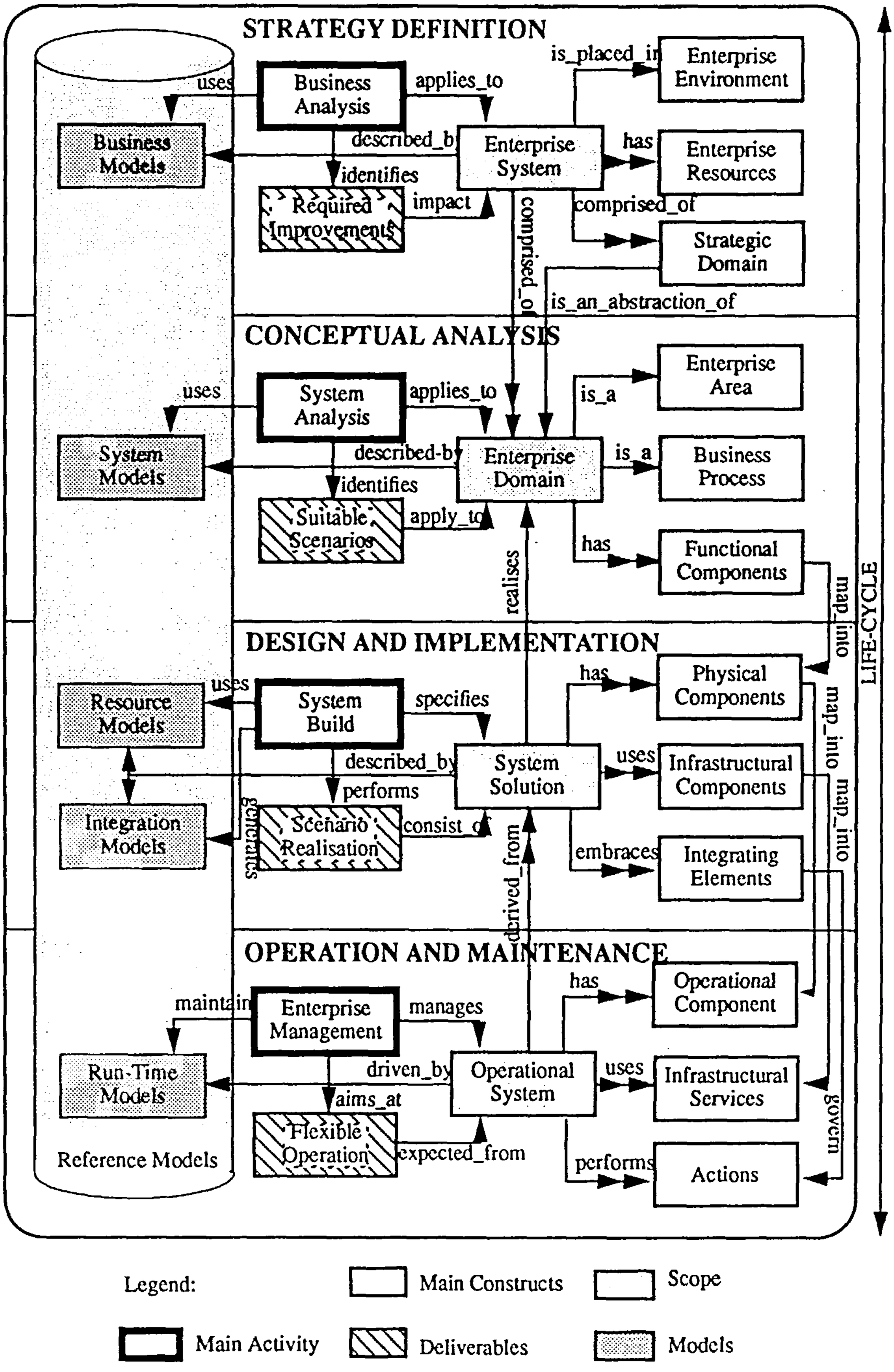


Figure 7 - Model of an integrated manufacturing system life cycle

d. **Operation and maintenance.** This meta-phase corresponds to the working life of the installed system, including stages during which evolutionary changes to system operation occur.

It is important to note that the model presented in Figure 7 seeks to identify key requirements, processes and outputs at each life cycle phase, and relationships among the constructs involved in each phase[1]. Therefore, neither flow nor definitive procedural relationships (between constructs within a meta-phase or constructs of different meta-phases) should be implied by Figure 7. The model does not intend to prescriptively structure the way in which system designers move on from one life cycle process to another (e.g. it should not be viewed as a 'water fall' model).

Additionally, the model is not meant to be complete so as to cover all issues of the IMS life cycle. It should rather be viewed as a vehicle for comprehending the key aspects of interest in this research. In this respect, the following features are worth noting.

a. Orientation of the model

The life cycle model is oriented towards addressing the mapping between enterprise models created at different levels of abstraction, leading to the construction and operation of a system whose components are integrated via an adequate information technology infrastructure (as illustrated in Figure 1). This embodies the need for mapping models which describe "how the system performs its task" with models that describe "how the system should be organised" so that it achieves its purpose through the inter-operation of its components. These tasks will typically involve the functionality content of each component (i.e. those that realise the component's own individual purpose) and tasks related to achieving a collective purpose through coordinating the execution of functionality of groups of components (i.e. business integration), thereby realising a required system behaviour[2].

b. Structure of the model

It should observed that the model structure has a similar pattern in each life cycle phase, i.e. this model has a fractal structure[3]. Such a structure reflects the essential elements that the author believes should be present at each phase, regardless of their

---

1. The notation used in this diagram is based on Backmann's entity-relationship model (from [Kay 1993]).
2. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], enterprise behaviour "is concerned with the flow of control or the way processes and activities are employed over time in reaction to events and according to the enterprise state".
3. A self-similar structure [Parunak 1985].

level of abstraction. Characteristic properties of the elements are described below (with examples being given for the strategy definition meta-phase of Figure 7):

- languages and frameworks required to support the activities carried out in the meta-phase (e.g. business analysis). This is the element to which the formalism proposed by contemporary architectures (such as those described in Chapter 2 can be more readily associated.
- a scope for the analysis (e.g. enterprise system);
- the principal classes of constructs which require manipulation at an appropriate level of abstraction, this reflecting the granularity of information processed at each meta-phase (e.g. enterprise environment, enterprise resources and strategic domains;
- the deliverable of each meta-phase (e.g. identification of required improvements);
- the essential knowledge required to carry out the activities at each meta-phase (i.e. reference models, e.g. business models). Although the term reference model is used here, knowledge in the form of formal models should not be viewed as mandatory[1]. Such knowledge could stem from the experience of designers or managers (i.e. knowledge of 'problems' and 'solutions' and how they inter-relate).
- an indication of the main types of agents involved during the execution of activities associated with each meta-phase (e.g. top-level managers for strategy definition).

The remainder of this chapter describes how parts of this IMS life-cycle model (related specifically to the three last phases, namely: conceptual analysis, design and implementation, operation and maintenance) can be populated via a combination of architectures.

3.2. Research Methodology

To accomplish the identified research objectives, the methodology shown in Figure 8 was proposed. This figure represents the major stages of the research and their associated deliverables. Indeed, this thesis is structured so that it reflects the form in which the stages depicted in its methodology have been executed.

3.2.1. Analysis of existing architectures

A starting point for this research was a thorough analysis of what reference

---

1. This type of formal knowledge can be obtained (for example) from some of the early reference models (see Section 2.2.1).

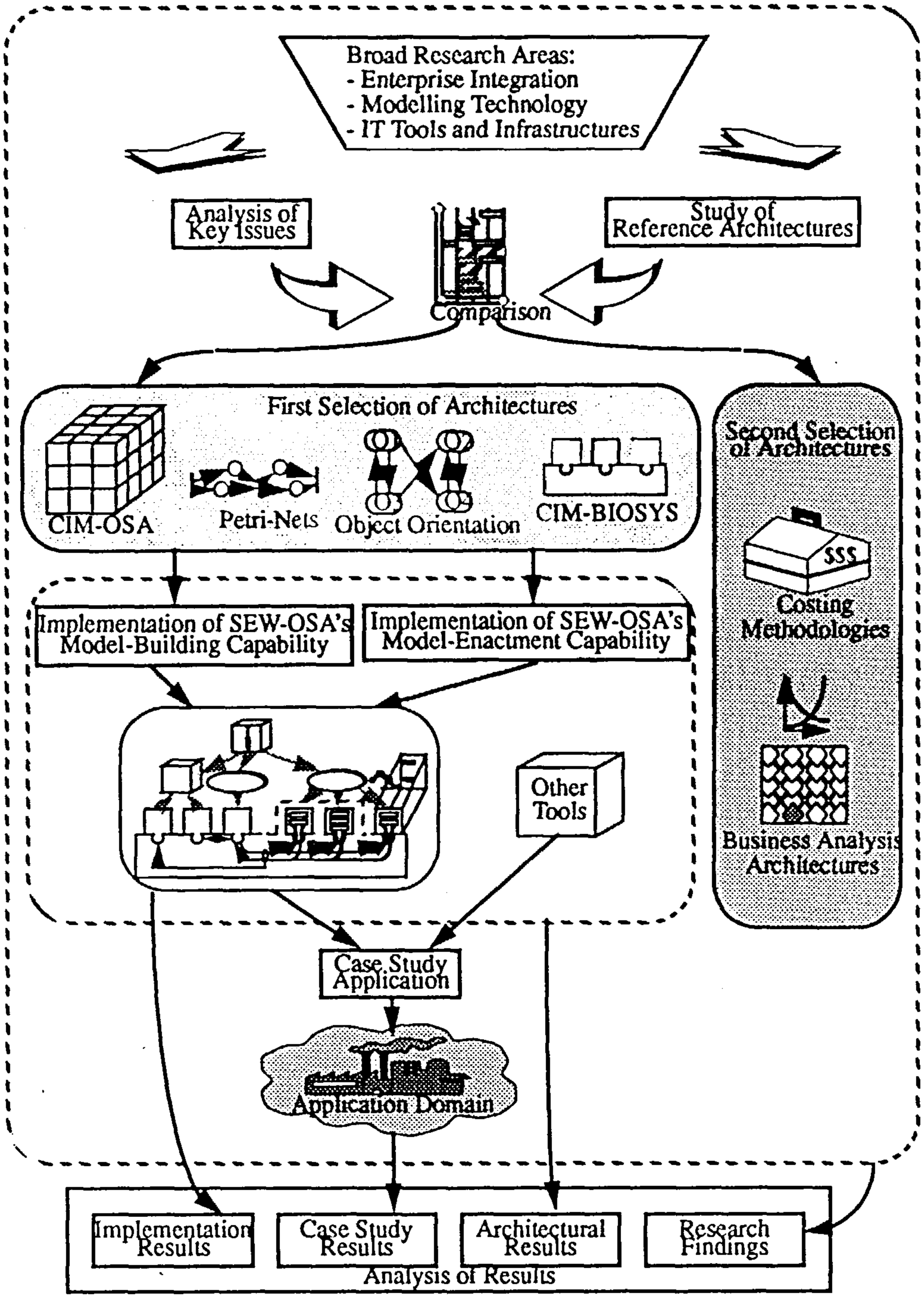


Figure 8 - Overall research methodology

architectures can offer to help cope with the complexity of enterprise integration. Two results emerged from such an analysis: an identification of some of the most relevant research projects on the definition of architectures for enterprise integration and a list of desirable features that an architecture should posses [Aguiar 1993e].

Subsequently, the results were used as a basis of an analysis which concluded (as discussed in Chapter 2) that:

a. there still exist considerable gaps in the formalisms of architectures, so that no single architecture can provide the required support for all the issues involved in the IMS life cycle, although CIM-OSA stands out as one of the most comprehensive;

b. an architecture resulting from a combination of the best features of a number of outstanding architectures is desirable.

Furthermore, as Chapter 2 has shown, a number of modelling tools, design tools and simulation tools (which originated from a variety of architectures) exist that can address isolated aspects of the IMS life cycle. However, in spite of the efforts of a number of research groups, no single initiative has provided comprehensive support to the IMS life cycle, nor embraced a methodology and its associated architectural concepts into a CASSE environment capable of supporting the modelling, analysis, simulation, rapid-prototyping, configuration and operation of an IMS in an integrated manner.

Although CIM-OSA provides a very comprehensive specification of a framework which can be used to realise key aspects of that purpose, it does not yet provide solutions for all of them. For example, current deficiencies are particularly evident for its function, resource and organisation views at the design specification and implementation description modelling levels [ESPRIT/AMICE 1993a][1], as discussed in detail later in this thesis.

3.2.2. Selection of architectures

Thence, two sets of architectures were selected for such a combination (as indicated in Figure 8). The first set comprises those architectures required to address the need for a CASSE environment to realise support of a wider scope of the life cycle than previously achieved; this by combining CIM-OSA [ESPRIT/AMICE 1993a], generalised stochastic time Petri-nets [Juanole 1989], predicate-action Petri-nets [Peterson 1981], object-oriented design methods [Booch 1991], and the services of the CIM-BIOSYS infrastructure [Weston 1993]. The second set consists of additional architectures candidates for integration that could be used to support other analysis and design considerations, such as architectures for business analysis. Consideration of these architectures is beyond the scope of this research.

The first selection of architectures provides means of overcoming primary limitations encountered in the CIM-OSA architecture, namely:

---

1. The modelling levels of CIM-OSA (i.e. requirements definition, design specification and implementation description) relate to two meta-phases of the life cycle, namely: "conceptual analysis" and "design and implementation".

- Petri-nets and object-oriented design were adopted to populate design specification and implementation description modelling levels of CIM-OSA, thus enabling complete support for model-building;
- Petri-nets were also adopted as a means of enabling analysis and simulation;
- CIM-BIOSYS was the only integrating infrastructure available (at the time that this research started) which could be enhanced in order to support model-enactment, thus enabling rapid-prototyping of an IMS.

3.2.3. Model-building and model-enactment capabilities

This first selection of architectures is envisaged to be encapsulated into a CASSE environment. Such an environment is necessary in order to enable the investigation of a plethora of integration 'problems' and possible 'solutions', by providing means of: (1) describing each problem and defining possible solutions through the application of a formal language; and (2) testing the solutions. These requirements lead, respectively, to the proposition of the two main capabilities for such an environment, namely: a **model-building capability** and a **model-enactment capability**[1] (as shown in Figure 8). Basically, each class of capability is required to support more than one modelling level of CIM-OSA [ESPRIT/AMICE 1993a], in a way which bridges gaps that exists within and between these modelling levels. Such capabilities are required to enable graceful migration from a modelling description of 'what' the system should do, to a description of 'how' the system should do it, by means of the actions executed by its components. Indeed, an important contribution envisaged for these two capabilities is that of defining a method for the organised application of the architectures, where their amalgamation would be under the framework defined by CIM-OSA.

Essentially, the method proposed would be based on the application of a model-building capability to generate models which formalise a possible relationship between problem and solution, which could then be made available to a model-enactment capability in order to test the relationship (see Figure 9). The integrated use of these two capabilities is the fundamental support provided by the CASSE environment that this research is aiming to construct for encapsulating the formalism of the first selection of architectures.

Figure 9 presents a proposed structure for the CASSE environment, which provides an indication of the major functional elements that it should embrace. This environment will be referred to in this thesis as SEW-OSA[2], a "system engineering

---

1. Model-enactment is viewed in this research as the ability to let a model evolve over time (i.e. the evolution of its dynamic behaviour).

workbench centred on CIM-OSA" due to its orientation, namely:

- it aims to address the **engineering** of an enterprise from a **systems perspective**;
- its underlying framework is structured chiefly on the formalism of **CIM-OSA**;
- it aims to provide the level of life-cycle support of a **workbench**. According to Brathwaite, a "workbench" consists of integrated tools for automating the entire design process, which encapsulates: the steps to be followed during the process (method); the constructs to be created and manipulated in order to formalise the design; the considerations, analyses and decisions to be made in each step; and associated documentation of design activities.

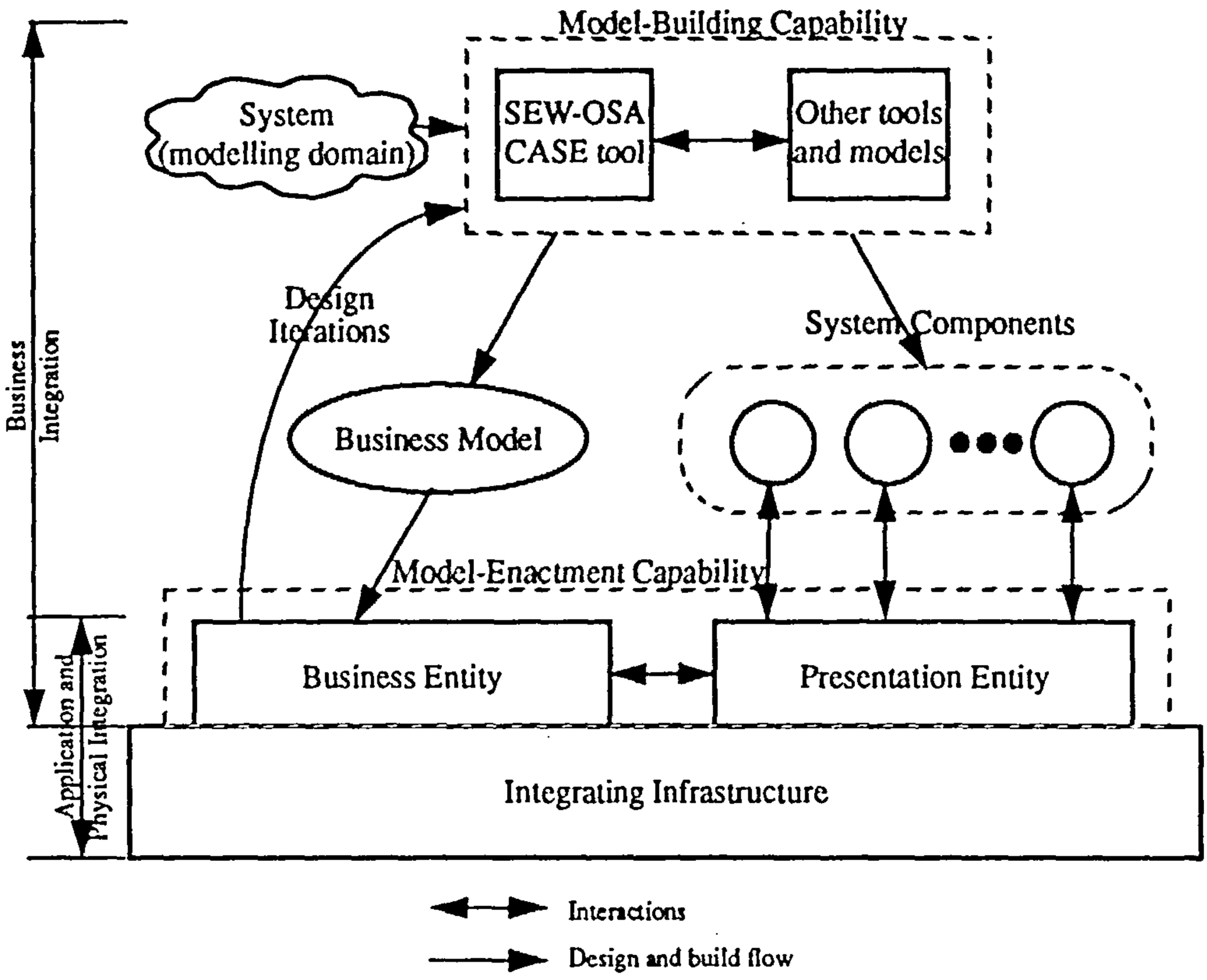


Figure 9 - SEW-OSA: system engineering workbench for CIM-OSA

3.2.4. SEW-OSA and the life-cycle support

By constructing, populating and applying SEW-OSA, support will be provided to enable the mappings required between phases of the IMS life cycle (see Figure 7):

---

2. The terms CASSE environment, workbench, environment, tool-set and SEW-OSA are used interchangeably, henceforth in this thesis.

a. Mapping between "design and implementation" and "operation and maintenance":

This mapping includes enabling system models to be enacted in such a way that they structure and control the operation of a real (or physical) system. Here, mapping is required between abstract models manipulated by the upstream activities of the life cycle and the integration process and physical entities employed in the real system

The realisation and application of the model-enactment capability is viewed by the author as the way of reflecting the CIM-OSA formalism within the CIM-BIOSYS integrating infrastructure. This was conceived as a means of using models to structure and enable rapid-prototyping, configuration and operation of an IMS. However, this hypothesis needs to fleshed out and tested and the extent to which it could be useful also needs to be established.

b. Mapping between "conceptual analysis" and "design and implementation":
This embraces further mappings between life-cycle processes (i.e. life cycle models and activities) to enable the specification of a physical solution based on an identified set of requirements.

This mapping includes enabling a "test" of the properties and the quality of models which describe a relationship between a "problem" and a "solution" (i.e. the relationship between system requirements and a possible system configuration). It is envisaged that such models should be "produced" by a model-building capability and "tested" with the aid of a model-enactment capability, this by means of an integrated support for modelling, analysis, simulation and rapid-prototyping. It is also envisaged that use should be made of a library of reference models (i.e. system models and resource models - see Figure 7) that could encapsulate knowledge of: (a) user requirements in the form of CIM-OSA-based models and (b) alternative system components modelled in an object-oriented language, thereby integrating two complementary descriptions of the system: a business process view and a object-oriented view.

c. Mapping between "strategy definition" and "conceptual analysis"[1]:
As discussed in Chapter 2, very few architectures have considered mapping issues

---

1. An illustration of this mapping can be found in the "Hamburger Bun" model proposed by Schofield and Bishop [Bishop 1989]. This model presents the business strategy issues as the top half of the "bun"; the integration and systems issues (related to the remaining life cycle phases) as the bottom half of the "bun"; and the meat of the "hamburger" as the chain of activities which add value to the product.

between strategy definition processes and the remaining phases of the life cycle, in such a way that business issues can be formally reflected within system design models.

The author's conception of the CASSE environment is that it should enable the incorporation of future architectures that address such issues which might itself include the second selection of architectures (see Figure 8).

Therefore, **realising, applying and evaluating the environment shown in Figure 9 constitutes the main focus for this research**. Once completed, this environment should support part the activities of the IMS life cycle from the conceptual analysis phase onwards, by:

- providing a systems engineering workbench to support the design activities encapsulated in the functions termed as 'system analysis' and 'system build' in Figure 7;
- providing a structure for populating the workbench with reference models (e.g. 'system models', 'resource models', 'integration models' and 'run-time models' - see Figure 7);
- providing the means of executing 'run-time models', thus enabling 'flexible operation' (see Figure 7);
- making available instances of models which are generated whilst applying of the workbench in manufacturing case studies;
- providing a wide scope workbench upon which additional elements of formalism could be incorporated into the life cycle in future research initiatives.

It should be emphasised that this research approach does not seek to produce fully compliant CIM-OSA solutions, nor does it seek to definitively appraise existing architectures. Rather **the aim is to take a "thin" slice through the IMS life cycle by building on a selection of formalisms which exists as a part of contemporary architectures.**

3.2.5. Realisation of SEW-OSA

It is envisaged that realising SEW-OSA by implementing, applying and evaluating both the model-building capability and the model-enactment capability as depicted in Figure 9 would involve the following activities (discussed in forthcoming chapters), namely:

a. Realisation of the model-building capability, by:

- realising a CASE[1] tool that encapsulates the formalisms of architectures into a method which enables its application to IMS modelling; and
- establishing links to other tools and models which have been created within the "Model-Driven CIM" research programme which address design aspects complementary to those addressed by SEW-OSA, namely: information and resource modelling (see Appendix 2 for an overview of "Model-Driven CIM").

b. Realisation of the model-enactment capability, by:

- establishing means of linking the SEW-OSA CASE tool to tools that enable analysis and simulation of the behavioural aspects of the system based on the use of a business model generated by the CASE tool;
- realising a business entity which functions as an engine that transforms the formalism encapsulated in the business model into interactions among system components; and
- specifying a presentation entity for CIM-BIOSYS which can serve as a means of relating the interactions generated by the business entity to the functionality provided by the physical components of the system.

c. Application and evaluation of the model-building and model-enactment capabilities, by:

- developing a case study application of SEW-OSA in an industrial site; and
- evaluating the results obtained.

These activities constitute the principal elements of work proposed to be performed as part of this research plan.

3.3. Structure for the Research Decisions Made

An initial challenge faced by this research was to delineate a focus for the research work as the scope depicted in Figure 10 is very wide indeed (see first stage in the methodology depicted in Figure 8), particular in the context of a Ph.D.

In achieving that delineation, a series of decisions had to be made to constrain the research without invalidating its conclusions. Figure 11 reflects the way in which

---

1. In this thesis the term "CASE (Computer Aided Software Engineering) tool" is used to designate a tool which supports model-building and code-generation, as opposed to "CASSE environment" which designates a set of tools which provide complete support for the IMS life cycle activities of interest.

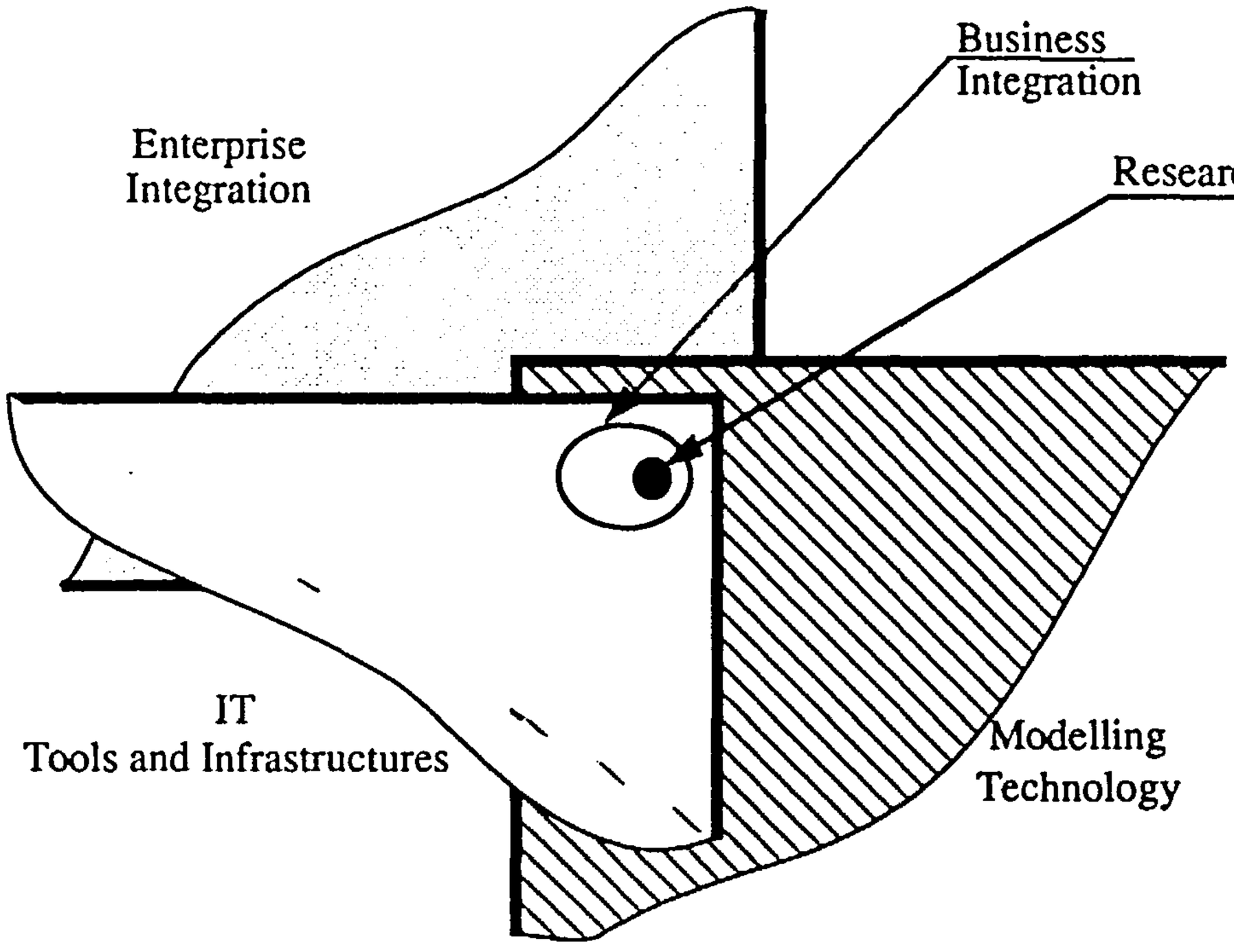


Figure 10 - Research focus

the scope of this research[1] was narrowed down from the broad areas represented in Figure 10. Each node in this structure embodies a decision based upon an assumed rationale. Corresponding to each course of action taken, a number of alternative approaches were identified (as discussed below). By so doing, it is fully recognised that **this research plan represents but one approach to establishing new methods and knowledge in the research areas involved.**

The focus on manufacturing enterprises also reflects the scope of contemporary reference architectures. Although other types of enterprises were not directly addressed in this research, it is a thrust of most architectures that their concepts can be extended to various types of enterprises.

The focus on IT stems from the orientation of the research within the MSI Research Institute during the period of this thesis, with particular emphasis to the "Model-Driven CIM" project. Alternative approaches could have focused on organisational and human-related aspects of the enterprise engineering process.

Following from the IT focus comes an emphasis on applying a 'hard-systems'

---

1. The "Workbench Development" box is represented in "bold" in this figure as it is considered in more detail later in the thesis.

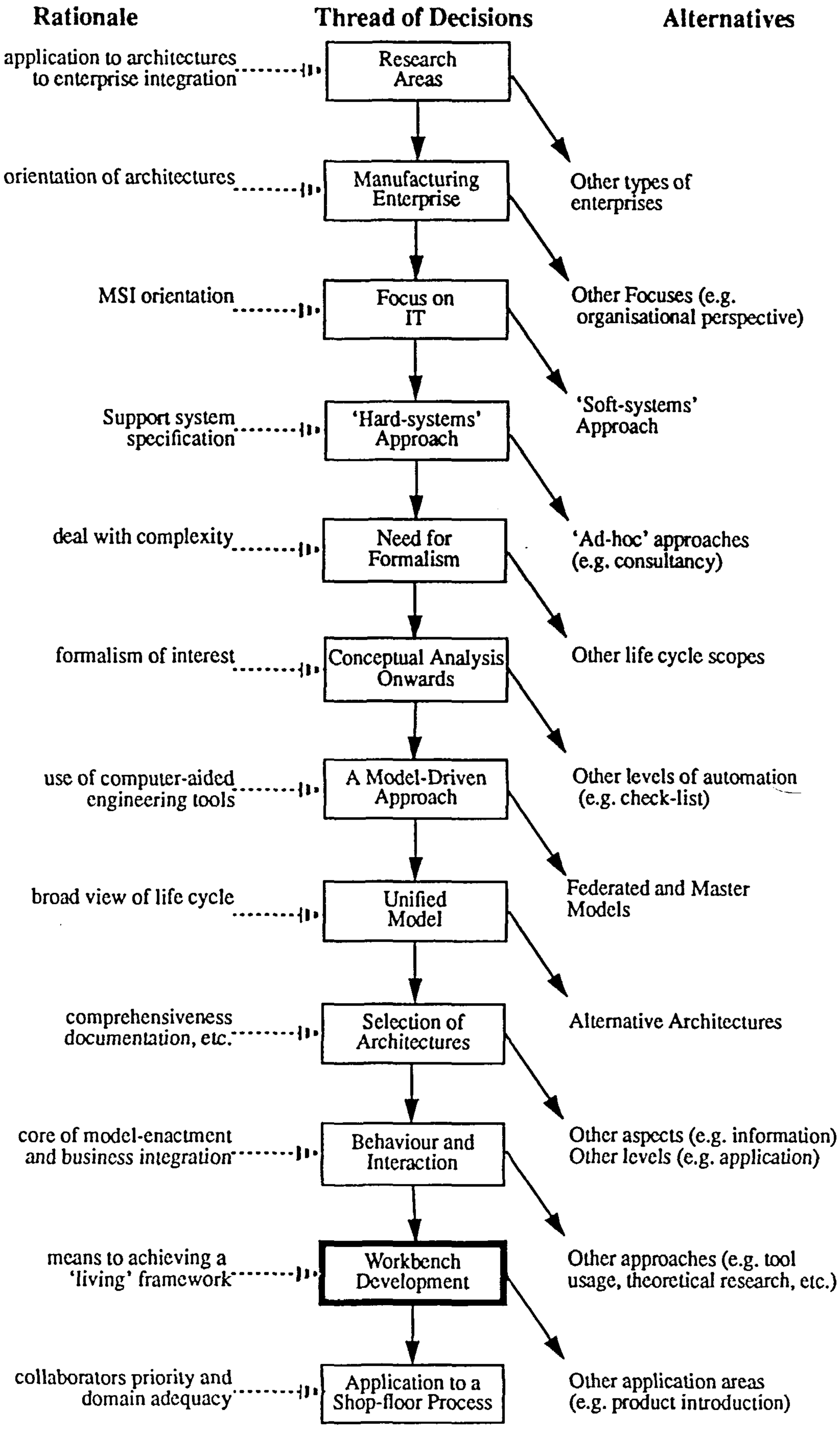


Figure 11 - Structure for the research decisions made

approach[1] as supposed to a 'soft-systems'[2] one. As this research aims to cover much of the IMS life cycle (leading to the definition of a configuration for the IMS), a reductionist view of the problem (inherent in the 'hard-systems' approach) is crucial. However, this does not imply that a 'soft-systems' approach cannot be used in combination with the 'hard-systems' one, in order to support the complete IMS life cycle. In fact, Weaver, [Weaver 1994] advocates that

> "the output from [a] soft-systems phase applied to the higher levels of a process [can be (i.e. Figure 1.c)] used as the input to the hard-systems phase at lower levels of detail [i.e. Figure 1.b]. The two approaches can therefore be regard as 'complementary' and form part of a single business process re-engineering methodology".

The need for formalism to structure and support the IMS life-cycle process is a prime driving factor for employing modelling technology as a means of coping with the inherent complexity of enterprise integration. Alternative approaches can be encountered in 'ad-hoc' procedures traditionally employed by systems integrators and consulting businesses, with occasional use of modelling methods (e.g. CIM-Plan [Hales 1989] [Hales 1990]).

Focusing the study on the 'conceptual analysis' processes onwards (rather than attempting to cover the complete life cycle) was a necessary measure to reduce complexity, whilst addressing key aspects of the enterprise engineering process. However, other aspects of formalism could have been addressed.

Formalism in the form of a life-cycle-wide model-driven approach would establish a facility for enterprise engineering, which could be encapsulated into a CASSE environment. However, varying degrees of formalism can be adopted and introduced in a computer-aided manner. For example, formalism could be introduced as a checklist of issues to be considered during engineering processes, such as that used

---

1. According to Trought (from [Weaver 1994]), the 'hard systems' approach "... assumes that within the system there is access to all the relevant data, [that data] is also accurate, and that all alternatives [...] choice and consequences are unambiguous".
2. According to Weaver [Weaver 1994], a 'soft systems' approach "starts from the perspective that the problem situation or process is ill-defined and the process is stochastic in nature. It would then attempt to create [a] "rich" picture of the problem situation and the systems involved [... which] would then be compared with reality. Feasible and democratic changes would be agreed upon through a debate and the changes implemented." In such a process, [...] properties that are only a result of interactions between all elements in a process (emergent properties) can be considered more easily".

by OPENframework [ICL/OFD 1993].

Within the context of a model-driven approach, a unified model was selected (see recommendations of the Enterprise Integration Program mentioned in Chapter 2 [Petrie 1992a]). This consists of adopting a homogeneous set of languages to support modelling activities along the complete IMS life cycle. Such an approach presents the advantage of enabling the description of a broad view of the life cycle, and the disadvantage of constraining modelling to the application of only a set of languages. Alternative models are [Petrie 1992a]: the master model and the federated model.

A selection of architectures was necessary to establish a basis of formalism upon which the life cycle activities could be fleshed out. The rationale for the definition of such a selection is based on the survey of existing architectures presented in Chapter 2, whereby issues of comprehensiveness and documentation among others played a major role (see early studies by the author [Aguiar 1992d] [Aguiar 1993e] [Aguiar 1993c] [Aguiar 1994m]). However, an alternative selection of architectures could have been adopted as a starting point.

From the set of modelling views[1] provided by the selected architectures (which can be related to function, information, structure, organisation and behaviour), functional, resource and behavioural aspects were selected as views of particular interest. This is chiefly due to the fact that these views play a central role in achieving model-enactment, as well as in driving the dynamic interactions among system components (i.e. "engineering" the relationships in Figure 1). This selection is also associated with the level of integration that this research is interested in addressing, namely: business integration, as supposed to application integration or physical integration (see Figure 2).

The development of SEW-OSA as a CASSE environment for bringing together the formalisms of architectures into a usable piece of software is viewed as being fundamental in constructing a 'living' framework (i.e. a structure to which functionality can be fleshed out as the formalism evolves or grows). Alternative approaches could be to use available tools regardless of their limitations and fill the gaps with 'patchy' software fragments and paper-based models.

As the SEW-OSA workbench was conceived and advanced, its use was appraised by conducting studies concerned with re-engineering issues connected with shop-floor process. This application area was selected by considering mainly: the interest of the collaborating company which provided the case-study information, as well as the fact that it provided application scenarios which could utilise and test the

---

1. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], a modelling view is "an abstraction viewpoint of one total view, which emphasises some particular aspects of the model and disregards others for ease of analysis".

functionality of the workbench (i.e. its model-building and model-enactment capabilities).

3.4. Concluding Remarks

Even after being limited by the constraints delineated in the previous section, the research objectives proposed in this chapter are still fairly ambitious. Thence, it is important to emphasise that the research activities proposed to be performed are founded upon two complementary foundations: (1) a thorough study of the formalism associated with the three latter phases of the life cycle and (2) a significant involvement in systems integration threads by the author[1].

These two factors combined (i.e. experience and formalism) can enable the development of the research program outlined in this chapter within its limited time-span (i.e. three years).

---

1. Such an experience embraces activities, such as: analysis of requirements of manufacturing systems, specification of solutions in terms of integrated computer systems to address those requirements and the development, implementation and integration of hardware and software components in those systems for a number of manufacturing industries in Brazil.

Chapter 4 - First Selection of Architectures and Associated Tools and Services

This chapter introduces the first selection of architectures and presents a description of their associated tools and services used to realise the research plan proposed in Chapter 3. It also describes the strategy adopted for the realisation of SEW-OSA by combining these tools and services under the umbrella of the CIM-OSA architecture.

4.1. CIM-OSA

Figure 12 depicts the two main deliverables of the CIM-OSA/AMICE consortium, namely: a **modelling framework** and an **integrating infrastructure.** This section briefly describes CIM-OSA and the state-of-the-art in regard to each of its two main deliverables, based on the information supplied in its formal reference base (FRB)[1] [ESPRIT/AMICE 1993a].

4.1.1. The CIM-OSA modelling framework

The CIM-OSA modelling framework embraces definitions of the complete cube of Figure 12. It provides a framework for the definition of a CIM-OSA compliant architecture for a particular enterprise, based on instantiations of the CIM-OSA reference architecture (i.e. generic and partial genericity levels of the cube in Figure 12). The three dimensions of the cube describe the modelling process as envisaged in CIM-OSA. Progression along these three dimensions is achieved in a manner which is referred to as: stepwise generation, stepwise derivation and stepwise instantiation.

a. Stepwise generation:

Stepwise generation defines the modelling aspects (or views). It embraces a description of function, information, resource and organisation. **Function view** provides a hierarchically structured description of the functions of an enterprise, their behaviour (dynamic aspects) and their functional content (static aspects), based on the objectives of the enterprise and reflecting external constraints imposed upon it (in the form of relevant inputs and outputs). **Information view** encompasses the description of all pieces of data and knowledge identified from the needs of enterprise users and applications. **Resource view** contains all relevant information on enterprise resources

---

1. An FRB is the most detailed documentation available about CIM-OSA. Its circulation was initially limited to the members of the ESPRIT/AMICE consortium.

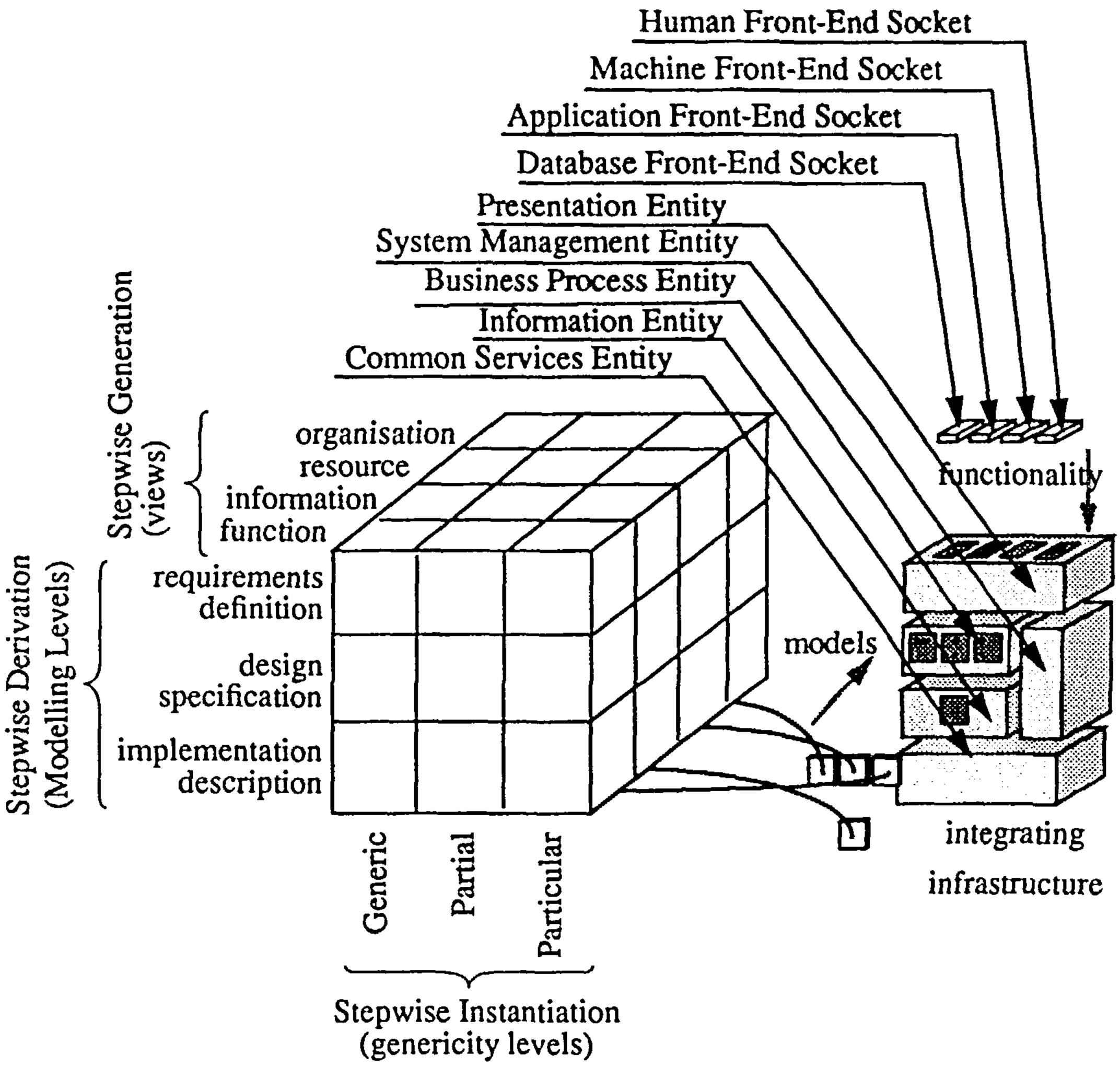


Figure 12 - CIM-OSA reference architecture

required to support the execution of enterprise functions. **Organisation view** holds the definition and identification of responsibilities and authority over functions and resources within an enterprise, for the sake of human intervention and decision making in situations where changes in requirements or exception handling are required.

b. Stepwise derivation:

Stepwise derivation is a life cycle view of the modelling process, structured within various modelling levels. Derivation embraces the definition of three modelling levels, namely: requirements definition, design specification and implementation description, which imply a translation from a business description language into a system description one.

At the **requirements definition modelling level** the business requirements of an enterprise are identified. This modelling level describes the enterprise from a user's point of view.

At the **design specification modelling level** the user requirements are restructured, detailed and optimised based on a consistent model which simultaneously takes into account business and technical constraints, in order to specify solutions for those requirements.

At the **implementation description modelling level** means of executing the enterprise model are defined by selecting information technology and manufacturing technology components such as human resources, machines and application programs[1] required to support enterprise operation.

c. Stepwise instantiation:

Stepwise instantiation is related to the genericity levels, namely: generic, partial and particular. The first two levels (i.e. generic and partial levels) constitute the CIM-OSA reference architecture.

At the **generic level,** the CIM-OSA modelling constructs[2] presented in Table 1[3] comprise its modelling language. It should be noted with respect to this table that controversy still exists within the AMICE consortium as to whether the same constructs manipulated in distinct modelling levels (sometimes in different ways) should be named differently. In this table, the renaming process was adopted as a means of listing the main constructs of interest at each modelling level (e.g. event, specified event and implemented event). However, in this thesis, no distinction is made between these constructs across modelling levels.

At the **partial level,** partial instantiations of these generic constructs and/or combinations of constructs are used to describe modelling scenarios common to certain types of enterprises (e.g. partial models of shop-floor in the aerospace industry).

However, it is at the **particular level** that an architecture is defined which is suitable for a particular enterprise and is instantiated from the reference architecture. Relationships among these levels are illustrated by Figure 13. In this figure, three levels of genericity are associated with an enterprise activity and its related inputs and outputs.

4.1.2. The CIM-OSA modelling methodology

The following provides a description of the main elements of the modelling methodology associated with the definition of a particular architecture (see Figure 12).

---

1. The terms *application programs*, *software applications*, *application functional entities* and *active resource components representing software applications* are used interchangeably through this thesis.
2. A construct is a generic building block which characterises an element of formalism for a modelling method or language.
3. Definition of some of the terms listed in this table are presented later in this thesis, being a detailed discussion of their meaning presented in [ESPRIT/AMICE 1993a].

Table 1 - CIM-OSA constructs

views	requirements definition	design specification	implementation Description
function	domain domain Relationship event domain process business process enterprise activity objective/constraint declarative Rule procedural Rule capability	functional operation specified event specified domain specified domain process specified business process specified enterprise activity specified capability	implem.functional operation implemented event implemented domain implemented domain process implem. business process implem. enterprise activity implemented capability
information	object class enterprise object object relationship object abstract mechanisms information element integrity rule object view configuration configuration element	external schema conceptual schema entity relationship attribute integrity constraints database transactions enterprise object	implem. external schema internal schema logical data model physical data model enterprise object
resource	NULL	resource (cell and set) resource unit location	implemented resource implemented resource unit implemented location
organisation	organisational guidelines	authority organisational unit organisational cell organisational element organisational relationship responsibility	authority profile implem. organisational unit implem.organisational cell Impl. organisational element Impl. organis. relationship responsibility profile

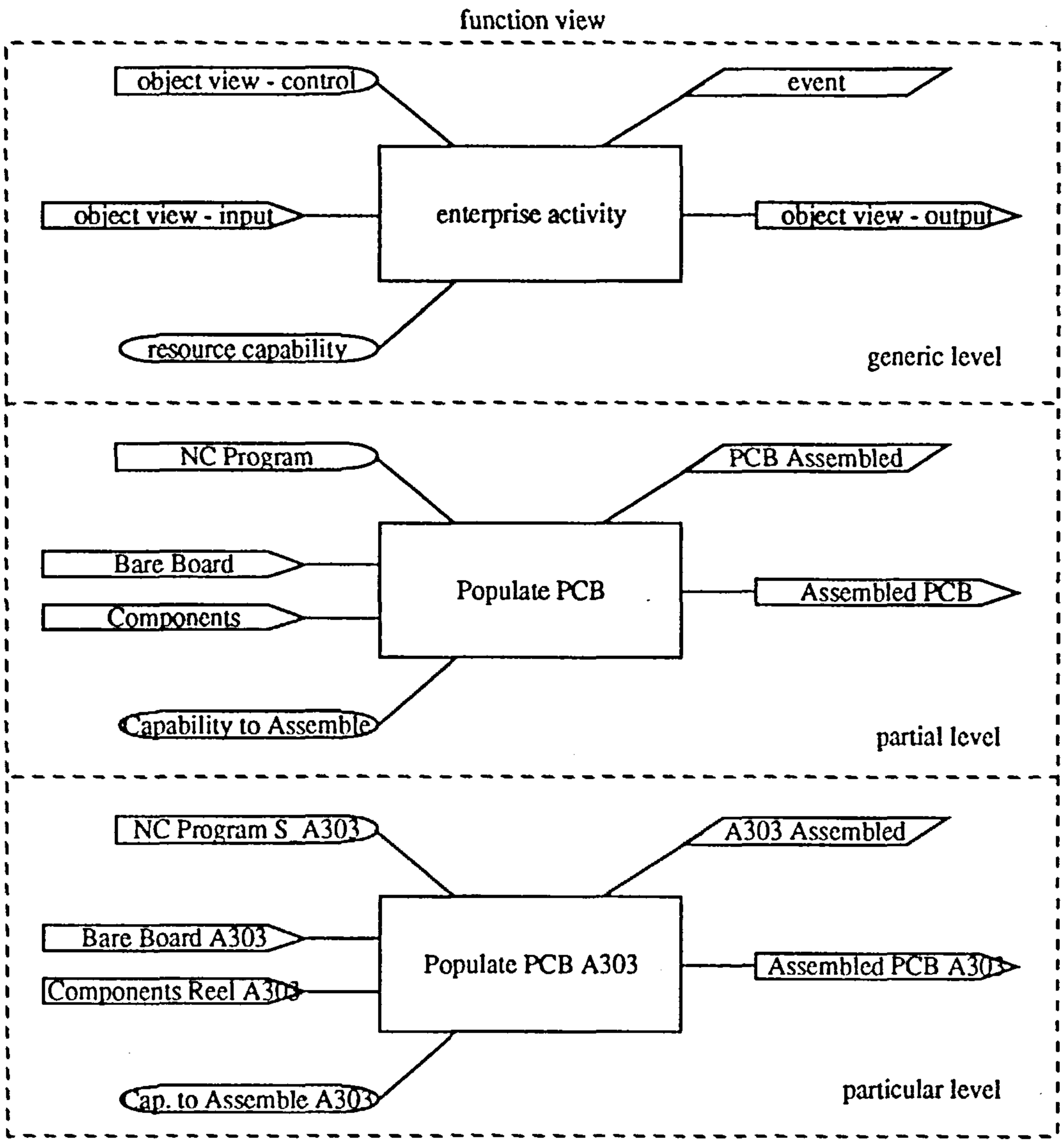


Figure 13 - Relationships amongst genericity levels - enterprise activity example

a. Function view

The **requirements definition modelling level** in the function view is the most complete and detailed in CIM-OSA. It describes the functional, behavioural and structural aspects of an enterprise, by means of two levels of description [Kosanke 1992], namely: enterprise level and process level.

At the **enterprise level**, an enterprise is modelled as a set of loosely coupled processes sharing common enterprise objects. Objects communicate with one another by exchanging object views and events, as illustrated in Figure 14 [Kosanke 1992]. A domain represents an area of the enterprise which can either be CIM-OSA-compliant or non-compliant. A CIM-OSA-compliant domain possesses an internal structure

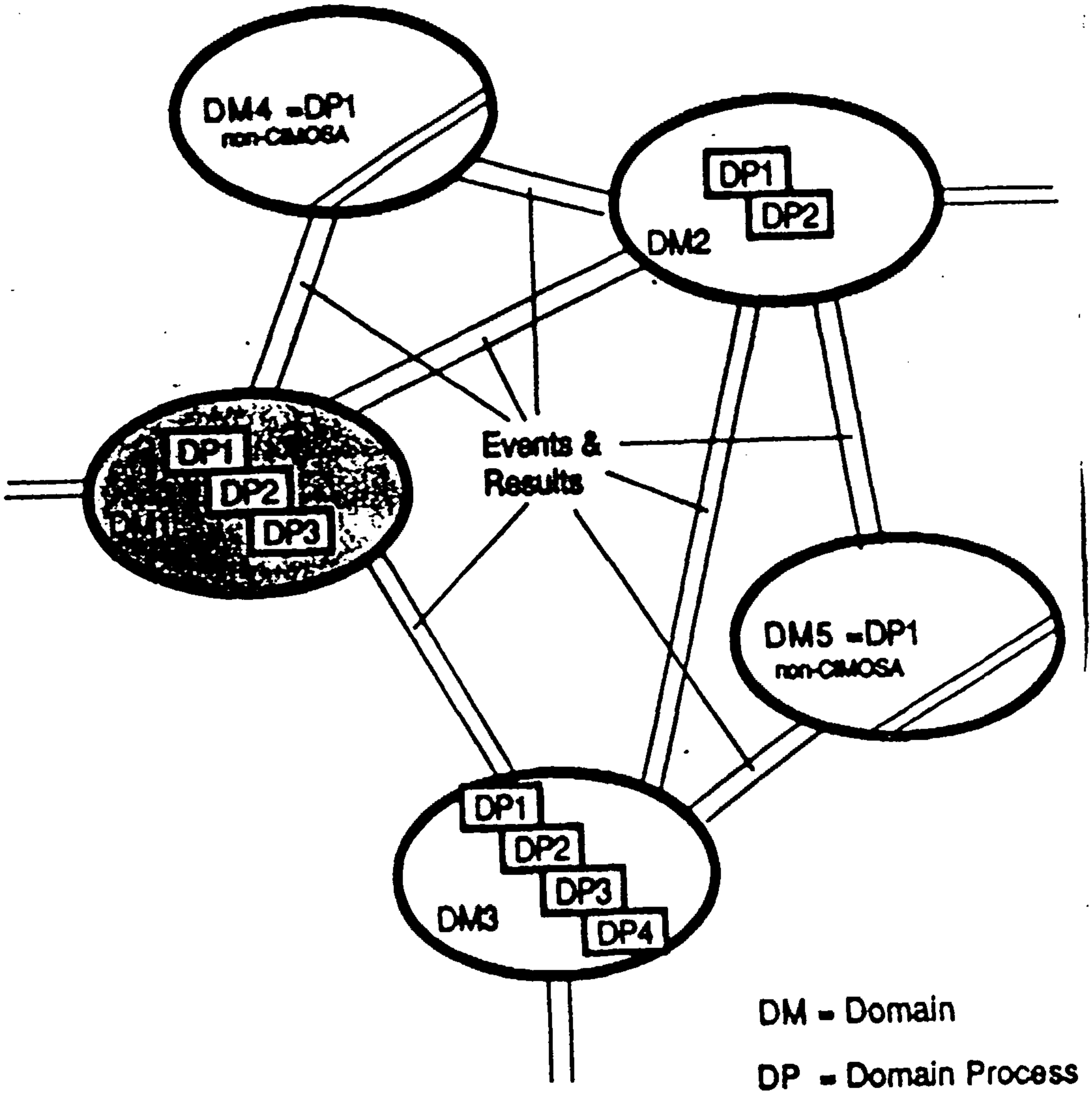


Figure 14 - Enterprise level

modelled by means of domain processes. A non-CIM-OSA-compliant domain can represent sub-systems of the enterprise which are not going to be modelled by CIM-OSA, although they may have to interact with CIM-OSA-compliant domains. Typical non-CIM-OSA-compliant domains are legacy systems or ancient applications requiring interaction with other domains, but whose internal organisation is not germane to the integration process.

At the **process level**, the structure, behaviour and functionality of domain processes belonging to CIM-OSA-compliant domains are completely defined. As illustrated in Figure 15 [Kosanke 1992], the structure of a domain process is defined in terms of its decomposition into business processes and enterprise activities[1]. Enterprise

---

1. In this thesis, the term process (when referring to a modelling construct) is used as a generalisation of business process and domain process. The term function or enterprise function (when referring to a modelling construct) is used as a generalisation of business process and enterprise activity.

activities represent pure functionality (i.e. atomic functions capable of performing transformation), namely: acting upon its inputs in order to produce desired outputs. Business processes (and domain processes) capture behaviour; that is, the manner by which the processes use their component processes and activities, in order to describe their functional content by means of procedural rules (i.e. relationships of the type "precedes").

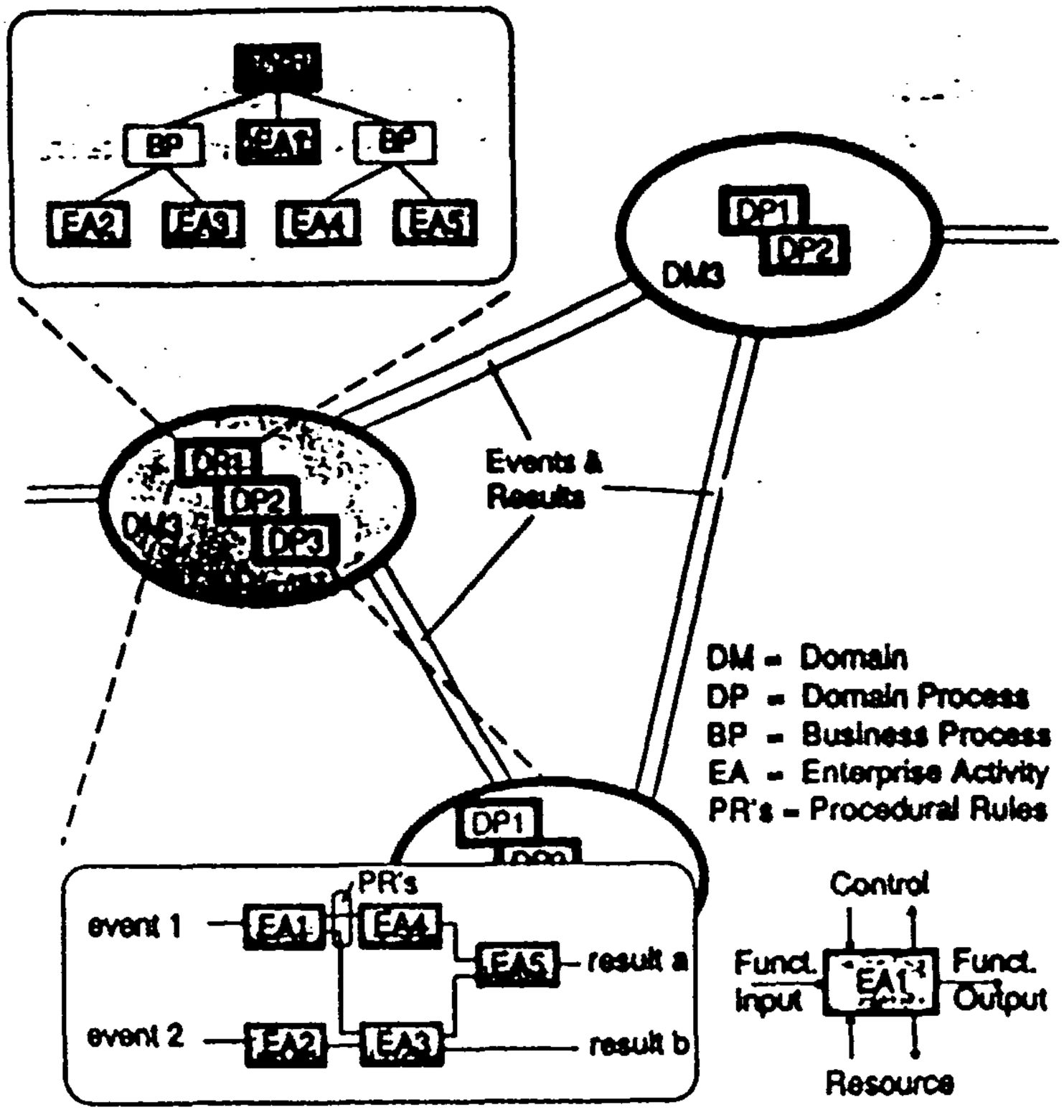


Figure 15 - Process level at the requirements definition modelling level

Basically, domain processes are high-level processes of an enterprise which cluster lower level processes and functions. They provide the most abstract representation of a domain (i.e. they are at the top of its functional decomposition). Domain processes are triggered by events and they are not recursive. Business processes are used by domain processes which, in turn could use other business processes or enterprise activities. Business processes and enterprise activities are of a recursive nature. That is, they are considered to form a pool of functions which are typically reused by different domain processes and business processes within and without the same model.

The result of such a recursive decomposition is a network of enterprise activities (i.e. transformation functions) connected by procedural rules (see Figure 15). Using

CIM-OSA terminology, this network is referred to as the **business model.**

At the **design specification modelling level,** the function view checks the requirements definition model for consistency and redundancy, leading to the definition of the 'specified' constructs shown in Table 1). This view also defines functional operations which are atomic units of work performed by the resources that the enterprise activities utilise (in order to perform their functional transformations). These functional transformations are described in terms of the internal behaviour of enterprise activities when triggering their functional operations (see Figure 16).

At this level, a number of solutions are examined in terms of alternative resources which can provide the functionality required by the enterprise activities, by taking into account time and synchronisation issues, resource requirements and overall system performance considerations. Although, CIM-OSA specifies the content of

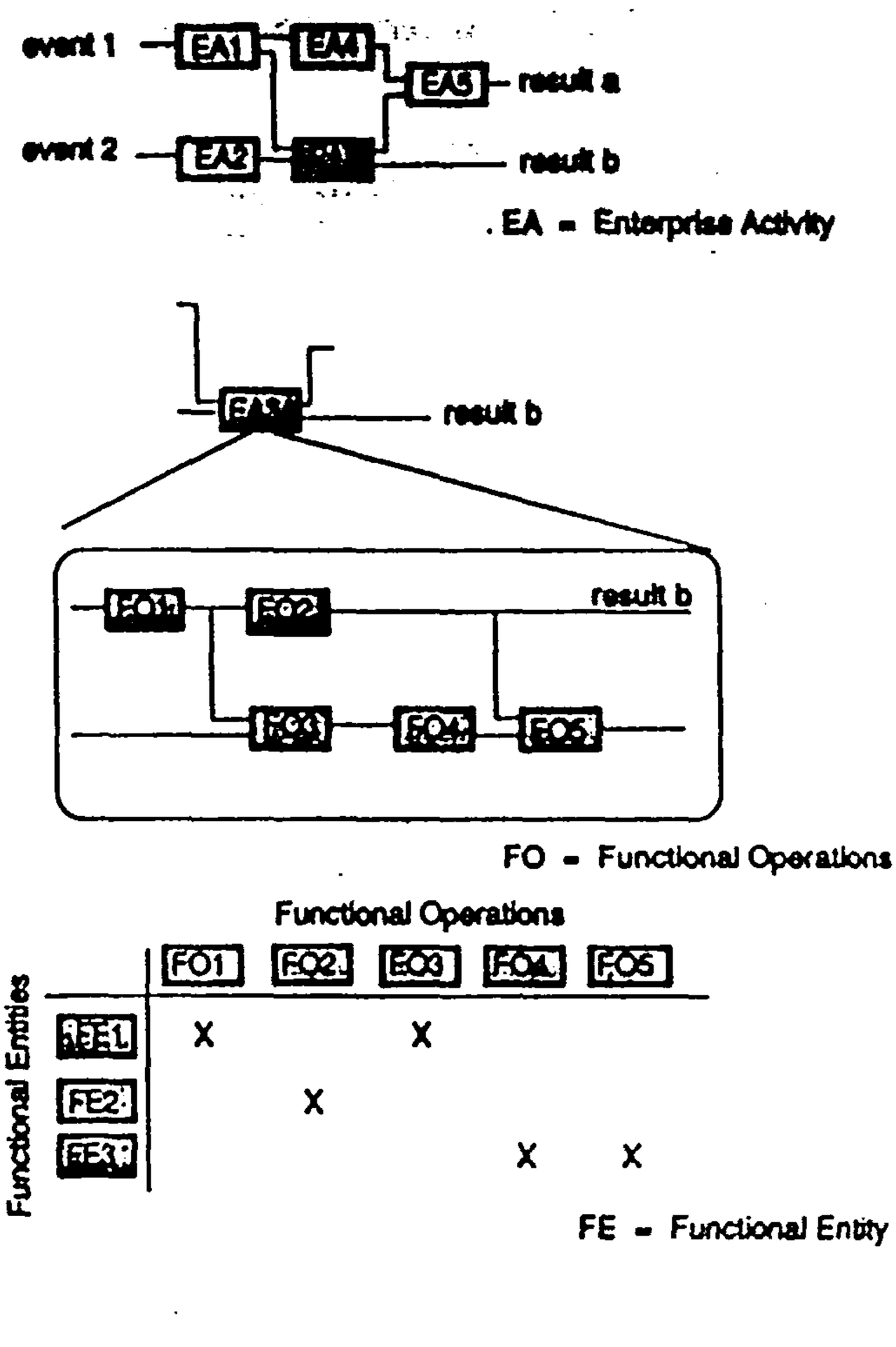


Figure 16 - Process level at the design specification modelling level

activities of this modelling level, it does not define how these activities should be performed.

At the **implementation description modelling** level a description of how the system will be configured and how it will work is detailed. Run-time information related to the execution of the business model associated with this system must be defined as must an executable form of functional operations (e.g. code for application programs, scripts[1] for machines and code for human interfaces and database transactions). Although there has been some research effort focused on populating this modelling level [Hou 1993] [ESPRIT/AMICE 1993b], no consensus has yet been reached within the CIM-OSA community as to how it should be implemented.

b. Information view

At the **requirements definition modelling level**, the information view structures the inputs and outputs of enterprise activities and any additional information items relevant to the system into classes (i.e. object views). It also creates an object-oriented model to formalize high level semantic associations with that information. The language to create such a model is based on constructs such as: enterprise objects, object views (of an enterprise object), object relationships, information elements and integrity rules (see Table 1).

At the **design specification modelling level**, the information view structures all system information (based on the semantic data model defined at the previous modelling level), this by using an entity-relationship-attribute (ERA) language (for the description of the conceptual schema and of its external schemata). Database transactions are described as operations performed on the conceptual schema to realise data manipulation (i.e. they are concerned with the dynamics of data). Database transactions can also be viewed as functional operations which are defined within the information view but are used by the function view.

At the **implementation description modelling level**, the information view utilises the relational data model to describe the internal schemata. This is done through two successive schema derivation processes: the first resulting in the logical data model and the second to produce an executable version of the data structure of the internal schema. Moreover, implementation descriptions of the external schemata are provided using a SQL-like data language. Final implementation of the data stores can use any commercially available data model (e.g. flat files, relational models, hierarchical structures and network models).

---

1. Scripts for machines consist of 'macros' which aggregate the commands (which a machine is able to execute) into a functional operation that the related enterprise activity is able to manipulate. A more detailed discussion of scripts is presented later in the thesis.

c. Resource view

The resource view is empty at the **requirements definition modelling level.** However, requirements in terms of capability to be provided by the resources which support business processes and enterprise activities are defined at this level within the function view (see Table 1).

At the **design specification modelling level,** the resource view focuses on providing a detailed specification of resources. Here, the notion is to allow alternative solutions to be tested in respect to their expected capability, by means of simulation. The main construct at this level is the resource which can appear as the following: resource component, functional entity, resource cell and resource set. The precise use of these constructs for resource component specification has yet to be defined by CIM-OSA, although work in this direction is under development elsewhere [Naeger 1993][1].

At the **implementation description modelling level,** the resource view provides run-time information which describes the distribution of resources required to execute enterprise activities and business processes. However, CIM-OSA has yet to prescribe how such a definition can be realised.

d. Organisation view

At the **requirements definition modelling level,** the organisation view defines requirements for the definition of an organisational structure for the enterprise, based on organisational guidelines (i.e. objectives and constraints).

At the **design specification modelling level,** the organisation view assigns tasks related to responsibility and authority to entities of the organisation view (e.g. people, departments, divisions, etc.). Subsequently, this view defines an organisational structure for the enterprise through establishing relationships among its entities. The main constructs manipulated at this level are organisational elements (e.g. people), organisational units, organisational cells (e.g. groupings of people) and organisational relationships (i.e. relationships in terms of authority and responsibility).

The **implementation description modelling level** of the organisation view has yet to be defined in CIM-OSA, although the constructs shown in Table 1 are envisaged.

An illustration of the relationships among some of the constructs manipulated in the four views, across the genericity levels, is shown in Figure 17. this example corresponds to a case study developed at Fiat [Naccari 1994] by the AMICE consortium.

---

1. Definitions for these constructs and the manner in which they were used in SEW-OSA are discussed later in this thesis.

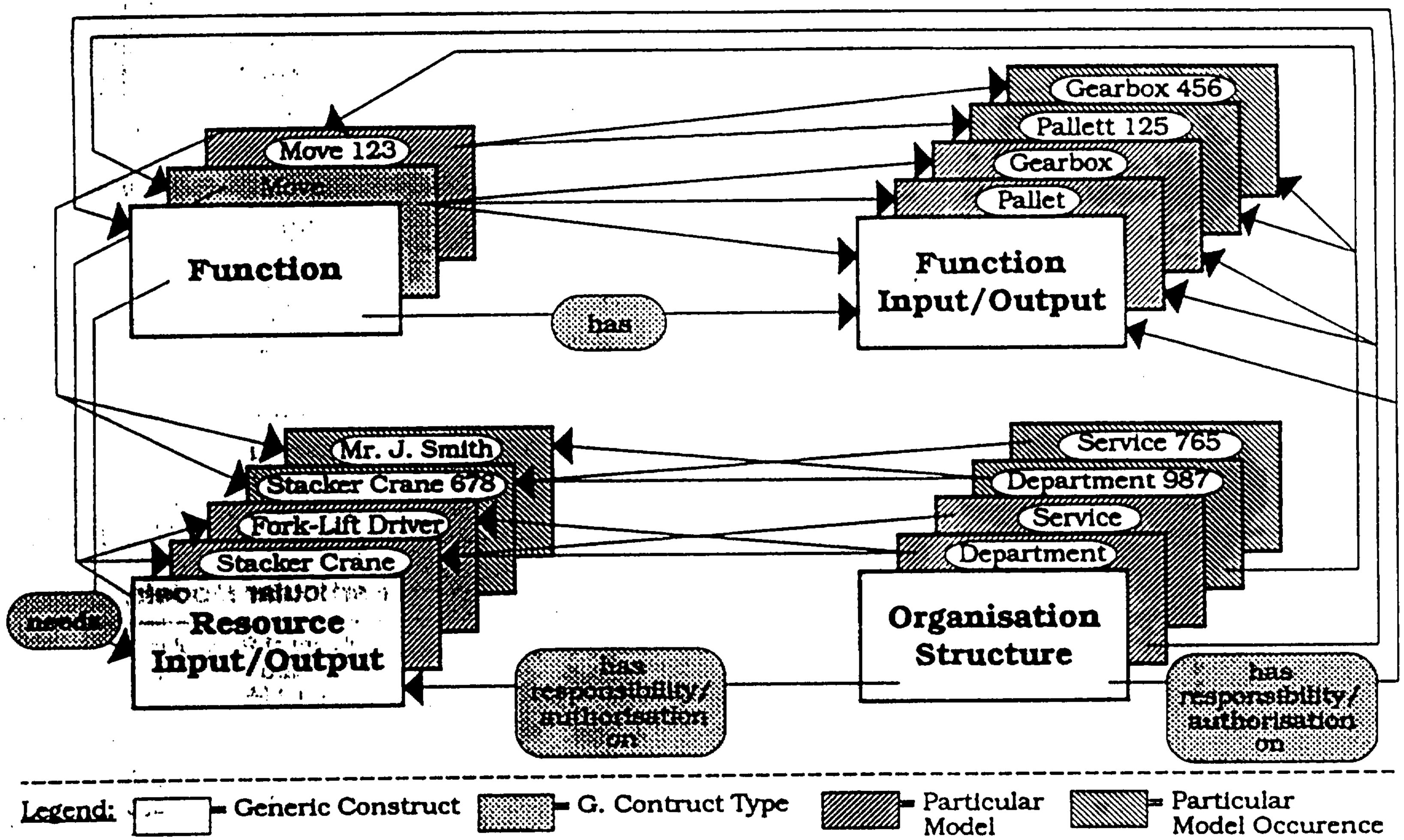


Figure 17 - Relationship among views

4.1.3. The CIM-OSA integrating infrastructure

The models created within the context of the CIM-OSA modelling framework capture information specific to a particular enterprise or type of enterprise. However, CIM-OSA argues that certain functions are generic to any enterprise. Thus, the **integrating infrastructure (IIS)** of CIM-OSA is proposed as a layer of software which aggregates all the IT services and features that are common to all enterprises. As pictorially represented in Figure 12, the IIS is meant to operate as a mechanism for dealing with integration issues associated with an enterprise system (which are captured in models) and reusable, open-ended system components supplied by the market on an 'off-the-shelf' basis.

The IIS is functionally represented as being composed of five separate, but interacting entities, namely: a business entity, an information entity, a presentation entity, a common services entity and a system management entity. These entities are implemented upon an IT infrastructure of programming environments, operating systems, networks, hardware platforms, etc.

The **business entity** includes functions required to control the enterprise operation as described in its business model (i.e. function, resource and organisation models). This entity plays a key role in enabling business integration as represented in Figure 2.

The **information entity** comprises of generic functions for data access, data integration and data manipulation, as defined by the information model. Hence, it provides support to the levels of business, application and physical integration (as depicted in Figure 2).

The **presentation entity** provides means of integrating enterprise components (including legacy components), thereby helping support the levels of application integration. This entity maps the CIM-OSA internal protocol into protocols that are understood by enterprise components (e.g. proprietary machine commands). This provides the remaining entities of the CIM-OSA IIS with a uniform means of interacting with heterogeneous system components. Such a mapping is defined based on the models manipulated by the function and resource views.

The ***common services entity***, as implied by its name, provides common services to the remaining entities. Basically, these services provide an adequate level of distribution transparency (e.g. message passing amongst the several instances of entities, this in a way which is independent of the IT infrastructure upon which the IIS is implemented). This entity helps support application and physical integration (see Figure 2).

The **system management entity** offers generic facilities to configure, set up, maintain and monitor the IT components of the enterprise.

In simple terms, the IIS operation can be explained as follows. The business entity is an engine which executes the business model. As part of model execution, procedural rules of domain processes and related business processes are executed, as is the internal behaviour of enterprise activities. On executing their internal behaviour, enterprise activities request the execution of functional operations. Functional operations are either related to the functionality of resource components or to data-transactions. In the former case, the execution of functional operations is performed through interactions with system components governed by the presentation entity, based on the business model (i.e. models of function and resource views). In the latter case, data access is obtained via the information entity, based on the information model.

The ability to perform these interactions amongst the IIS entities and system components across a distributed system (this regardless of the details of distribution) is enabled by the common services entity.

4.1.4. CIM-OSA environments

CIM-OSA foresees the need for two environments to support enterprise engineering and operation, respectively: the integrated enterprise engineering environment (IEEE) and the integrated operation environment (IEO). The IEEE is where the models supported by the modelling framework are created and tested (i.e. the cube in Figure 12). Within this environment a complete enterprise solution is derived in terms of: (1) the definition of integration issues captured in a variety of models and (2) the specification of components to accomplish the enterprise functions. Once a complete solution is obtained, it is made available to the IEO and put to work upon the IIS.

The actual implementation of the final system comprises of:

- purchasing or developing the required resource components organised in such a way as to comply with the IIS 'modus operandi';
- configuring the system by plugging in its components and the resulting models in their appropriate "sockets" (as illustrated in Figure 12);
- deploying the system for operation.

At the end of the modelling process, it is envisaged by CIM-OSA that the system is structured in such a way that its functionality (i.e. what the system actually does) is made independent of related integration issues. Referring back to Figure 1, this means that the system components encapsulate all the functionality of the system whereas the inter-relationships among system components is captured by models.

4.1.5. Some of the limitations and deficiencies in CIM-OSA

Although CIM-OSA provides a very comprehensive specification for the major aspects that ought to be considered in enterprise modelling, it does not provide solutions for all these aspects. This is particularly the case for function, resource and organisation views at the design specification and implementation description modelling levels and for the integrating infrastructure (see Figure 12).

Additionally, although there are various on-going initiatives (e.g. [Katzy 1994] [Siemens 1994] [ESPRIT/VOICE 1992] [ESPRIT/VOICE 1993] [FIAT 1994a] [FIAT 1994b] [Naccari 1994] [Bruno 1994] [Didic 1992] [Didic 1993]), CIM-OSA has yet to provide an organised method which implements all (or most of) its constructs into wide-scope CASE tools. Wide-scope CASE tools should support a gradual progression of the design process across its modelling levels leading to the generation of a system running upon an integrating infrastructure.

Some of the limitations of CIM-OSA are discussed in the following sub-sections, this as a means of introducing the tools discussed in the remainder of this chapter[1].

a. Lack of an organised method:

The derivation process provides the underlying structure for the definition of a method. However, such a structure does not reach a level of completeness at which gradual progression from requirements definition to physical specification can be achieved. This is evident and is an open issue pointed out in the specifications of CIM-OSA (i.e. [ESPRIT/AMICE 1993a] formal reference base B0-4100 v.5.0 p. 1-27):

> "A clear definition [of links] between requirements level and design level is difficult to establish precisely and is still to be stated [in] CIM-OSA. How complete a requirements definition model should be in terms of functionality is still an open question."

b. Gaps in the modelling levels:

Although the modelling framework of CIM-OSA defines what should be included in each view at each modelling level (i.e. constructs to be manipulated), it does not define how certain tasks are to be performed, in terms of a formal design method.

---

1. Further observations about limitations and deficiencies of the CIM-OSA architecture were analysed by the author in separate documents [Aguiar 1992a] [Aguiar 1992e] [Aguiar 1993a].

This deficiency is recognised by the AMICE consortium when they state that ([ESPRIT/AMICE 1993a] formal reference base B0-4300 v.5.0 p. 1-31)

> "The exact content of an implementation description model has not yet been fully defined."

Additionally, the form by which the design specification modelling level is to be populated is also under investigation, as confirmed in ([ESPRIT/AMICE 1993a] formal reference base B0-4200 v.5.0 p. 1-29):

> "No common agreement has been reached in the project as to whether there are several design specification models at this modelling level or just one."

In this context, the status of CIM-OSA, in terms of methods and tools to support the design process, can be illustrated in the following statement ([ESPRIT/AMICE 1993a] formal reference base B0-4200 v.5.0 p. 1-29):

> "The use of formal techniques and formal languages to support computer processing of design specification models is under investigation. The use of Petri-nets to analyse model behaviour and predict system performance is also being investigated."

In regard to the exact nature of the constructs to be manipulated at this modelling level, ([ESPRIT/AMICE 1993a] formal reference base B12-1300 v.3.0 p. 10-6):

> "CIM-OSA defines an instantiated version of the requirements definition model constructs at the design specification model (e.g. specified domain process, specified business process, etc.). However, 'the added value between requirements definition and design specification constructs are to be demonstrated [...] and redundant constructs are expected to be deleted."

c. Mapping between "design and implementation" and "operation and maintenance":

As stated in previous chapters, enabling such a mapping represents one of the most important objectives of this research. The need for such a mapping is stated by the AMICE consortium in the following statement ([ESPRIT/AMICE 1993a] - Formal Reference Base B0-4300 v.5.0 p. 1-31):

> "How [the] CIM-OSA integrating infrastructure uses the implementation description model contents remains to be defined in detail."

In summary, the usability of CIM-OSA depends upon the availability of tools for model-building (and code generation) across its three modelling levels, as well as of services for the execution of such code across an integrating infrastructure. Neither tools, services nor infrastructure were available at the time that this research started.

SEW-OSA is viewed by the author as the means by which tools, services and infrastructure can be put together in order to overcome some of the limitations of CIM-OSA. Therefore, as discussed in Chapter 3, this research selected the following tools and technologies for the realisation of SEW-OSA:

- CASE technology in the form of a meta CASE tools for building tailored CASE tools which amalgamate the architectures selected;
- the functionality of the CIM-BIOSYS integrating infrastructure to provide part of the functionality envisaged at the level of application integration (as depicted in Figure 2);
- Petri-nets as a formal basis to support modelling the behavioural aspect within function view at the design specification and implementation description modelling levels of CIM-OSA;
- object-oriented design methods to support the description of interactions among system components in the function view at the design specification and implementation description modelling levels of CIM-OSA;
- an IDEF0-based functional modelling tool to organise the functional aspects of the case study data whilst the SEW-OSA CASE tool was being development (see discussion later in this thesis).

The proprietary tools and services selected and used in this research to support this activity included: CASE tool technology supplied in the form of the IPSYS-ToolBuilder meta CASE tool, the BuilderXcessory interface generator for developing software applications running in the X-Windows/Motif programming environment, the ARP (an analysis and simulation tool for Petri-nets) and the Design/IDEF modelling tool. A detailed explanation of the decisions related to the selection of these tools and services is presented later in this chapter. Following is a brief description of the primary features of the proprietary tools and technologies selected.

4.2. CASE Tools

Two CASE tools have been used to support the realisation of SEW-OSA, the ToolBuilder meta-CASE tool and the BuilderXcessory interface generator. Details about BuilderXcessory are included in the Appendix 3, whilst the meta-CASE is briefly discussed as follows.

A meta CASE tool facilitates the process of capturing and formalising part of the design process in software tools, by providing a highly reconfigurable CASE tool environment used to develop other CASE tools [Endres 1991]. Based on an evaluation of currently available meta CASE tool products (conducted by I. S. Murgatroyd), ToolBuilder [Alderson 1991] was selected as the most suitable for the purpose of encapsulating the formalism of the selected architectures[1].

ToolBuilder is comprised of the building blocks shown in Figure 18. A CASE tool is built using ToolBuilder by defining data associated with each of those building blocks. At the heart of a CASE tool so built is the **CASE Tool Structure**, formalised in a Backmann diagram (from [Kay 1993])[2]. This structure defines the main constructs to be manipulated in the model and the relationships among them. An entity in the Backmann diagram can represent a CASE tool diagram, an object manipulated in a diagram or in a hyper-text template, any of which can represent the CIM-OSA constructs shown in Table 1. Relationships define how entities are related to one another (i.e. how they are inter-linked or how navigations can be performed between them).

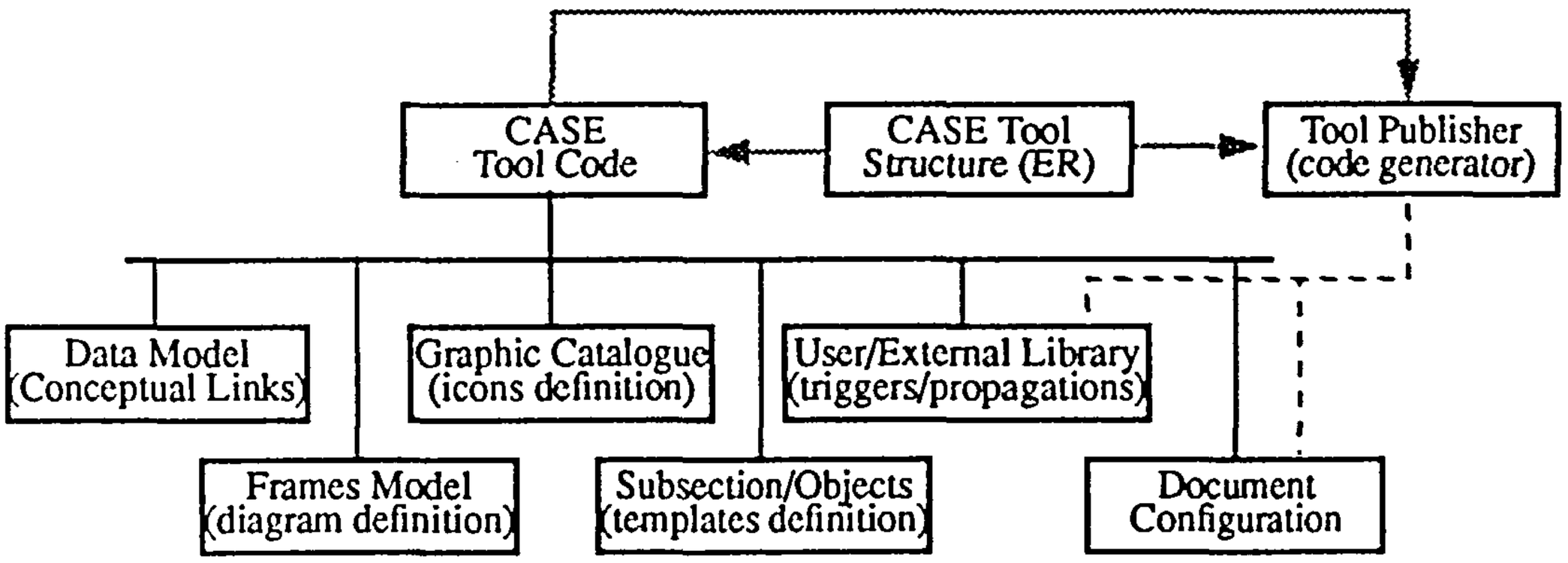


Figure 18 - IPSYS-ToolBuilder meta-CASE tool

Based on such a structure, the descriptions of each construct or diagram are

---

1. This evaluation also considered a comparison (recently updated [Howard 1994]) performed by an independent consultant which appointed ToolBuilder as the best meta-CASE tool in the market.
2. A Backmann diagram represents entity types and relationships of ownership among them (of the type used to define the IMS life cycle in Figure 7).

fleshed out, so that the **CASE Tool Code** can be generated in a structured manner. A detailed description of each of the remaining modules depicted in Figure 18 is presented in the Appendix 3.

A CASE tool to amalgamate the selected architectures can be built by populating the elements of the meta-CASE tool shown Figure 18 with constructs extracted from the architectures. That is, the methods embodied in the architectures need to be formally described within the meta-CASE tool. Further information on how such a formalisation process occurs can be obtained from the appropriate ToolBuilder manuals [Ipsys 1992a] [Ipsys 1992b] [Ipsys 1992c] [Ipsys 1992d] [Ipsys 1992e] [Ipsys 1992f] [Ipsys 1992g].

4.3. The CIM-BIOSYS Integrating Infrastructures

The implementation of the environment depicted in Figure 9 has its basis on CIM-BIOSYS (CIM Building Integrated Open SYStems), an integrating infrastructure developed by the MSI Research Institute, which provides means of building integrated manufacturing systems.

Basically, the CIM-BIOSYS infrastructure was selected to overcome the unavailability of an integrating infrastructure for CIM-OSA, as: (1) it provides a minimum level of distribution transparency required for concentrating the research on the issues of business integration; and (2) it already implements part of the functionality of some of the entities of the CIM-OSA integrating infrastructure.

4.3.1. Distribution transparency

Brenner [Brenner 1993] defines seven kinds of distribution transparency, namely:

- **Access transparency**: concealing the access mechanism (e.g. communication network) involved in the interactions among system components;
- **Location transparency**: concealing detail about location of components (e.g. host computer);
- **Migration transparency**: concealing dynamic relocation of components (i.e. a dynamic kind of location transparency);
- **Liveliness transparency**: concealing the form by which the components reside (i.e. migration between an active form in memory and a passive form on data stores;
- **Replication transparency**: concealing the existence of replicas of components as well as the mechanism to support replication and synchronisation;
- **Concurrence transparency**: concealing the sharing of services among

components, by avoiding mutual interference;

- **Failure transparency**: concealing partial completion of transactions through all-or-nothing mechanisms (i.e. atomic transaction).

The CIM-BIOSYS infrastructure provides access transparency and location transparency which are considered by the author to be sufficient to support model-enactment at the current stage[1].

4.3.2. IIS entities

CIM-BIOSYS provides an open approach to resolving issues of data fragmentation, inter-process communication and interaction in manufacturing environments, which typically comprise a distributed and heterogeneous set of process.

Essentially, the CIM-BIOSYS infrastructure provides integration services to (support) software applications in an "open" manner. Here, applications only need to have knowledge of how to use CIM-BIOSYS services, with the integrating infrastructure taking responsibility for dealing with configuration issues.

CIM-BIOSYS can be used with a family of system build services, which includes: templates to deal with proprietary (non-conformant) manufacturing machines (i.e. device drivers) and software packages (i.e. alien applications); this providing means of interacting with legacy components.

Figure 19 [Coutts 1992] depicts an overview of the functionality of the CIM-BIOSYS integrating infrastructure[2]. This figure details the four principal functional elements or managers of CIM-BIOSYS, namely: the service manager, the driver manager, the run-time manager and the configuration manager. Details about these elements are also included in Appendix 3.

Basically, the service manager provides the services termed in the CIM-OSA IIS as the common services entity, in addition to embracing some of the functions of the information entity. The driver manager contributes with part of the functionality of the presentation entity. The run-time and the configuration managers populate part of the system management entity. Although the CIM-BIOSYS infrastructure supports a considerable amount of the functionality of the CIM-OSA IIS, it does not yet support the services of the business entity. Neither does it completely populate the presentation and information entities. Thus, this research seeks to provide complementary services

---

1. It is important to notice that the motivation for the realisation of the CIM-BIOSYS infrastructure was not purely to provide these levels of transparencies.
2. Much richer descriptions of CIM-BIOSYS and some of its applications are developed by Shaharoun [Shaharoun 1994], Edwards [Edwards 1993], Singh [Singh 1994], Coutts [Coutts 1992], Gascoigne [Gascoigne 1992] and Gilders [Gilders 1991a] [Gilders 1991b].

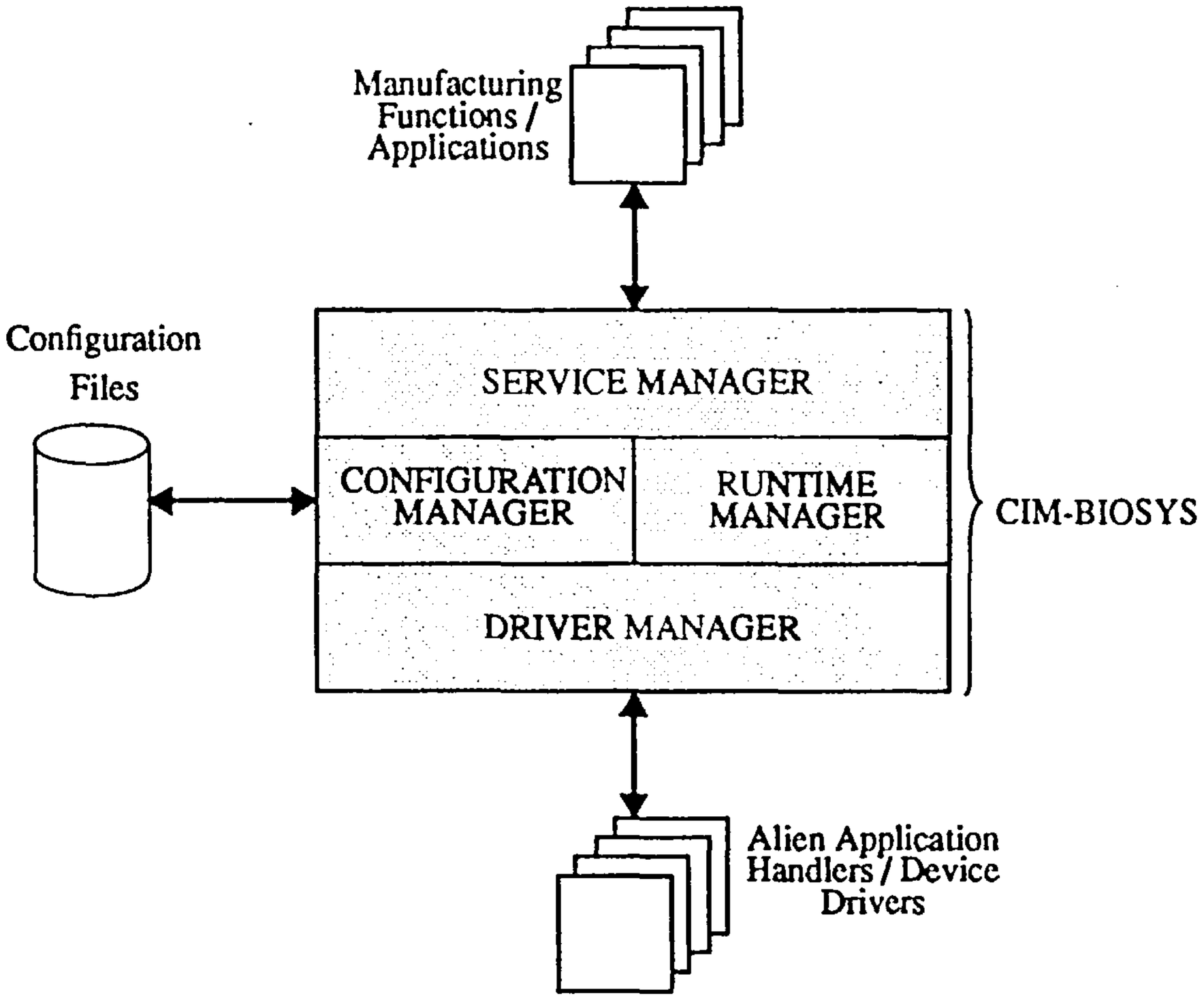


Figure 19 - A functional view of CIM-BIOSYS

for the CIM-BIOSYS infrastructure by adding to it elements of the CIM-OSA IIS entities.

4.4. Petri-nets

According to David [David 1994], Petri-nets present two interesting features:

> "Firstly, they make it possible to model and visualise behaviour comprising concurrence, synchronisation and resource sharing. Secondly, the theoretical results concerning them are plentiful. The tool enables qualitative analysis and its numerous applications have been still further added to by a number of research works to enable more condensed descriptions, including where the time factor intervenes."

Petri-nets have a number of extensions [Peterson 1981]. Two extensions of particular interest in this research are: generalised stochastic time Petri-net and predicate-action Petri-nets. An introduction to Petri-nets and their extensions is presented in Appendix 4.

Petri-nets are proposed to be used in the model-building capability as a language to describe the internal behaviour of enterprise activities. The adoption of

Petri-nets is also proposed as one of the means by which model-enactment is to be achieved[1]. Petri-nets can be viewed as a kind of 'assembly' language of modelling. Most modelling languages represent static descriptions that cannot be enacted directly. Petri-nets offer a means by which these static descriptions can be enacted.

Generating Petri-net models has also enabled the association of SEW-OSA with available tools capable of analysing, simulating and executing Petri-nets. A number of software tools available in the market or from academic institutions incorporate an engine for Petri-net execution to support this capability. For reasons of availability, the tool named ARP was adopted to be linked to SEW-OSA.

ARP[2] is a software tool developed by the LCMI/UFSC ("Laboratorio de Controle Microinformatica da Universidade Federal de Santa Catarina", Brazil) [LCMI 1989]. Use of this tool has enabled: (1) editing the business model generated by SEW-OSA (so that minor changes in the model can be made to suit simulation tests without re-generating the Petri-net model); (2) analysis of the basic properties of a Petri-net (i.e. limitation, conservation, livelihood, multi-sensibilisation, re-initialisation, live-locks and deadlocks); (3) analysis of invariants (of places and transitions); (4) step-by-step simulation of the Petri-net state evolution; and (5) performance analysis of certain parameters of the system, based on its execution in simulated time (i.e. dynamic simulation of its state evolution).

4.5. Object-Oriented Design

According to Adiga [Adiga 1993], object-orientation is a new computing paradigm which encapsulates data and procedures in the form of an "object" (where an object is an entity that exists at a level of abstraction higher than data and procedures). Data and procedures encapsulated by an object can only be accessed by external objects by means of messages exchanged between them. The benefits of object-orientation stem from the concepts of class (enabling inheritance of object features from parent objects), encapsulation (enabling information hiding) and polymorphism (enabling dynamic binding of data and procedures).

A decision to adopt concepts of object-orientation within SEW-OSA was made as a means of populating its design specification and implementation description modelling levels. It was envisaged that object descriptions could be used to model system resources and enable resource selection based on functional requirements, this to facilitate mapping between the *conceptual analysis* and *design and implementation*

---

1. The other means by which model-enactment was achieved was through direct interpretation of the CIM-OSA models by a business entity component (discussed later in this thesis).
2. ARP stands for Petri-net Analyser and Simulator (in Portuguese).

phases of the IMS life cycle.

No particular method was adopted and used to realise the object descriptions supported by SEW-OSA. However, the Booch [Booch 1991] method was adopted as the basis of another "Model-Driven CIM" tool with which the SEW-OSA CASE tool was designed to interact (i.e. a resource modelling tool developed by I.S. Murgatroyd [Murgatroyd 1993]).

4.6. IDEF and Design/IDEF

IDEF0 is a modelling language that evolved from SADT (Structured Analysis and Design Technique) which was developed by SofTech [SofTech 1976] as part of a commission by the US Air Force [Bravoco 1985a]. IDEF0 consists of a functional modelling language which enables modelling a system based on a structured decomposition of its functions. Such a functional decomposition can be used to capture the flows of information, material and control through the functions, as well as the resource requirements associated with each function.

In this research, it was envisaged that IDEF0 would be used as a substitute for the SEW-OSA functional modelling facilities. Indeed, this option was taken during the data gathering processes of the case study application of SEW-OSA (whilst the SEW-OSA CASE tool was being developed). Thus, IDEF0 models of the shop-floor were created as an intermediate step towards creating SEW-OSA-based models.

Design/IDEF, which is a tool for IDEF0 modelling supplied by Meta Case Technology, was chosen and used to create these models. Design/IDEF enables the creation of models using the following languages [Meta 1990]: IDEF0, IDEF1, IDEF1X and entity-relationship-attribute descriptions. Design/IDEF also possess a facility that enables models created in IDEF0 to be fed into Design/CPN, a tool capable of enacting IDEF0 models by associating its functional blocks with transitions of a coloured Petri-net (CPN).

4.7. A Strategy for the Realisation of SEW-OSA

The strategy adopted for realising the structure proposed in Figure 9 for SEW-OSA involved the following sets of activities:

a. **Realisation of a CASE tool for SEW-OSA**, which is envisaged to involve:

- a thorough study and review of the specifications of the CIM-OSA architecture and adoption of alternative solutions for those parts of CIM-OSA that have yet to be defined;
- the implementation of part of the CIM-OSA modelling methodology in

combination with predicate-action Petri-nets and an object-oriented representation in a CASE tool, through the use of a meta CASE tool.

b. **Establishment of links between the SEW-OSA CASE tool and other "Model-Driven CIM" tools and services,** which is envisaged to involve:

- the provision of interfaces with a set of information modelling tools which structure the information shared by system components;
- the use of an object-oriented resource model to facilitate the specification of system components.

c. **Establishment of a link between the SEW-OSA CASE tool and a simulation tool,** which is envisaged to involve:

- resolution of syntactic differences between CIM-OSA and a generalised stochastic time Petri-nets (GSTPN) through a mapping between these two languages;
- incorporation of such a mapping in the SEW-OSA CASE tool, so that code in a GSTPN format can be generated from a CIM-OSA model;
- use of this code to feed ARP (i.e. the simulation tool) which enables analysis, simulation and performance evaluation of the model.

d. **Development of a business entity for CIM-BIOSYS,** which is envisaged to involve:

- the realisation of a model-enactment capability for driving the interactions between system components;
- provision of a capability for the generation of rapid-prototypes of system structure and its components.

e. **Specification of a presentation entity for CIM-BIOSYS,** which is envisaged to involve:

- definition of how system components should be organised in order to be integrated to the system structure generated by SEW-OSA;
- the definition of interfaces associated with the various forms of system components, namely: device drivers (for machines), application enablers (for application programs) and human interfaces (for human beings)

4.8. Structure for the Decisions Made in Realising SEW-OSA

Figure 20 illustrates the main decisions made within the context of the "Workbench Development" box depicted in Figure 11. Basically, six primary decisions were made which led to the realisation of SEW-OSA. Their rationale and alternative courses of action which could have been taken are briefly explained below.

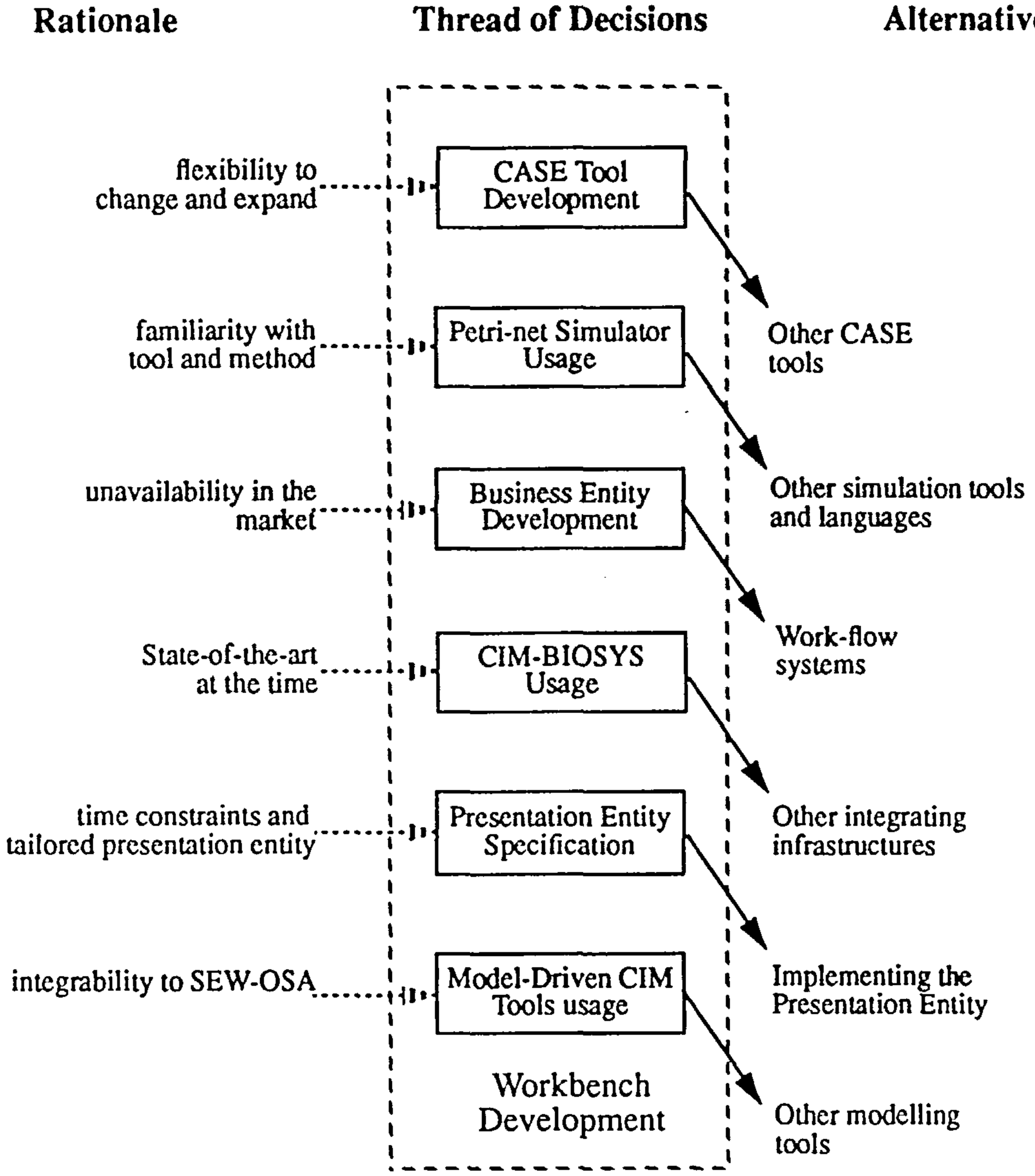


Figure 20 - Decisions as part of the "Workbench Development"

The realisation of a CASE tool to encapsulate the selected architectures was proposed due to the unavailability of alternative tools from the market. However, it was also understood that the author would access important pre-requisite knowledge related to mappings between life-cycle processes through achieving a consolidation and extension of existing architectures in a CASSE environment. Additionally, the use of a meta-CASE tool for such a realisation adds the essential flexibility required to incorporate future changes in the architectures, as well as for adding formalism from

other architectures. This could not have been achieved through acquiring CASE tools in the market.

The particular Petri-net simulator adopted was selected based on its availability, as well as on the familiarity that the author had with the tool (due to using it within the context of his M.Sc. dissertation [Aguiar 1989]). Other Petri-net engines or, indeed, other types of simulators could have been used. However, Petri-nets provide a unique feature not available in other simulation tools, namely: the support for qualitative analysis of systems (e.g. existence of deadlocks, livelocks, etc.).

The development of a business entity for CIM-BIOSYS was required as software was not yet available in the market to realise its purpose. Work-flow systems have improved their functionality over the last three years and now provide part of the functionality of the business entity (e.g. the ProcessWise integrator [ICL 1993]). However, achieving their integrated use with the CIM-BIOSYS infrastructure would present other problems that still remain to be solved.

In regard to the development of the business entity, the enabling tools used were the programming utilities traditionally used in the MSI Research Institute (e.g. X-Windows Motif [Heller 1991], BuilderXcessory [ICS 1991a] [ICS 1991b], Unix programming tools [Back 1986], etc.). A formal description technique (FDT) (e.g. Estelle [ISO 1988]) was not used to implement the business entity, because no FDT was available at the time that the research started which could generate code for the CIM-BIOSYS infrastructure. As a result of the work developed in the "Model-Driven CIM" project, an Estelle-based tool is now available which can be used for that purpose (i.e. the tool developed by P. Gilders [Gilders 1995] based on the environment provided by NIST [Sijelmassi 1991a] [Sijelmassi 1991b]).

CIM-BIOSYS was adopted as the integrating infrastructure because it was the most comprehensive available at the time that this research started, thus comprising the state-of-the-art in the area [Singh 1994]. Subsequently, a number of alternative forms of integrating infrastructures emerged which could be used as a replacement for CIM-BIOSYS, but technical problems of a similar nature to those tackled in the study would likely have arisen. In this respect, the CIM-BIOSYS infrastructure was 'accessible' to modification and well understood by other MSI researchers, whereas other integrating infrastructures have yet to prove sufficiently comprehensive or stable to be used in this type of work. Some of these alternative forms of integrating infrastructures are mentioned in Section 2.2.5.

A presentation entity was proposed to be specified in this research, although not implemented, due to time constraints and because other research groups known to the author [ESPRIT/VOICE 1992] [Didic 1992] were dedicating effort in this particular area. Hence, the issues dealt with in the presentation entity specification were those required to achieve integration among system components within the context addressed

by this research.

Other tools produced by other MSI researchers within the "Model-Driven CIM" project were used in association with SEW-OSA, in order to model aspects of a system which are not covered by SEW-OSA. These tools were used due to the possibility of completely integrating them into SEW-OSA at a later stage, thereby constructing a wider scope workbench.

4.9. Concluding Remarks

Figure 21 shows the areas of the CIM-OSA cube that have been covered in SEW-OSA by using the architectures, tools and services discussed in this chapter. Basically, efforts were concentrated in providing a working structure for CIM-OSA's function and resource views. It should be noted from this figure that SEW-OSA does cover completely all constructs and recommendations of CIM-OSA at the design specification and implementation description modelling levels. Such a constraint stems from the fact that the specifications of CIM-OSA define the relevant constructs and functions of each view at each modelling level, but they do not define how they should or could be implemented, thus, requiring solutions alternative to CIM-OSA. Therefore, the CIM-OSA definitions were enhanced by SEW-OSA with the introduction of constructs borrowed from object-orientation and Petri-nets.

An overview of where each solution adopted fits in each view at each modelling level is presented as follows.

4.9.1. Model-building capability

The model-building capability of SEW-OSA is envisaged to aggregate the constructs of CIM-OSA with object-orientation and Petri-nets in order to support the modelling process at the requirements definition and design implementation modelling levels. This support can be achieved as follows.

a. Requirements definition modelling level:

The function view is proposed to be completely implemented based on the CIM-OSA specifications available to the author [ESPRIT/AMICE 1993a].

b. Design specification modelling level:

The function and resource views at this modelling level is envisaged to combine CIM-OSA with an object-oriented description to define the interactions between enterprise activities and the active resource components (i.e. system components) that provide the functionality modelled in the enterprise activities.

For the resource view at this modelling level, models of alternative solutions in

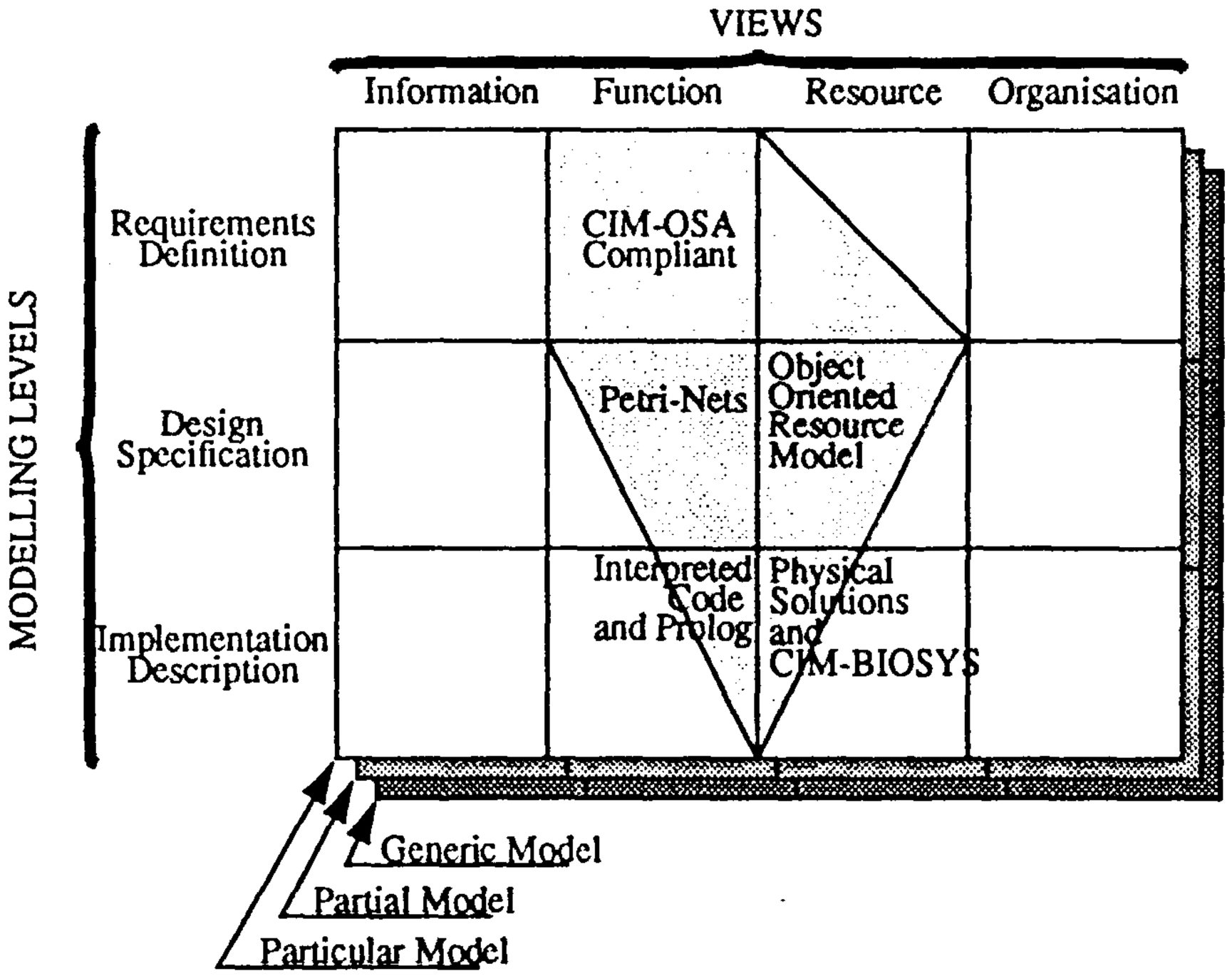


Figure 21 - Scope of the SEW-OSA CASE tool against the CIM-OSA cube

terms of active resource components available from a library of partial models (i.e. resource models), are envisaged to be used to configure a system.

4.9.2. Model-enactment capability

Basically, the information formalised in the business model with the use of the SEW-OSA CASE tool should then be passed to the model-enactment capability which should manipulate it by means of simulation and rapid prototyping (in order to test it). This is envisaged to be achieved through populating the design specification and implementation description modelling levels of CIM-OSA with the following facilities.

a. Design specification modelling level:

This modelling level is envisaged to embrace a two-staged process, namely:

- analysis of the business model; simulation of its dynamic behaviour; and evaluation of the system performance, in order to obtain metrics from the models (i.e. average, maximum and minimum time values, queue sizes, etc.).
- rapid-prototyping of the system and its (emulated) components[1] (i.e. code generation for the system structure and for its components), in a form that can be

1. The term "emulated component" is used to mean an executable specification of the internal behaviour of the component which excludes any detail of its actual functionality.

executed by the integrating infrastructure, this in order to test a particular solution in terms of system configuration (see Figure 9).

b. Implementation description modelling level:

This level should involve the configuration of the physical system. This is envisaged to be accomplished by gradually replacing its (emulated) components by the physical ones (i.e. machines, application programs and human beings).

Although the strategy for populating views and modelling levels may convey the idea that the design process is to occur in a 'water fall' manner, SEW-OSA should offer support for iterating through modelling, analysis, simulation, rapid-prototyping, configuration and operation of an IMS (i.e. iterations between model-building and model-enactment) a number of times before a complete system is finally deployed. In this respect, an extremely valuable contribution of SEW-OSA is expected to come from facilitating the execution of these iterations in a consistent manner.

The remaining chapters of this thesis describe the development, application and evaluation of a workbench embracing a model-building capability and a model-enactment capability. These capabilities are materialised within SEW-OSA. Hence, SEW-OSA is a provider of facilities for modelling, analysis, simulation, rapid-prototyping, configuration and operation of an IMS, this as a first step towards an incremental approach to formalising the activities across different phases of the IMS life cycle.

A similar organisation is followed in each chapter which describes: how related capabilities of SEW-OSA support these facilities, how these capabilities were implemented and what their main limitations and contributions are (in the light of related contemporary research work).

Chapter 5 - Model-Building Capability

This chapter describes the implementation of the CASE tool as the core element of the model-building capability of SEW-OSA. The structure of the CASE tool is presented by means of a top-down description of the method embedded in it to carry out the modelling process.

The method is illustrated through samples of diagrams, templates, reports and code generated (which use part of the data gathered in the case study application of SEW-OSA discussed later in the thesis).

Although a considerable level of detail was included in this chapter, the importance of this detail is twofold:

- it defines clearly how the CIM-OSA constructs were used within SEW-OSA; and
- it presents the essential modelling concepts realised by SEW-OSA which will be used to explain the research activities described in the forthcoming chapters.

5.1. Overview of the Design Methodology

A valuable contribution of this research work is the definition of an organised method for the application of CIM-OSA. Basically, as discussed in Chapter 3, SEW-OSA implements such a definition by providing two classes of capability associated with its design methodology: the **model-building capability** and the **model-enactment capability.** Each of these capabilities supports more than one modelling level of CIM-OSA. At each modelling level, a number of constructs are manipulated in the form of diagrams and templates which are used to capture the information relevant to the design process. Essentially SEW-OSA enables the application of the model-building capability depicted by Figure 22 to generate models which can be used by the model-enactment capability illustrated in Figure 9.

Following is an overview of the model-building capability. Correspondence between the acronyms used in the text below and the constructs manipulated in each diagram of the SEW-OSA CASE tool is indicated in the model example represented in Figure 22.

a. Requirements definition modelling level:

- **Context diagram.** This diagram defines the domains (i.e. main areas of an enterprise) under consideration, and the relationships between the domains. One context diagram is created for each individual enterprise model.
- **Domain diagram.** This diagram defines the major domain processes of a domain.

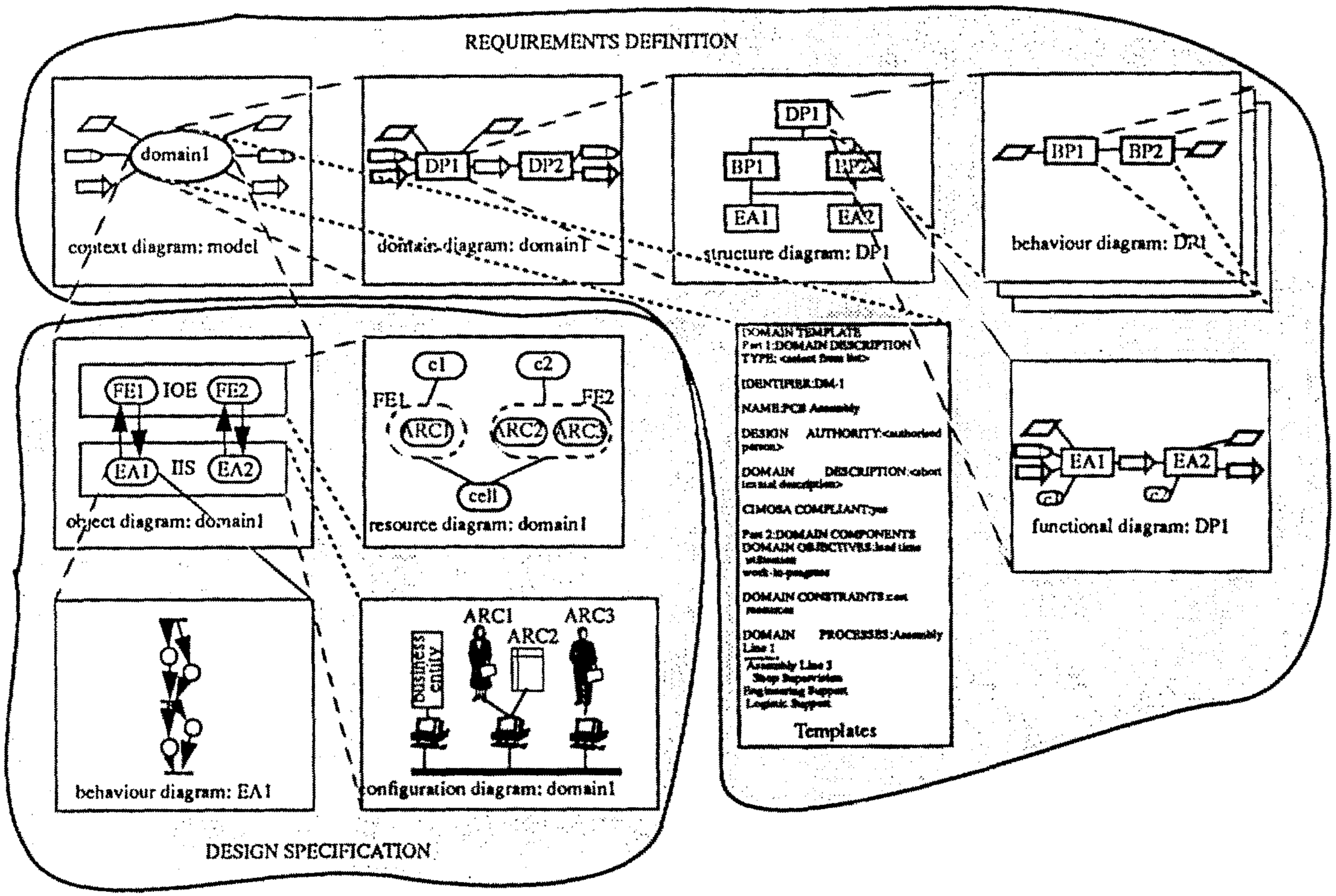


Figure 22 - SEW-OSA model-building capability

One domain diagram is created for each CIM-OSA-compliant domain[1].

- **Structure diagram.** This diagram defines the functional decomposition of a domain process[2] in terms of enterprise activities and business processes[3] in a structured manner. One structure diagram is created for each domain process.
- **Behaviour diagrams.** These diagrams define the flow of control used to execute the functionality of a domain process and its business processes. Behaviour diagrams can also be referred to as process diagrams. One behaviour diagram is created for each domain process and subsequently for business processes within the domain process structure.
- **Functional diagram.** This diagram defines flows of material, information and control through the atomic building blocks of the domain process (i.e. enterprise activities[4]). One functional diagram is created for each domain process.

b. Design specification modelling level:

- **Object diagram.** This diagram defines the flow of messages between functional entities[5], the business entity (i.e. enterprise activities) and the information entity of CIM-OSA. One object diagram is created for each CIM-OSA-compliant domain.
- **Activity behaviour diagram.** This diagram defines how the enterprise activity uses

---

1. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], domains "identify well-defined, totally integrated, functional areas of the enterprise". A domain maps into the concept of "enterprise domain", illustrated in Figure 7. Domains are modelled either as CIM-OSA-compliant domains or non-CIM-OSA-compliant domains. A CIM-OSA-compliant domain identifies the area to be engineered within the enterprise. A non-CIM-OSA-compliant domain identifies other areas with which the area to be engineered interacts.
2. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], domain processes (or DP's) "are high-level processes [...] triggered by some events and producing a defined end result (function output). Domain processes are at the level of functional decomposition of domains. They must be triggered by nothing else than events" [ESPRIT/AMICE 1993a]. Domain processes can also be viewed as objects which communicate via exchange of information, material and events.
3. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], a business process (or BP) "is a sub-process of a domain process. It cannot be directly triggered by events and is always called by a parent process." A business process works as an intermediate construct between domain processes and enterprise activities.
4. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], enterprise activities (or EA's) "describe [the] basic enterprise functionality (i.e. things to be done). They are defined by their function input, function output, control input, control output, resource input, resource output and ending status and have no behaviour defined at the requirements definition modelling level. They are always called by a parent process."

(via message exchanges) the integrated operation environment (i.e. a grouping of functional entities, or FE's), in order to perform its basic functionality (i.e. functional operations[1]). One activity behaviour diagram is created for each enterprise activity.

- **Entity behaviour diagram.** This diagram defines the expected external behaviour of a functional entity as perceived by the enterprise activities. Such a description is used to emulate the behaviour of an active resource component during the rapid-prototyping stage of a system. One entity behaviour diagram is created for each functional entity.

- **Resource diagram.** This diagram specifies instances of active and passive resource components (i.e. ARC and PRC)[2] associated with the classes specified by their functional entities which are able to execute the functional operations required by an enterprise activity. These functional operations are related to the capability required by the enterprise activity; the resource capability being defined in the functional diagram (see Figure 22). One resource diagram is created for each integrated operation environment associated with a particular domain (stemming from the IOE construct, as illustrated in Figure 22).

- **Configuration diagram.** This diagram defines the computer configuration of the system (i.e. where each active resource component will be executed or interfaced with). One configuration diagram is created for each segment of the IIS which serves a particular enterprise domain.

A hyper-text template is associated to each symbol represented in the diagrams

---

5. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], a functional entity "is a resource able to perform, completely on its own, a (class of) functional operation(s)". functional entities are viewed in this thesis as functional representations of active resources components required to fulfil the capability associated with enterprise activities identified in the functional diagram.

1. According to the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], a functional operation "is a basic unit of work defined at the design specification modelling level (i.e. lowest level of granularity in the function view). At run-time, [a functional operation is] fully executed or not at all."

2. An active resource component identifies a component of a system which is able to execute functional operation(s) on its own. It can also be a modelling description which characterises either a human being, an application program or a machine that possess a computerised controller (i.e. human functional entity, application functional entity and machine functional entity [ESPRIT/AMICE 1993a]). A passive resource component is an object used by the active resource component when performing functional operations.

of Figure 22 which enables the user to define all its attributes.

In the next sections, the modelling method implemented in the SEW-OSA CASE tool is described, as if the designer were modelling a system for a "green-field" site (i.e. design from 'scratch'). This consists of populating the particular level in the CIM-OSA modelling framework which, in this case, will be done regardless of the instantiation dimension (see Figure 12). The sequence of design steps (i.e. diagrams and templates to be completed) as illustrated in Figure 22, defines the path through which the CASE tool guides the design process.

5.2. Requirements Definition Modelling Level

A new design would usually start with the user[1] browsing through a library of reference models (i.e. "system models" in Figure 7), in order to find previously created designs that can be instantiated for the particular problem that is being addressed. However, the description presented here will assume that no reference model is available to be instantiated. Therefore, the model-building process starts with the definition of a context diagram.

5.2.1. Context diagram

The context diagram contains the information described in Section 4.1.2, as the "enterprise level". A context diagram represents the major domains of concern in an enterprise model, as well as key relationships between those domains. Relationships characterise the interface between domains by defining the type of constructs exchanged between them. These constructs can be events (EV's) and object views (OV's), in the form of information OV's or physical OV's (e.g. material flow). Figure 23 shows an example of a context diagram[2]. Whilst at the stage of defining a context diagram, the SEW-OSA CASE tool offers a user the following options:

- to add domains, with a choice of CIM-OSA-compliant or non-compliant;
- to add major (physical and information) object views and events that flow amongst domains;
- to add relationships between domains which summarize the meaning of the

---

1. The term "user" is used here to refer to the person who is using SEW-OSA to formalise requirements. The term "designer", which appears later in the thesis, refers to the person who is using SEW-OSA to design a system (in order to address these requirements). These two perspectives are represented in Figure 22 by the two 'bubbles'.
2. The diagrams and templates presented in this chapter were printed directly from the SEW-OSA CASE tool.

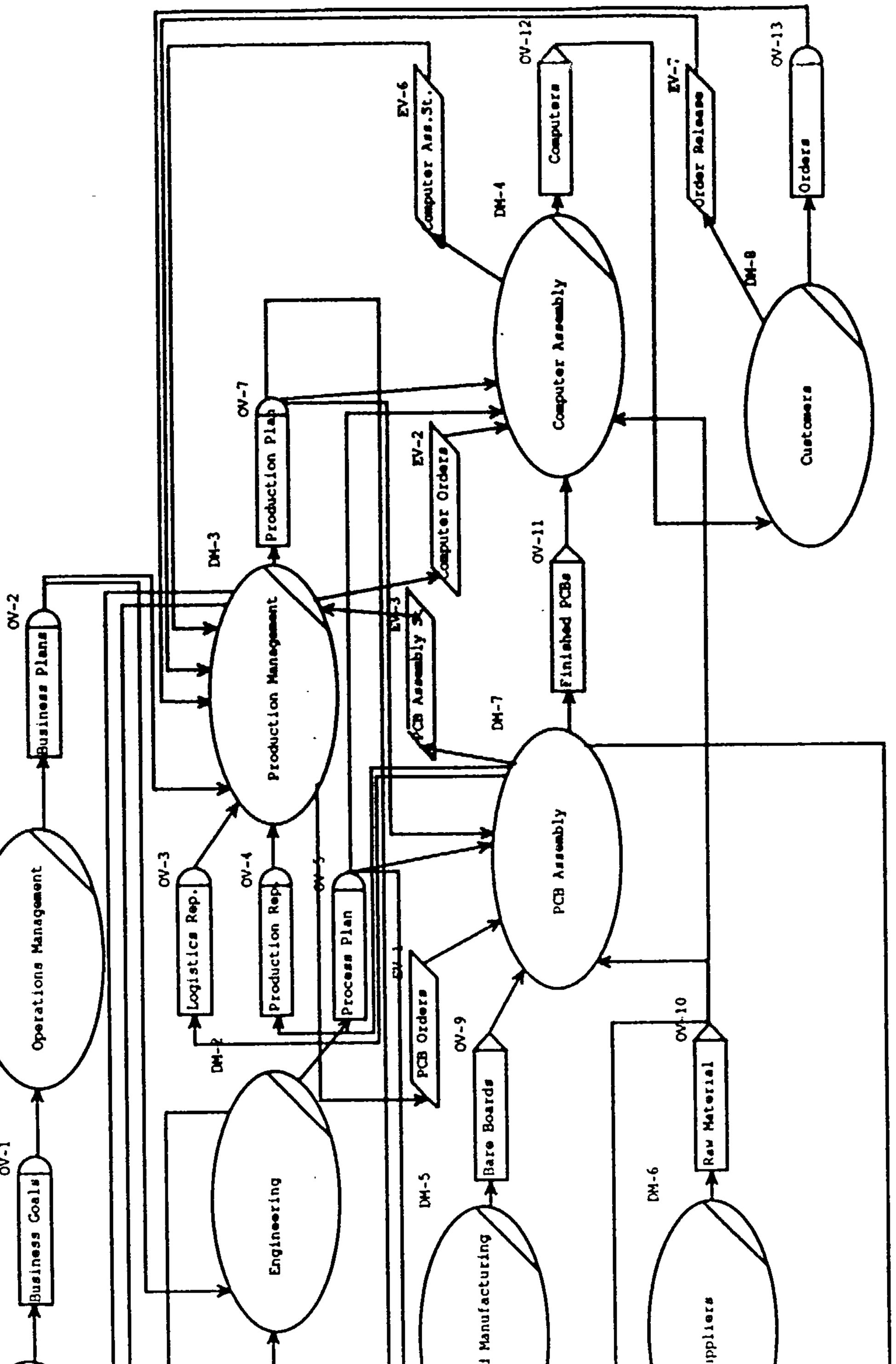


Figure 23 - Example of a context diagram

interactions between them.

For each construct, additional information about the construct's attributes can be included by selecting an object-operation[1] on the object representing the construct. Such an object-operation triggers the CASE tool to display the construct template (i.e. its hyper-text description). An example of a domain template is shown in Figure 24. This option is not only available in the context diagram but in all diagrams. Indeed, text templates are the main means by which the formalism of CIM-OSA was incorporated into the CASE tool.

Text templates are extremely important as means of documenting the design process. However, SEW-OSA has implemented these templates in such a way that only a minimum amount of information is required in order to enable model-enactment. This information consists basically of the attribute "IDENTIFIER" (e.g. DM-1 in Figure 24) which is, in any case, generated automatically by the CASE tool when the construct is created. The attribute "NAME" is also desirable for improved readability during the model debugging process. This means that a model can be created in SEW-OSA simply by manipulating graphical descriptions.

Hyper-text templates also enable the execution of navigations[2] between diagrams and templates, or any other object-operation to be triggered by selecting the appropriate field in the template (i.e. context sensitive fields). For instance, by selecting the name of an objective or construct in the domain template (see Figure 24), the user may navigate to the objective or constraint template. Additionally, the hyper-text template presents the user with the relevant data that has already been entered elsewhere in the model or any data that can be extracted or deducted from a diagrammatic description.

Once this diagram is complete (with all constructs and their attributes defined) the user may proceed to define a domain diagram associated with each CIM-OSA-compliant domain. At this point, the 'process level' model can be created (see Section 4.1.2).

5.2.2. Domain diagram

A domain diagram represents the domain processes (DP) contained within a CIM-OSA-compliant domain. This diagram also describes the relationships between

---

1. The term "object-operation" is used by ToolBuilder to denote the actions that are available in a "pop-up" menu associated with each object represented in a diagram, when the operator selects it with the computer 'mouse'.
2. In database terminology, a navigation consists of moving the context of processing from one entity to another along a relationship.

DOMAIN TEMPLATE	
Part 1:	**DOMAIN DESCRIPTION**
TYPE:	<select from list>
IDENTIFIER:	DM-1
NAME:	PCB Assembly
DESIGN AUTHORITY:	<authorised person>
DOMAIN DESCRIPTION:	<short textual description>
CIMOSA COMPLIANT:	yes
Part 2:	**DOMAIN COMPONENTS**
DOMAIN OBJECTIVES:	lead time utilisation work-in-progress
DOMAIN CONSTRAINTS:	cost resources
DOMAIN PROCESSES:	Shop Supervision Assembly and Test
BOUNDARY:	
OBJECT VIEWS:	Process Plan Bare-Boards Raw Material Production Plan Finished PCBs Engineering Rep. Production Rep. Logistic Rep.
EVENTS:	PCB Orders PCB Assembly St.

Figure 24 - Example of a domain template

DP's, in terms of flows of events and object views. A domain diagram is created from an object-operation selected from a CIM-OSA-compliant domain in the context diagram (see Figure 23)[1]. When a domain diagram is created, the images of the domain's related constructs (i.e. event and object view inputs and outputs of the domain) are automatically inherited by the domain diagram (i.e. bold OV's and EV's in Figure 25 which shows an example of a domain diagram)[2]. In a domain diagram, a user

1. Figure 22 depicts how each diagram is created by representing the relationship between object-operations executed on some key constructs and their associated diagrams by dashed lines.

has the options to add domain processes and physical and information object views and events[1]. Object views and events are processed and generated within the scope of the domain as inputs and outputs to domain processes.

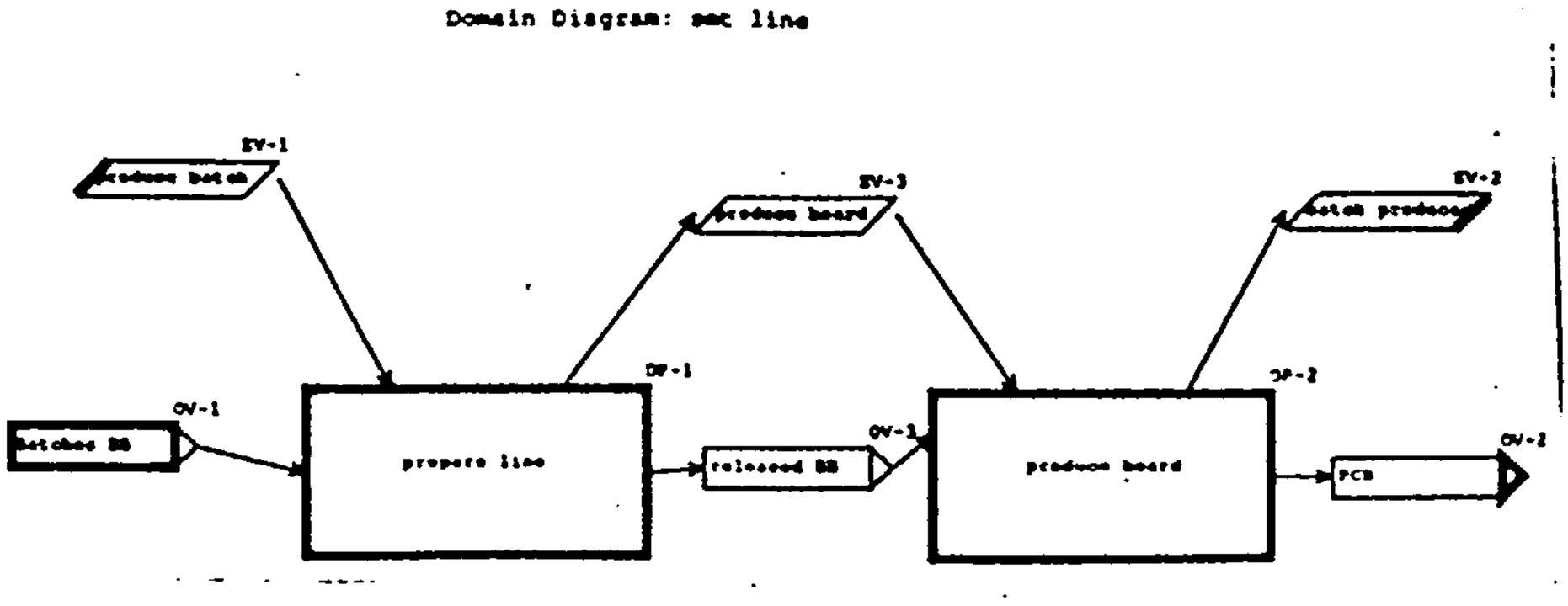


Figure 25 - Example of a domain diagram

Here, one should notice that a domain process is merely a functional representation of a class of tasks required to be executed within a certain domain. In Figure 25, for example, a domain process does not represent a physical assembly line or segment of line. It rather denotes types of functionality required to be performed by resources. The functionality of an occurrence of a domain process at run time can end up being provided by a number of physical assembly lines. Conversely, a number of occurrences of a domain process can be generated at run-time and compete for the resources made available by a limited number of assembly lines.

Similarly to the context diagram, once the domain diagram is complete the user may proceed to the definition of structure diagrams associated with each domain processes.

---

2. This domain diagram is a simplification of the domain diagram created from the context diagram shown in Figure 23. This domain diagram was extracted from a model of the surface mount technology (i.e. SMT) assembly line, whilst the context diagram was extracted from a model of the complete shop-floor. The remaining diagrams presented in this chapter relate to the model of the SMT assembly line.
1. Domain processes also require the definition of objectives, constraints and declarative rules (the latter aggregates objectives and constraints into logical equations which define when and how they are applied). SEW-OSA enables a domain process to inherit some or all objectives and constraints that have been defined for the domain that it belongs to. The objectives and constraints of a domain must be distributed amongst the domain processes that are contained within the domain, in such a way that objectives and constraints that are assigned to domain processes are defined in accordance with objectives and constraints assigned for the domain as a whole.

5.2.3. Structure diagram

A structure diagram represents the structural decomposition of the functionality of each domain process. A domain process consists basically of a hierarchy of business processes and enterprise activities. Business processes encapsulate behaviour whereas enterprise activities contain only functionality (at the level of granularity required by this modelling level).

The structure diagram inherits the name of its parent domain process. When this diagram is created, a graphical image of its parent domain process is automatically copied into the top of the hierarchy of functions represented by the structure diagram. Figure 26 shows an example of a structure diagram which enables the user to[1]:

- add business processes and enterprise activities;
- define hierarchical relationships between business processes, enterprise activities and the domain process. The relationships are of the type "used_by" and identify those functions that are required by the domain process or a business process to describe its functional content.

It is important to notice that in the hierarchical relationships described in this diagram, lower level functions can be used by and shared between more than one higher level function. Moreover, as it can be seen in Figure 26, a lower level function may use functions which are at the same hierarchical level or even at a higher hierarchical level than that of its own. The only structural relationship that this diagram does not allow is the representation of a function using itself directly or indirectly. Direct use (i.e. a function calling itself in a recursive manner) has no structural meaning (i.e. a function is always composed of itself!). However multiple executions of its own functionality (i.e. recursive triggering) is dealt with in the behaviour diagram, by means of an appropriate procedural rule (i.e. "loop"). Indirect use (i.e. a function using a higher level function which, in turn, uses the function in question) is neither addressed by SEW-OSA nor supported by CIM-OSA. The author envisages that such a situation, despite of being relatively rare, could be handled by SEW-OSA if appropriate ending-status for the functions are used as a means of "breaking the loop".

From the structure diagram the user may proceed to define either the behavioural or the functional content of each domain process and its subordinate

---

1. This diagram also enables the user to define which objectives, constraints and declarative rules of a certain process (i.e. domain process or business process) are inherited by the functions that it uses (i.e. business process or enterprise activity). Therefore, in addition to defining hierarchical relationships in terms of functional content, this diagram also represents the structure based on which metrics can be associated with functions.

Structure Diagram: produce board

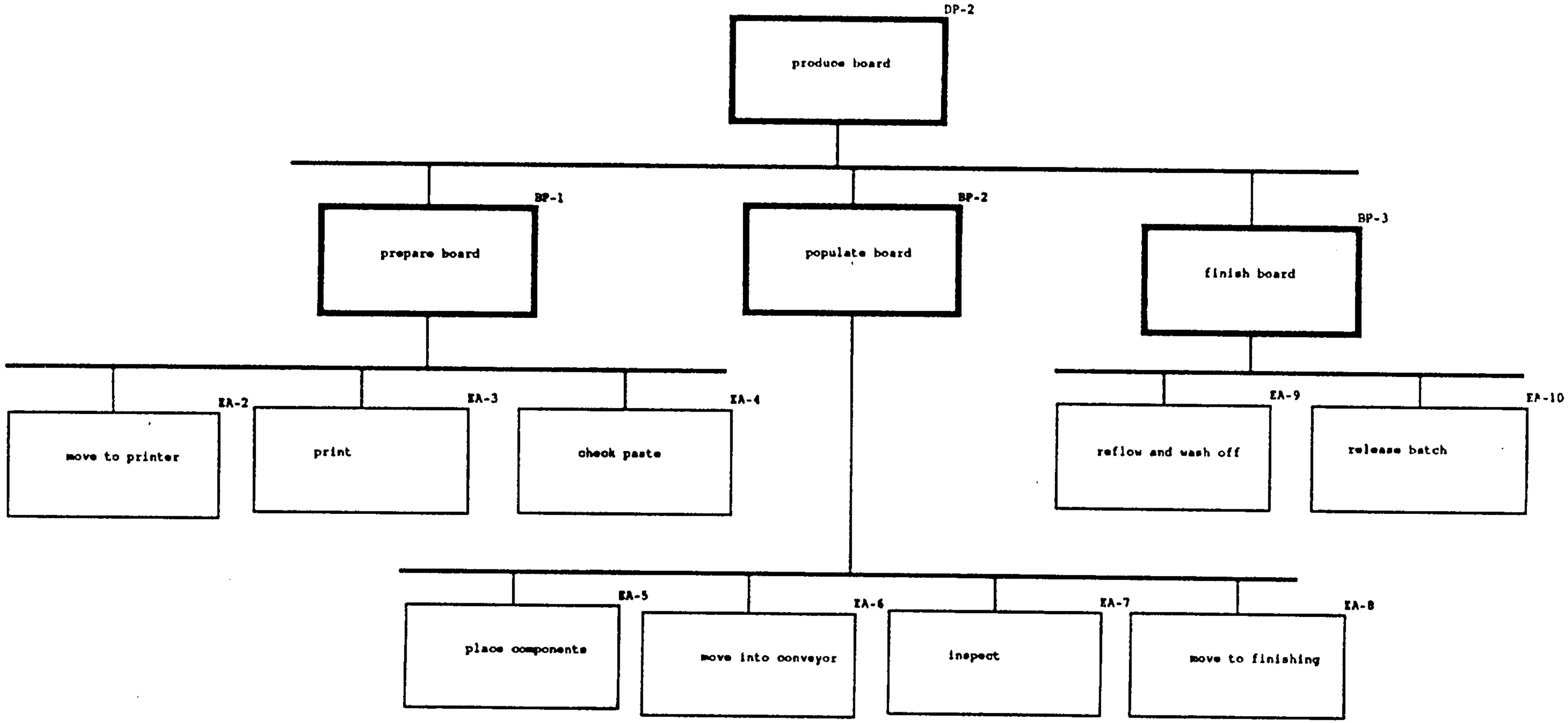


Figure 26 - Example of a structure diagram

functions. Although no sequence of definitions is imposed by SEW-OSA, the author recommends the definition of the behaviour description before the functional description for, whilst developing the former, the user is more likely (than whilst developing the latter) to change the structure of the domain process.

5.2.4. Behaviour diagrams

Behaviour diagrams are hierarchical sets of diagrams which describe the behavioural content of each domain process. The hierarchy followed by this set of diagrams is based on the inherent hierarchy defined in the structure diagram (see Figure 26). When the top-level behaviour diagram is created, all event inputs associated with the domain process are copied into the diagram. Furthermore, the functions "used-by" the domain process, as defined in the structure diagram, are inherited by the appropriate behaviour diagram. Figure 27 shows a behaviour diagram associated with the structure diagram of Figure 26. In this diagram, the options available to the user are to[1]:

- add business processes and enterprise activities that are used by the function, as defined in the structure diagram. This means that additional functions that have not been previously foreseen in the structure diagram can be added;
- link business processes, enterprise activities and events by means of procedural rules;
- change the "ending status" of functions with respect to their procedural rules.

The procedural rules implemented in SEW-OSA are presented in Table 2. This table also defines how different types of procedural rules apply to the constructs of the behaviour diagram (i.e. a condition which is not defined in CIM-OSA). Additionally, when implementing these rules, it was identified that the rule named "conditional" prescribed by CIM-OSA was redundant. Hence, this rule was excluded from behaviour diagrams.

The options available to the user in 'child' diagrams are similar to those at the top-level behaviour diagram, except for the inexistence of events (used to mark the start of the execution of the behavioural description of a diagram). Hence, threads of behavioural execution described by behaviour diagrams belonging to business processes are triggered by events internal to the domain process (i.e. the event 'Start',

---

1. From a top-level behaviour diagram, 'child' diagrams can be created from object-operations on business processes. Recursively, 'grand-child' diagrams can be created from 'child ones', up to the point where every domain process and business process has a child diagram associated with it.

Behaviour Diagram: populate board

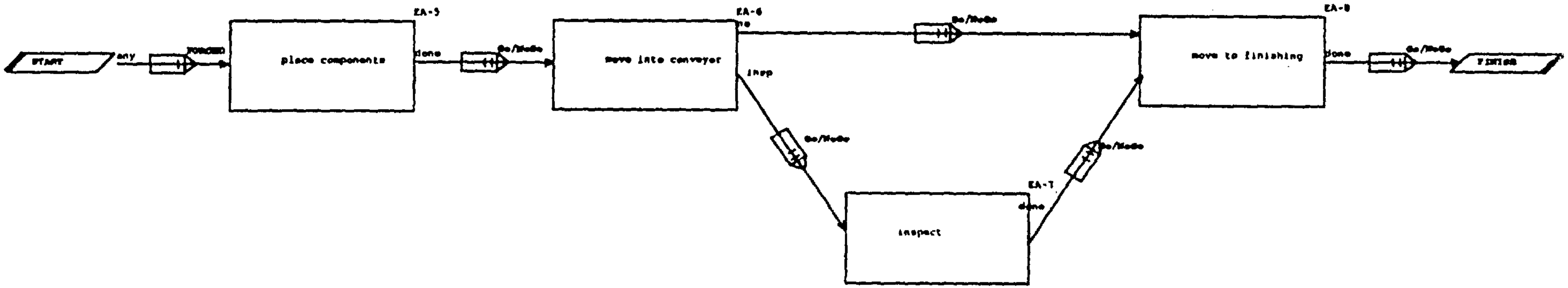


Figure 27 - Example of a behaviour diagram

as indicated in Table 2). Both types of diagrams (i.e. those belonging to domain processes and those belonging to business processes) are terminated by internal events, namely: FINISH (normal conclusion) and TERMINATE (abnormal conclusion), as shown in Table 2. In the current implementation of SEW-OSA, abnormal conclusions are not used. This, in principle, does not affect the description power of the model-building capability.

Table 2 - Procedural rules implemented in SEW-OSA

Procedural Rule	Permitted Inputs from	Permitted Outputs to
FORCED (FRC)	event enterprise function Internal event "Start"	event enterprise function internal event "Finish" internal event "Terminate"
GO/NOGO (GNG)	event enterprise function	event enterprise function internal event ("Finish") internal event ("Terminate")
SPAWN (SP)	event enterprise function internal event ("Start")	event enterprise function
RENDEZVOUS (RE)	event enterprise function	event enterprise function internal event ("Finish") internal event ("Terminate")
LOOP (LP)	enterprise function	enterprise function

External events that are generated by a domain process (as illustrated in Figure 25) are defined in the functional diagram. In SEW-OSA, these events are only generated by enterprise activities.

Figure 28 shows an example of a business process template, with its associated behavioural description in the form of procedural rules obtained from its behaviour diagram (see the bottom part of the figure).

Once all behaviour diagrams are complete, the user may proceed to define the functional diagram.

5.2.5. Functional diagram

A functional diagram represents the functional content of a domain process. This diagram describes the flow of object views and events through the enterprise activities of a domain process, as well as resource requirements associated with each enterprise activity. This is done by formalising the three types of inputs and outputs of each enterprise activity contained within a domain process. The three types of inputs are: main inputs (i.e. information and physical object views - to be transformed), secondary inputs (i.e. controls, statuses, and information and physical object views -

BUSINESS PROCESS TEMPLATE

TYPE:	<select from list>
IDENTIFIER:	BP-2
NAME:	Populate board
DESIGN AUTHORITY:	<authorised person>
DESCRIPTION:	<short textual description>
OBJECTIVE:	
CONSTRAINT:	
DECLARATIVE RULE:	
COMPRISES:	place components move into conveyor inspect move to finishing

BEHAVIOUR:

ON (START) DO place components
ON (ES(place components) = done) DO move into conveyor
ON (ES(move into conveyor) = no) DO move to finishing
ON (ES(move into conveyor) = insp) DO inspect
ON (ES(inspect) = done) DO move to finishing
ON (ES(move to finishing) = done) DO FINISH

Figure 28 - Example of a business process template

used in the transformation) and tertiary inputs (i.e. the resource capability - requirements upon the objects required to perform the transformation). Hence, enterprise activities resemble functional boxes in IDEF0 [Bravoco 1985a]. However, in a manner which is dissimilar to that in IDEF0, the functional content of a domain process is represented in just one diagram. This is due to the fact that in a domain process only enterprise activities actually possess functionality. An alternative solution[1] would be to create a hierarchy of functional diagrams, defined according to the structure used in the behaviour diagrams, whereby business processes would posses the three types of inputs and outputs, similarly to enterprise activities. Although this may improve the readability of the models produced, it may convey the misleading idea

1. This issue proved to be controversial up to the latest meetings of the CIM-OSA consortium [ESPRIT/AMICE 1993a]. See Figure 15 for the CIM-OSA representation adopted.

that business processes actually process objects when in fact they do not.

Likewise in the behaviour diagram, the functional diagram inherits the inputs and outputs (defined in the domain diagram, as depicted Figure 25) and enterprise activities of its parent domain process (defined in the structure diagram shown in Figure 26). Figure 29 shows an example of this type of diagram, created by:

- adding enterprise activities that are used by the domain process (as defined in the structure diagram);
- adding object views internal to the domain process;
- defining the resource capability required by each enterprise activity; and
- inter-connecting enterprise activities by means of a flow of internal object views and external object views and events. External object views and events are the interface between the domain process and its external environment.

5.2.6. Changes and adjustments in the RDML

Completion of the functional diagram marks the end of requirements definition modelling (see Figure 22). It is important to notice that the complete business model (at this modelling level) describes a few domains and a number of domain processes[1]. To each domain process is associated one structure diagram, one functional diagram and a set of as many behaviour diagrams as there are business processes in each domain process. Therefore, it is quite important to enable the user to navigate between diagrams and change them in such a way that the model remains consistent. To an extent, SEW-OSA allows these changes to be made at any stage in the modelling process. For instance, new constructs can be added in any part of the model and appropriate updates will be propagated to the remainder of the model. Some of the features that enable consistent change of models are:

- **the ability to navigate directly from a certain stage in the model to any other stage**, as long as there is a logical path between them. For instance, from the structure diagram to a particular behaviour diagram belonging to a certain business process (represented in the structure diagram); from a behaviour diagram to any of its parents[2] as well as to children of business processes described in the diagram; from a bottom-level to a top-level behaviour diagram and vice-versa.

---

1. CIM-OSA-compliant domains do not share with other domains instantiations of their component constructs (i.e. their function, structure and behaviour). However, their component constructs may have been instantiated from the same partial models (i.e. reference models).
2. Multi-parenthood is accepted and encouraged in CIM-OSA.

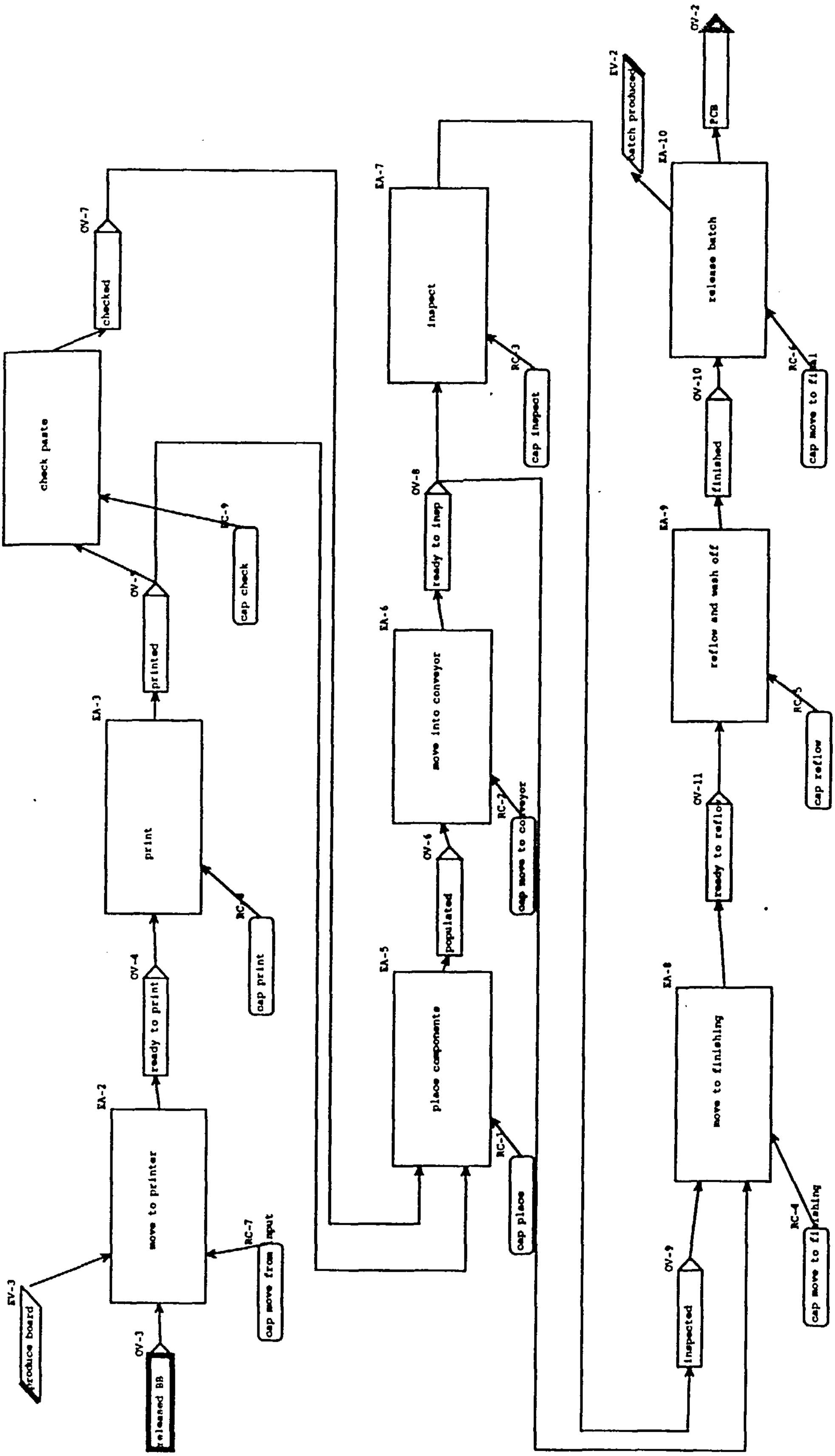


Figure 29 - Example of a functional diagram

- **propagation of changes from a diagram to the rest of the model.** When a construct (e.g. an object view) is added or deleted on a diagram, its corresponding images in lower level diagrams are respectively added or deleted. Similar propagations are also realised for attribute changes.
- ***unique identification of constructs.*** *SEW-OSA automatically creates and* maintains the attribute "IDENTIFIER" associated with each modelling constructs (e.g. EA-1 in Figure 29). *Such a facility also keeps track of all previously defined* identifiers in order to provide a unique identification for each construct instance. That is, every time a new construct instance (e.g. domain, event, object view, etc.) is created or deleted, the identifiers of all related constructs in the model are updated.
- **consistent naming scheme.** This includes a scheme by which diagram names always take the name of the entity from which they have been created (see Figure 22). In turn, the context diagram inherits the name of the model to which it relates.

Additionally, inherent features that the SEW-OSA CASE tool provides (which turned out to be quite important for documentation purposes) are:

- to check the existence, adequacy and consistency of logical links between constructs previous to the creation of such links;
- to distinguish between external constructs (i. e. those defined at a higher level in the model) from internal ones;
- to position links adequately in relation to the object which represents the construct graphically (e.g. primary inputs should always come into the box that represents an enterprise activity through its left hand side border; this convention being borrowed from the IDEF0 modelling language) [Bravoco 1985a].

The requirements definition model, completed in the manner described in the foregoing, is a description of **what** the system is expected to accomplish. However, **how** the system is going to accomplish its task, with repercussions to the definition of system configuration and its components, is not defined at this stage. For instance, examples depicted in Figures 23 to 29 describe the functions to be performed by the shop-floor. However, those descriptions provide no insight about how these functions will be performed in terms of the specification, number and capacity of the components that they may require and the manner by which they are organised.

5.3. Design Specification Modelling Level (DSML)

The design specification modelling level starts by working upon what was delivered from the requirements definition modelling level, as implemented by SEW-OSA, namely:

- a complete function view model;
- the definition of resource capabilities (to be used by the resource view);
- the identification of physical and information object views to be used by the information view.

The requirements definition modelling level is fairly well defined by CIM-OSA. Thus, apart from transforming those recommendations and specifications into an organised modelling method implemented in a CASE tool, the activities described in Section 5.2 primarily represent enhancements to existing recommendations of CIM-OSA. However, some of these enhancements involved the definition of certain issues left undefined in CIM-OSA and reviewing and validating issues whose need had not even been validated.

However, the design specification modelling level is still fairly open, this in regard to the specification of its design activities and the way these activities should be formally supported. Basically, the design specification modelling level is where requirements captured by the requirements definition model are transformed into system specifications (also captured by models).

The design specification model in SEW-OSA consists basically of a set of descriptions derived for each domain identified in the context diagram (see Figure 23), each of which comprises the following diagrams (represented in Figure 22), namely: an object diagram, a resource diagram, a configuration diagram, a set of activity behaviour diagrams, and a set of entity behaviour diagrams (described as follows).

5.3.1. Object diagram

An object diagram is used to define interactions between the integrating infrastructure and the integrating operation environment (i.e. a functional representation of all active resource components within a domain). Such interactions are described in terms of messages exchanged among three parties, namely: representations of active resource components in the form of functional entities, the enterprise activities defined in the functional diagram and the information entity (see Figure 30). All messages (i.e. functional operations) presented in Figure 30 represent how enterprise activities implement their transfer function defined in the functional diagram. Basically, each occurrence of an enterprise activity executes an algorithm (i.e.

the transfer function) which utilises fragments of functionality resident in functional entities and in the information entity.

It is important to notice that functional entities represent the external behaviour, as perceived by the integrating infrastructure, expected from the types of resources required by the system. Functional entities represent types of resource rather than individual instances of resources. Messages exchanged in the object diagram are associated either with functional operations to be executed by functional entities or data transactions with the information entity (see Figure 30).

The object diagram inherits the enterprise activities (defined in the functional diagram shown in Figure 29) and defines the two interacting environments of interest (i.e. the IIS and the IOE). The information presented in Figure 30 is formalised by using facilities to: (1) add functional entities; and (2) link enterprise activities, functional entities and the information entity by means of functional operations (i.e. messages that can be exchanged amongst these three elements, as depicted in Figure 30).

Having completed the object diagram, a designer needs to define the internal behaviour of enterprise activity objects and functional entities, whilst responding to the external interactions depicted on the object diagram. This is accomplished by defining a behaviour diagram (i.e. a form of state-transition diagram) for each enterprise activity and functional entity depicted in Figure 30.

5.3.2. Activity behaviour diagram

This diagram defines the internal behaviour of an enterprise activity object by means of predicate-action Petri-nets. Predicates and actions represent either functional operations (which describe the interactions with functional entities and the information entity) or internal conditions and actions. Internal conditions and actions represent activity processing which is not captured by the Petri-net formalism, although it must be considered for the sake of decision making within the enterprise activity. An activity behaviour diagram is created from an object-operation selected from an enterprise activity in an object diagram. The activity diagram pre-defines a start and an ending transition which represent, respectively, the triggering and the completion of the enterprise activity within the context of its procedural rule execution (see Figure 27). Both start and ending transitions are "fired"[1] only once at run time, namely: when the enterprise activity is created and when it is completed. An example of such a diagram is shown in Figure 31, whereby the following options were available for the designer in order to create the diagram, namely:

---

1. The reader should refer to Appendix 4 for a explanation about Petri-nets and their associated terminology.

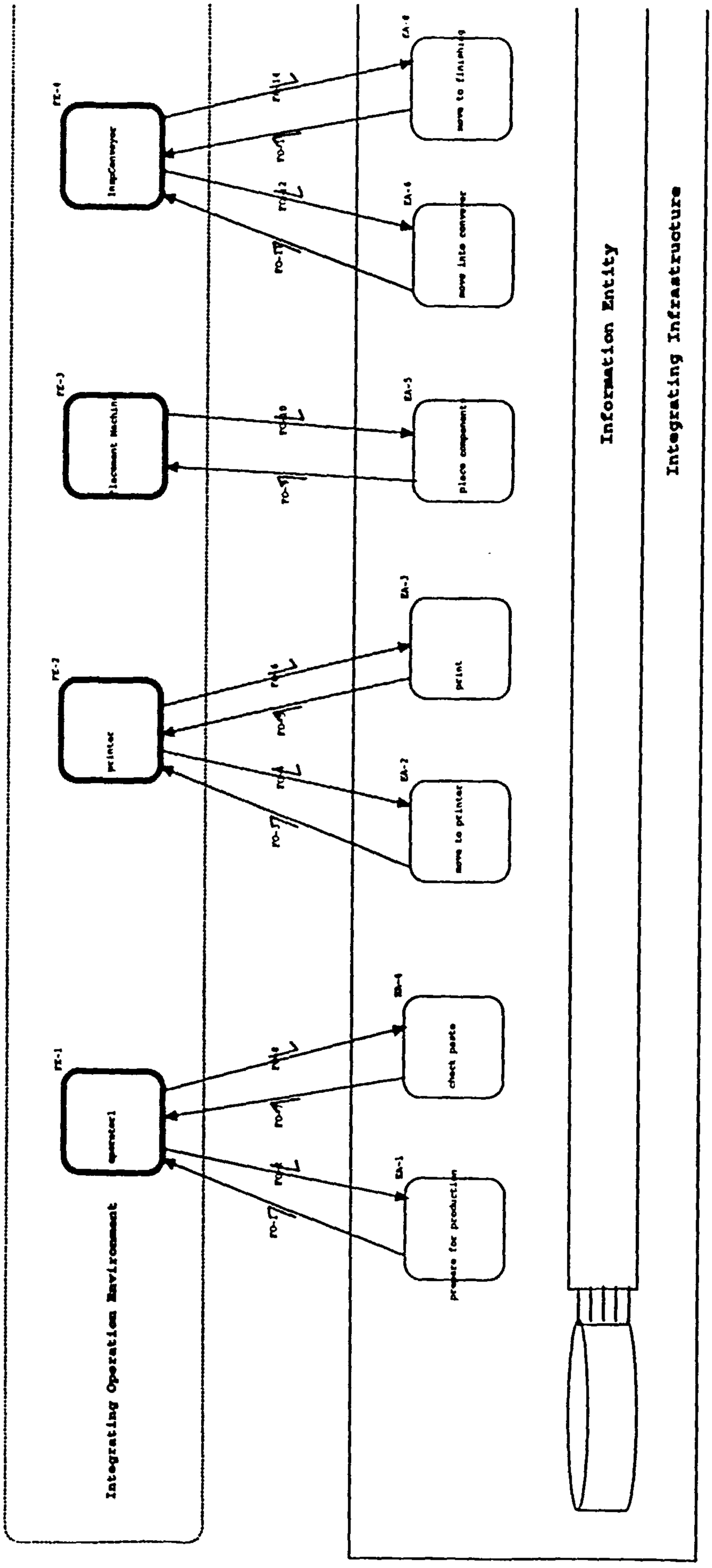


Figure 30 - Example of an object diagram

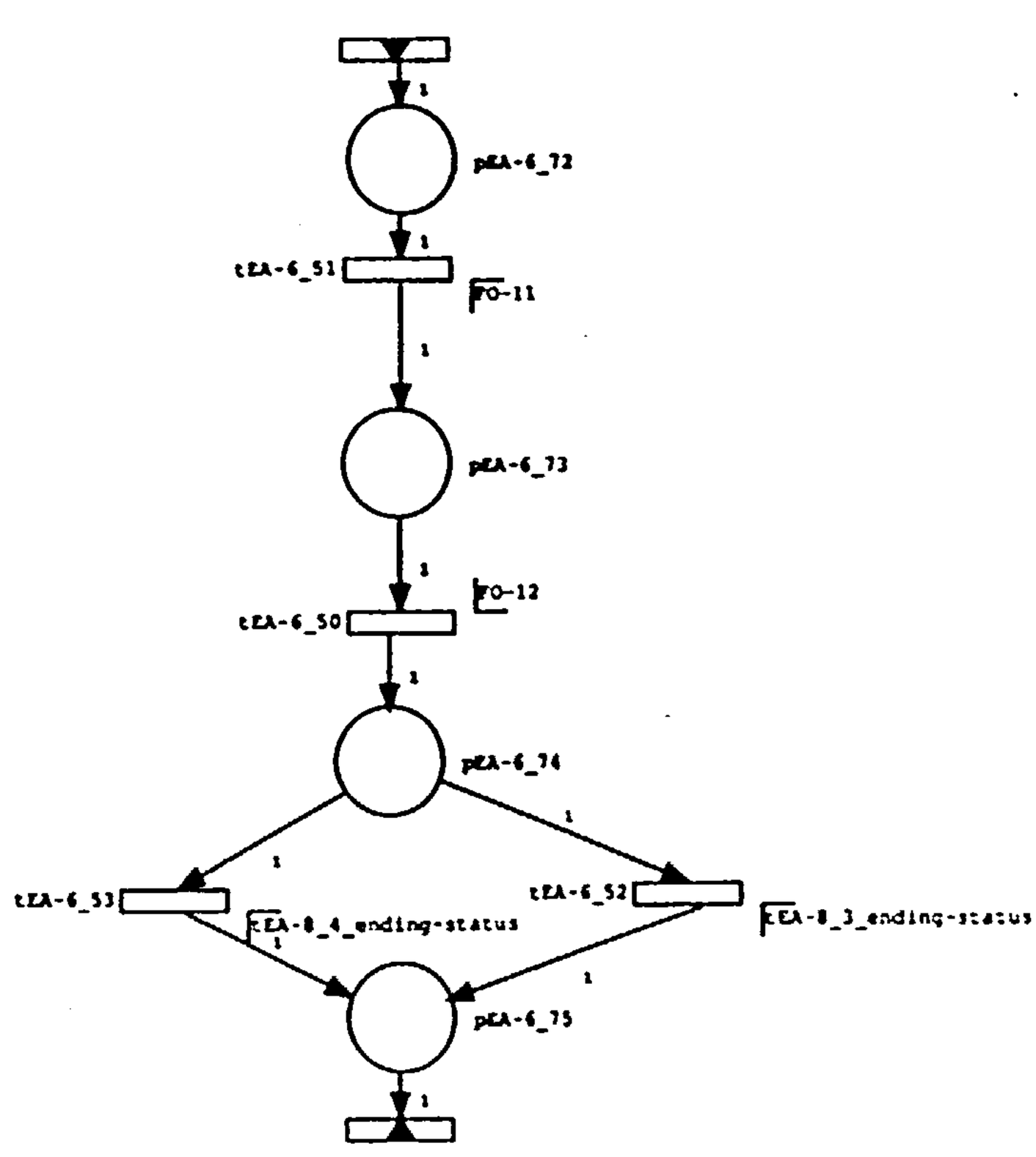


Figure 31 - Example of an activity behaviour diagram

- to add transitions and places;
- to link places to transitions (and vice-versa) by means of arcs;
- to assign predicates and actions to transitions;
- to assign tokens to places;
- to assign weights to arcs (the default weight value is one);
- to define the content of internal conditions and actions;
- to change the ending status of the enterprise activity (the default status is "done").

Ending statuses are the means by which the internal processing of enterprise activities are linked to the decision flow captured by process behaviour diagrams (exemplified in Figure 27).

5.3.3. Entity behaviour diagram

This diagram is used to emulate the behaviour of active resource components during the rapid-prototyping of a system. As with the previous diagram, it defines the internal behaviour of a functional entity object by means of a predicate-action Petri-net. This diagram is created from an object-operation selected on a functional entity in an object diagram (see Figure 30). The entity behaviour diagram pre-defines only the (initial) idle state of the functional entity. An example of such a diagram is shown in Figure 32, which offers the designer the same options available in the activity diagram except for the option related to the ending-status of an enterprise activity, which has no meaning for a functional entity.

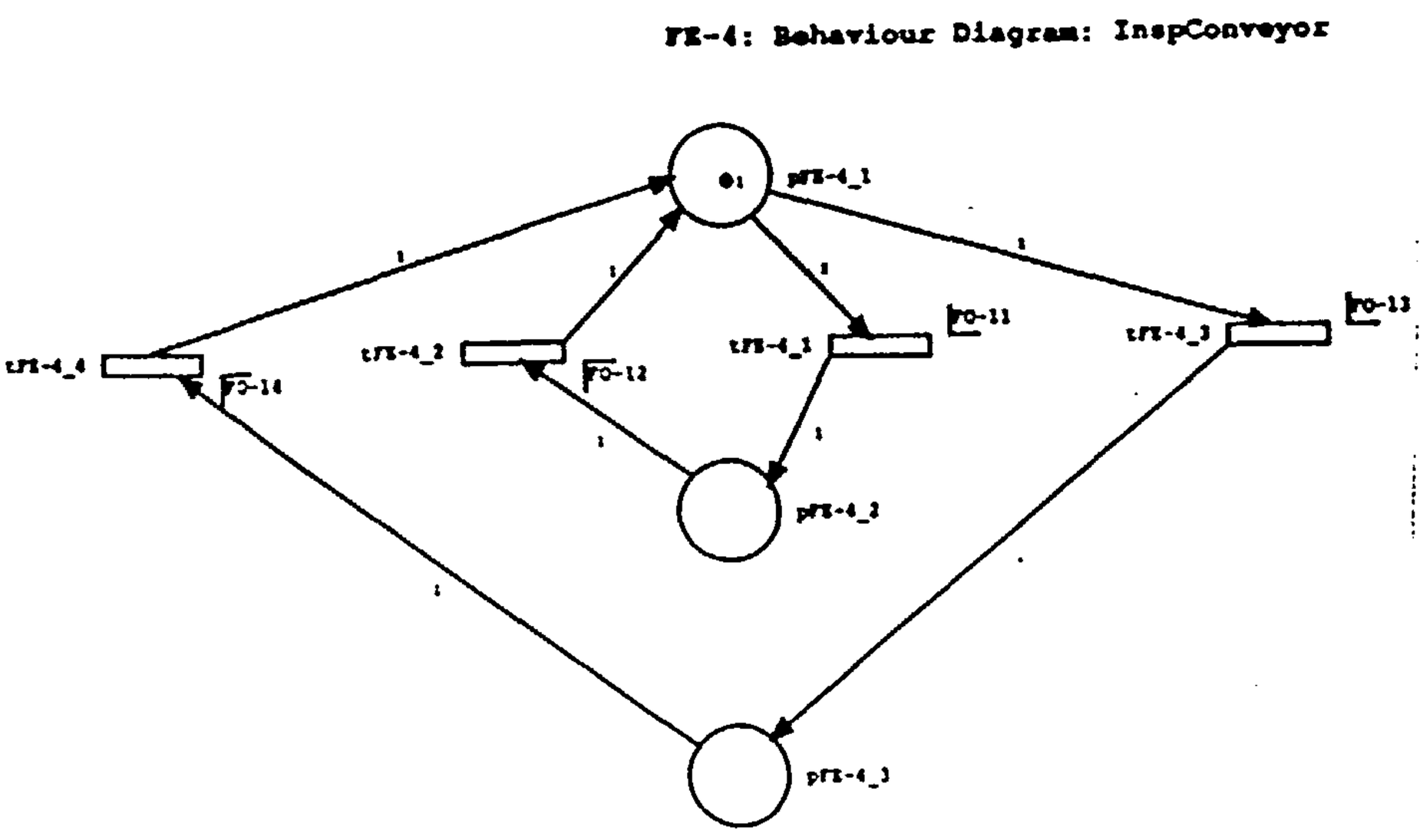


Figure 32 - Example of an entity behaviour diagram

Once behaviour diagrams for every activity and entity in the object diagram are complete, the designer is at a stage in which the functional decomposition of a model is defined to the finest level of granularity (i.e. functional operations). However, up to this stage, the business model does not define either what physical resources will be required to provide such a functional composition nor how they are organised.

5.3.4. Resource diagram

A resource diagram defines the mapping between the functionality of the system and the resources which should provide that functionality. This is accomplished by providing means of relating the resource capabilities defined in the functional diagram (see Figure 29) with the external behaviour of functional entities defined in the object diagram (see Figure 30), in order to identify possible candidates to active resource components. These candidates must both fulfil the capabilities defined at the requirements definition modelling level and provide means of interaction which are

compatible to those defined for its associated functional entity. The identification of candidates to physical resource components should also be a model-driven process, based on models of resources (the manner in which this process has been implemented is explained later in the thesis). Additionally, the resource diagram also defines how resources are organised for the sake of scheduling.

The resource diagram inherits the resource capabilities defined in the functional diagram for each enterprise activity belonging to the domain under consideration (see Figure 29) and the functional entities defined in the object diagram (see Figure 30). Figure 33 shows an examples of this type of diagram, created by using facilities to:

- add active and passive resource components associated with functional entities;
- define which resource components (active and passive) fulfil given resource capabilities; and
- to group resource components into resource cells and resource sets. Here, a practical definition of resource cells and sets was devised. A resource set defines a grouping of resources which contain physical inter-dependencies between them (e.g. an assembly line is a resource set, for when a job is assigned to the first machine on the line, it is implied that the job will have to be processed by the remaining downstream resources of that line). A resource cell defines a pool of resources which can be interchangeably used to process the same types of jobs. It is important to note that resource cells can also group resource sets and vice-versa. This enables complex inter-relationships among resources to be captured.

An important message emerging from the diagram in Figure 33 is that of adopting a late-binding philosophy (as opposed to an early-binding one) for the allocation of resources to functions. This means that the actual allocation of a certain active resource component to perform a functional operation triggered by an enterprise activity is to occur only at run-time. The active resource component will be allocated from the pool of resources from its pertaining functional entity (represented as dotted boxes in Figure 33).

The completion of the resource diagram means that each and every function in the business model has a candidate resource to perform it. However, as these resources will be integrated by means of a computer infrastructure, a configuration for such an infrastructure must be defined (e.g. the addresses of resources on a networking environment). This is the type of definition supported by the configuration diagram.

Resource Diagram: ant line

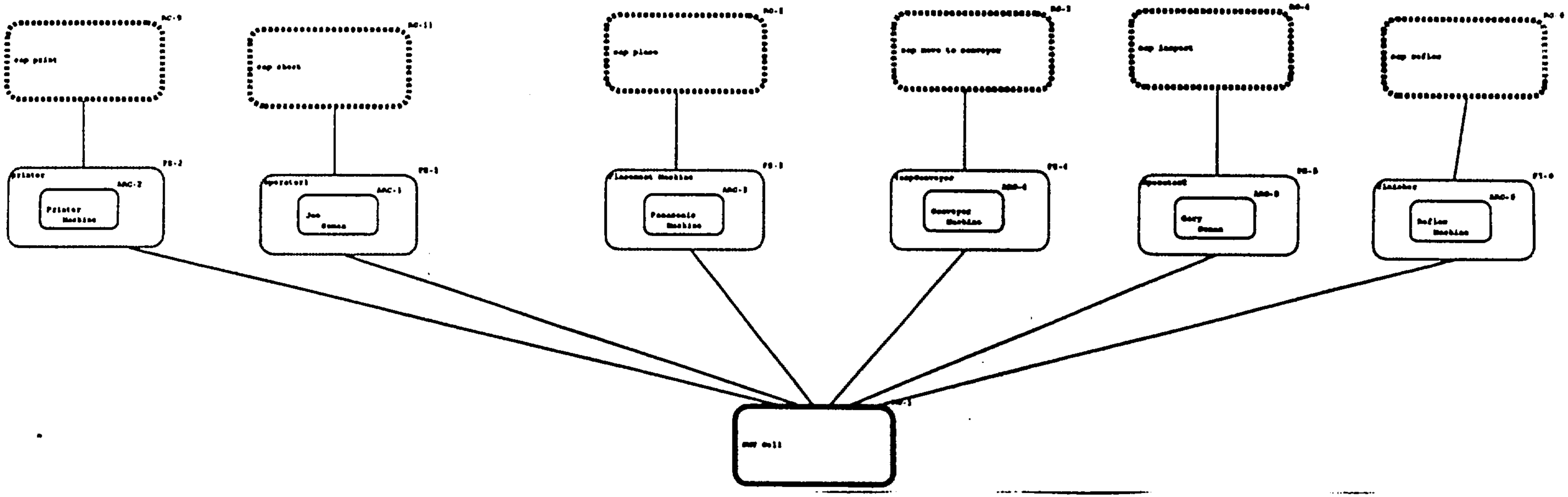


Figure 33 - Example of a resource diagram

5.3.5. Configuration diagram

The configuration diagram (1) inherits the definitions of the resource diagram, in terms of specified active resource component types (i.e. human, machine or application software - see Figure 33), (2) associates an appropriate icon to each type, and (3) defines an initial topological configuration for the computer environment (see Figure 34). This initial configuration consists of assigning the business entity components to the host in which the SEW-OSA CASE tool is executing. Figure 34 shows an example of this type of diagram which provides the following facilities to a designer, namely:

- to add hosts (i.e. computer stations on which a process representing an active resource component or a business entity component can be executed);
- to assign processes to hosts;
- to change the initial configuration (defined by the CASE tool) in order to re-distribute the business entity across the system.

Completion of the configuration diagram marks the end of the modelling process. A design specification model so completed, can then be tested by enacting it via the model-enactment capabilities of SEW-OSA.

5.3.6. Structure of the modelling process

In broad terms, the functionality provided by the model-building capability enables the derivation of a system specification (i.e. an object-oriented description) based on a process-oriented description of what the system is expected to achieve. The constructs manipulated between these two descriptions are presented in Figure 35. This diagram also represents the information refinement process involved in creating a business model. This process extends from the identification of domains to the definition of their content in terms of domain processes, business processes, enterprise activities and functional operations.

As shown in Figure 35, functional entities and active resource components, represented by models, are selected based on the capability required by the enterprise activities they support, as discussed later in the thesis.

5.4. Realisation of the Model-Building Capability

Figures 36 and 37 present the backbone structure of the SEW-OSA CASE tool. As outlined in Figure 18, the CASE Tool Structure consists of an entity-relationship representation which is used to implement the method in the meta-CASE tool. The model shown in Figures 36 and 37 (which is a simplified version of the CASE Tool

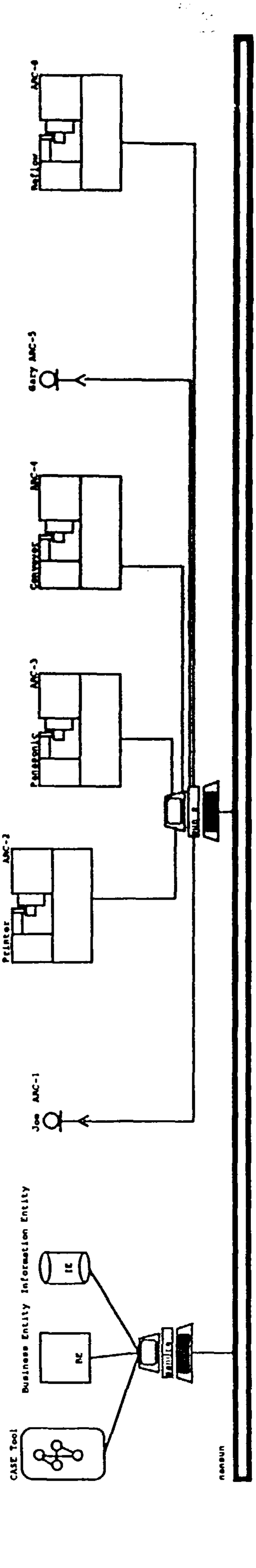


Figure 34 - Example of a configuration diagram

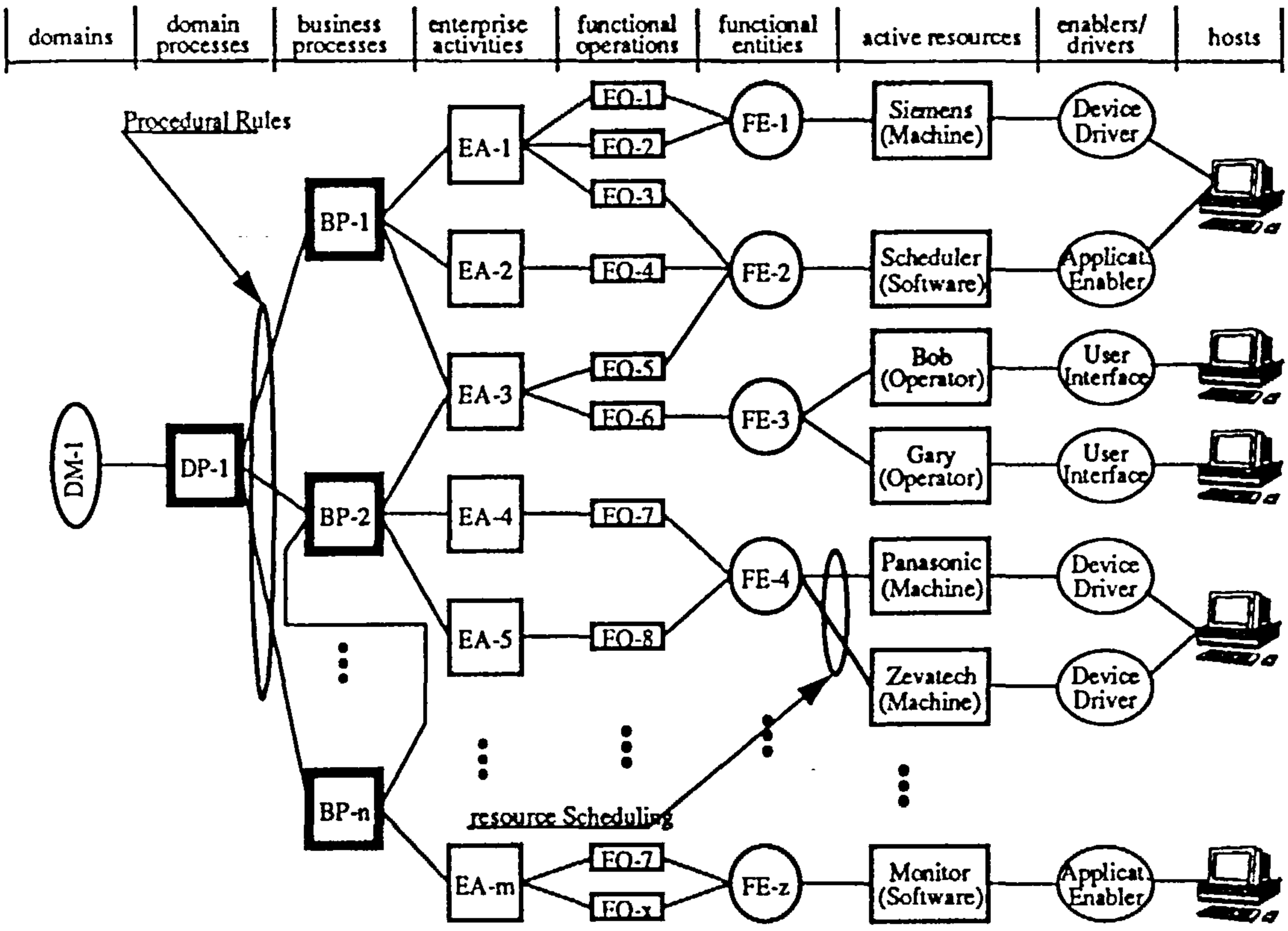


Figure 35 - Structural definition provided by SEW-OSA

Structure) is a representation of two of the dimensions of the CIM-OSA cube, implemented in the SEW-OSA CASE tool, namely: instantiation and derivation (see Figure 12). Along the instantiation dimension, the CASE tool provides:

a. **An overall description of the basic building blocks of CIM-OSA and how they relate to one another** (i.e. the generic level in Figure 12).

   The SEW-OSA CASE tool was built in such a way that the nature of the CIM-OSA building blocks is embedded in the constructs handled during the design of a particular enterprise case. That is, the meaning of each construct is inherited by its images manipulated in each diagram of the model-building capability.

b. **A structure for the separation of libraries of system models and resource models** (as part of the reference models shown in Figure 7), inter-related via connectance models (discussed later in the thesis).

In regard to system models, whenever the designer needs to create a new design, a library of reference models of previously created designs should be available for him to browse and retrieve (i.e. copy) instances of models[1]. These instances should then be placed (i.e. pasted) into an appropriate area of a particular design (thus, embracing one or more diagrams in Figure 22) and adapted to adjust to the

particularities of the design.

In respect to resource models, SEW-OSA is envisaged to interface with a resource modelling tool with which it would share a resource model. This would enable the specification of possible solutions candidate to address an identified capability required by an enterprise activity. Such an interface is also discussed later in the thesis.

c. **Means of creating a complete particular model** which basically implements the methodology described in Section 5.3.6.

   Along the derivation dimension (at the particular level), the method described in Section 5.3.6 guides the design process by means of navigations along the entity-relationship diagram, as depicted in Figures 36 and 37.

In each diagram, the constructs mentioned in Section 5.3.6 are formalised by defining their semantic and syntactic functions (i.e. by assigning an entity in the model to a construct defining its attributes and fragments of code associated with operations executed upon them).

The formalisation of constructs, diagrams and their inter-relationships in an organised method consisted of the main effort of realisation of the SEW-OSA CASE tool. This involved the implementation of the CASE Tool Structure based summarised by Figures 36 and 37. These figures represent only a simplification of the constructs that constitute the backbone of the SEW-OSA CASE tool, for they possess the main relationships required for navigations across the CASE tool constructs. However, such a structure by no means reflects the complexity involved in the implementation[1] of the CASE tool structure. In this respect, Chapter 11 presents a synopsis of the size and complexity of the CASE tool.

5.5. Limitations

Two major factors have limited the process of understanding, reviewing and implementing the specifications of CIM-OSA:

i. The fact that the author was not participating directly in the activities of the ESPRIT/AMICE consortium which partially overlapped with the activities of this research. This factor augmented considerably the effort required to study the specifications which were being made available to the author, initially by D2D

---

1. As indicated previously, SEW-OSA provides the structure for such a process to occur, but it does not incorporate any reference models as yet.

1. The complete CASE Tool structure is presented in an internal report [Aguiar 1994i].

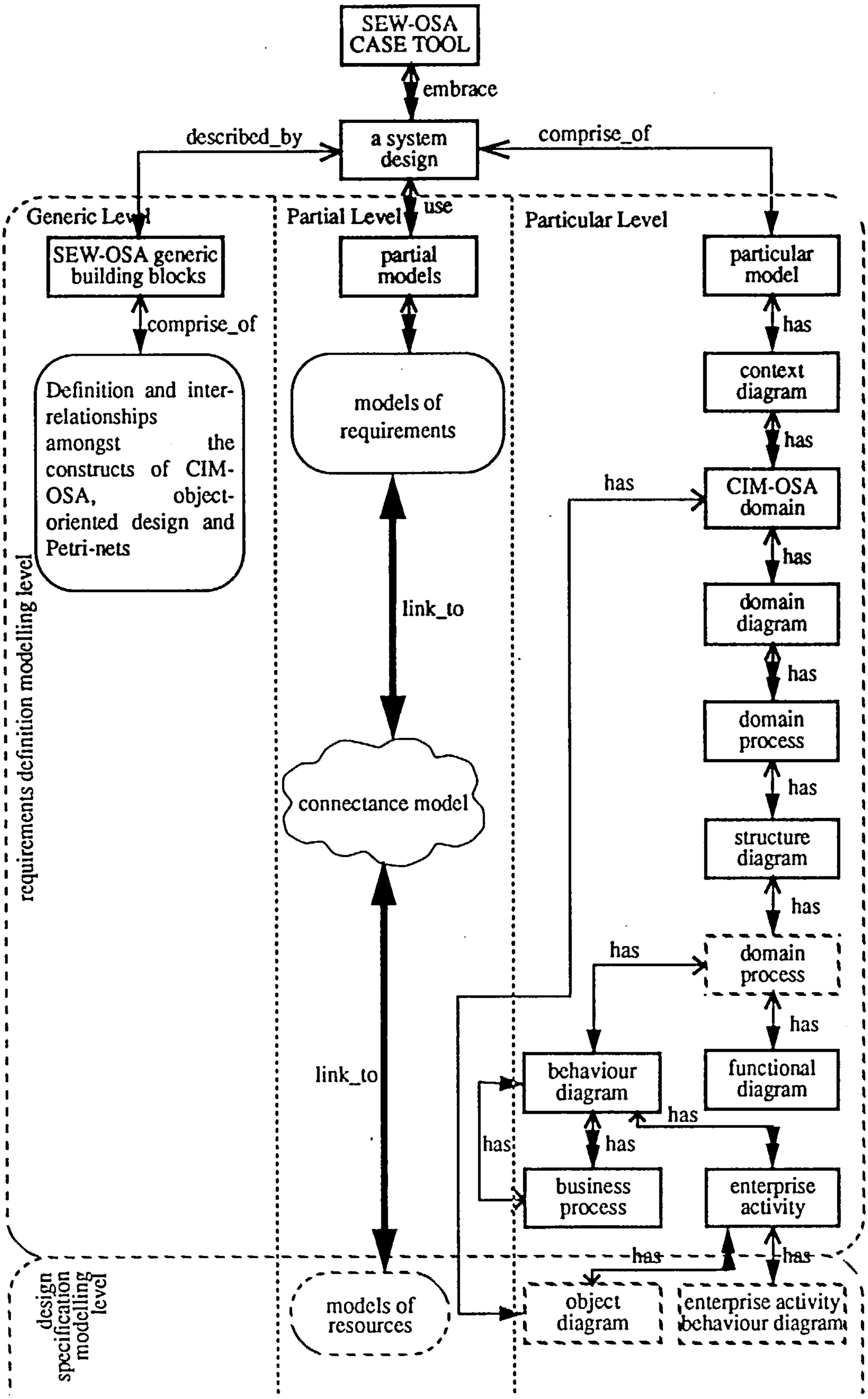


Figure 36 - Simplified structure of the CASE tool at requirements definition

(which was participating in the ESPRIT/AMICE consortium during the first year of this research) and subsequently by a research institute at the University of Karlsruhe, Germany. The specifications were extracted at first from publicly available documents [Beeckman 1989] [Clark 1989] [ESPRIT/AMICE 1987a] [ESPRIT/AMICE 1987b] [ESPRIT/AMICE 1989] [ESPRIT/AMICE 1991a] [ESPRIT/AMICE 1991b] [ESPRIT/AMICE 1991c] [ESPRIT/AMICE 1991d] [ESPRIT/AMICE 1991e] and then from the formal reference base [ESPRIT/AMICE 1992a] [ESPRIT/AMICE 1992b] [ESPRIT/AMICE 1993a]. The FRB on which this implementation is based was released in January 1993 [ESPRIT/AMICE 1993a] and constitutes the "frozen" version adopted by this research. The use of CIM-OSA in the context of SEW-OSA should be viewed as the author's interpretation of content of FRB's and other documents about CIM-OSA. Few opportunities occurred for an in-depth discussion of such an interpretation with members of the ESPRIT/AMICE consortium knowledgeable about CIM-OSA.

ii. The changing nature of the CIM-OSA specifications, where at times there were considerable discrepancies and contradictions between documents separately published by different members of the consortium.

iii. A lack of publicly available software tools[1] from which to extract guidelines for the implementation of CIM-OSA, as interpreted by the AMICE consortium.

These factors coupled with limitations of time and resources contributed, in turn, to certain limitations in respect to the functionality embodied within the SEW-OSA CASE tool. These limitations include:

a. Compliance with CIM-OSA:

A major limitation with respect to compliance with CIM-OSA, as clearly indicated in Figure 21, is that no part of the information and organisation views has been implemented. Although function and resource views are at the centre of the definition of a system structure, the four views are not independent. Analysis and manipulation of constructs of information and organisation views can help create a more adequate function and resource views' model (due to the introduction of additional checkpoints in the design process).

Some minor limitations are listed as follows:

- A structure has been implemented in SEW-OSA for the definition of objectives,

---

1. An exception here was the Demonstration tool [Emond 1988], which was produced in 1988. However its functionality was limited to support the behaviour and functional diagrams.

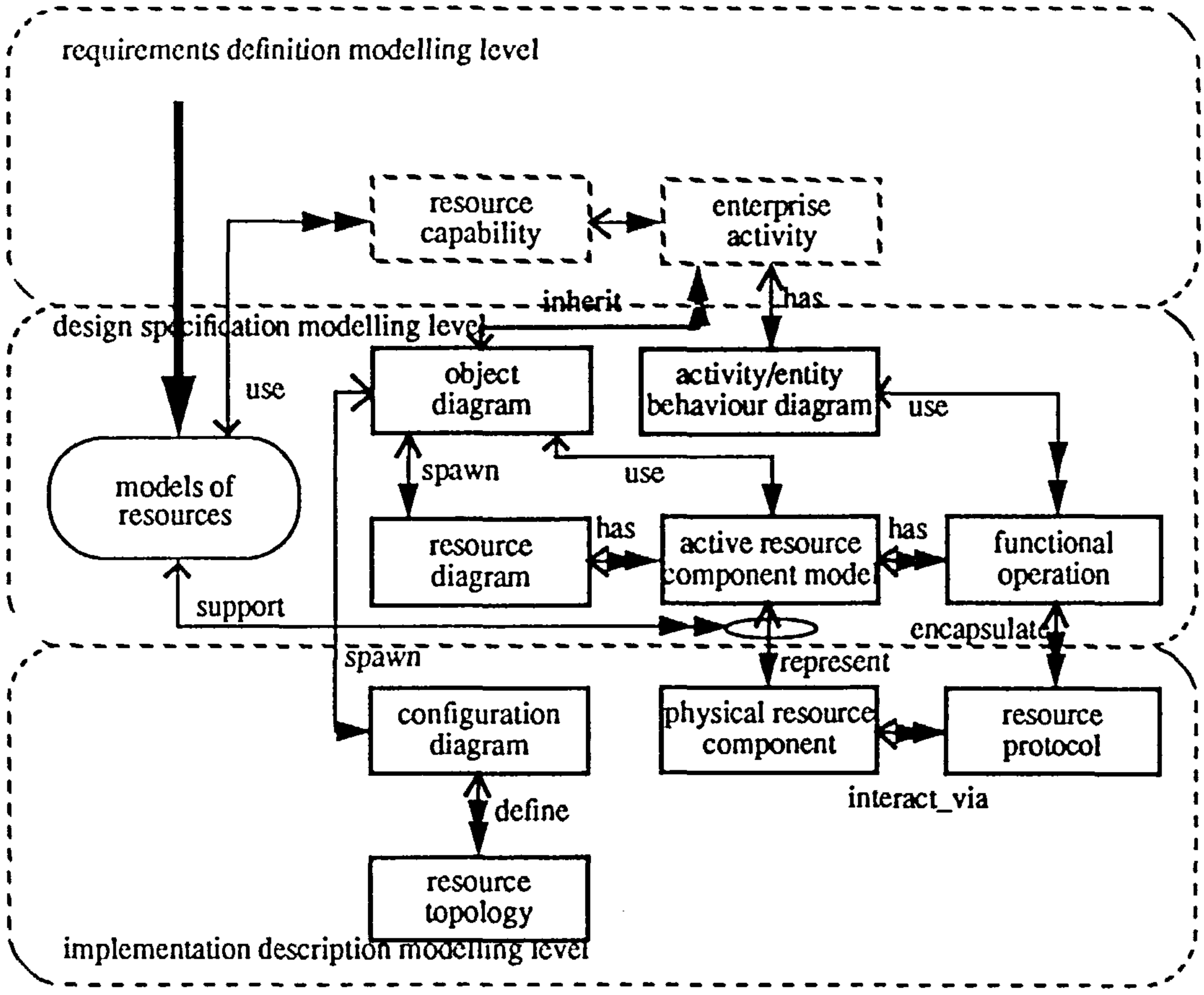


Figure 37 - Simplified structure of the CASE tool at design and implementation

constraints and declarative rules, as well as for their inheritance across the *functional decomposition from the level of a CIM-OSA-compliant domain down to* enterprise activities. However, declarative rules do not play an active role in the execution of the functionality of enterprise activities at run-time (i.e. declarative rules are not enacted). These constructs are used in SEW-OSA for documentation purposes only.

- CIM-OSA advocates that business processes and enterprise activities should be re-used through a model. At the current stage, instantiated business processes and enterprise activities can only be shared among the functions defined within the context of a domain process.

- It is envisaged that the ending-status of a business process be generated through the logical combination of ending-statuses of the functions it uses. The SEW-OSA CASE tool does not support the definition of such a logical combination.

- Support for using partially instantiated system models is limited in SEW-OSA to the reuse of complete models from previous designs. Facilities for reuse of

constructs or fragments of system models have not been included in the SEW-OSA CASE tool.

b. Features of the SEW-OSA CASE tool:

Although an effort was made in this research to realise the SEW-OSA CASE tool as a usable product, a considerable product engineering effort is still required if the tool is to be commercially exploited, namely:

- **Transformation of constructs.** This consists, for example, of translating a domain process into a business process and re-using it without losing all the definitions that have already been included in it when the construct was a domain process.
- **Improved consistency check.** The consistency checking function implemented in the SEW-OSA CASE tool is limited to guarantee the uniqueness of identification of every construct in the business model.

5.6. Contributions

The main contribution of the SEW-OSA model-building capability is the realisation of a modelling method which associates business process and object-oriented descriptions in order to produce a complete business model. Such a method captures the essential definitions of the CIM-OSA modelling constructs, organises them into a usable form and enhances them with additional constructs from object-oriented formalisms and Petri-nets. In order to achieve such a method, two major tasks have been accomplished, namely: (1) to **review CIM-OSA** (by analysing, correcting and complementing those parts that were not well defined) particularly in regard to design specification and implementation description modelling levels and (2) to **enhance CIM-OSA** with modelling constructs from the other architectures. As a result of these tasks, a number of incremental contributions (in their own right) have been made. Some of these contributions are exemplified as follows.

i. **Review of CIM-OSA:**

- the definition of a diagrammatic representation for each CIM-OSA construct, whilst CIM-OSA is limited to defining text templates[1];
- the definition of an appropriate meaning for terms introduced by CIM-OSA (such as functional entity);

---

1. Standardisation of the graphical representations of the CIM-OSA constructs is in discussion at the European standardisation level [CEN 1994a]. When such a standard becomes available, the flexibility provided by the "Graphical Representation Catalogue" of the meta-CASE tool (see Figure 18) allows the SEW-OSA CASE tool to easily comply with it.

- the definition of the actual effect of concepts used in the modelling process such as resource sets and cells, events and enterprise activities at run-time, to name just a few;
- the proposition of an implementation for the mechanism of inheritance of objectives, constraints and declarative rules, associated with processes and functions in the function view.

ii. **Enhancement of CIM-OSA**, associated with the realisation of:

- the object diagram (see Figure 30), as a turning point in the modelling process, linking descriptions at the requirements definition modelling level, which are inherently process-oriented, with intrinsic object-oriented descriptions of the remaining modelling levels;
- the resource diagram (see Figure 33), as a tool for: (1) selecting alternative resource components to address certain functional capabilities and, hence, coupling the functional content of a system with the functionality to be provided by its components; and (2) defining how components are to be allocated to function at run-time, by means of a late-binding strategy (discussed later in the thesis);
- the configuration diagram (see Figure 34), as a means of rapidly configuring CIM-BIOSYS. Before the introduction of this definition, the configuration of CIM-BIOSYS was a tedious process (as demonstrated by Gilders [Gilders 1991a] [Gilders 1991b]) that required the generation of a number of files containing fragments of information. This information can now be generated automatically by the CASE tool;
- the activity behaviour diagram (see Figure 31), which provides a graphical description of decisions within enterprise activities. In CIM-OSA, it was envisaged that this would be described in a text form;
- the entity behaviour diagram (see Figure 32), as a means of describing the expected behaviour of a selected resource element. This notion was not considered in CIM-OSA.

Another important feature of the SEW-OSA CASE tool is its potential for change, as it was implemented with a meta-CASE tool. Here, new constructs can be added, relationships between constructs can be changed or attributes of existing constructs re-defined. Such changes can be made without writing code, e.g. by simply editing the diagram represented by the CASE Tool structure (shown in Figures 36 and 37).

5.7. Concluding Remarks

This chapter described the SEW-OSA CASE tool from a top-down perspective, whereby all the details of the model had to be defined. The details of some of the diagrams described in this chapter (e.g. the activity behaviour diagram shown Figure 31) may convey the idea that the model-building process requires a considerable effort to be completed. That may be the case for designs that start from scratch. Nonetheless, if partially instantiated models of resources and systems are used, the effort required to populate requirements definition and design specification diagrams can be reduced considerably.

Chapter 6 - Model-Enactment for the Purpose of Simulation

This chapter describes the use of Petri-nets for modelling, analysis, simulation and performance evaluation, as a first step towards model-enactment. The main focus of this chapter is on describing the transformation process incorporated in the SEW-OSA CASE tool which enabled the automatic generation of a Petri-net from a SEW-OSA business model.

6.1. Simulation Methodology

In this study, simulation[1] has a twofold objective: to validate the model delivered by the model-building capability of SEW-OSA and to serve as a means of testing alternative configurations embedded in the business model (i.e. to support contingency analysis).

Figure 38 depicts the methodology adopted to support simulation. Here, the behavioural aspects of a complete business model was converted (or mapped) automatically by SEW-OSA into a generalised stochastic time Petri-net (GSTPN) representation. Such a representation captured the elements of information formalised in the templates and diagrams discussed in Chapter 5, which are required to describe the overall behaviour of a system.

The SEW-OSA CASE tool enabled the generation of code in the form of a structured text description, in a format compatible with the input format of ARP (Petri-Net Analyser and Simulator introduced in Section 4.4). Once accessible by the ARP, the GSTPN models can be manipulated so that analysis and simulation can be carried out and performance parameters obtained. The following sections describe each step of the simulation methodology presented in Figure 38 and how they fit into the model-enactment capability of SEW-OSA.

---

1. According to Roberts [Roberts 1984], "a simulation model is a model that produces behaviour over some period of time; it is dynamic instead of static"; for Roberts, simulation means to imitate. According to Pidd [Pidd 1992], "computer simulation involves experimentation on a computer-based model of some system. The model is used as a vehicle for experimentation, often in a 'trial and error' way to demonstrate the likely effects of various policies." He also describes the main advantages of simulation over direct experimentation as being: cost, time, replication and safety.

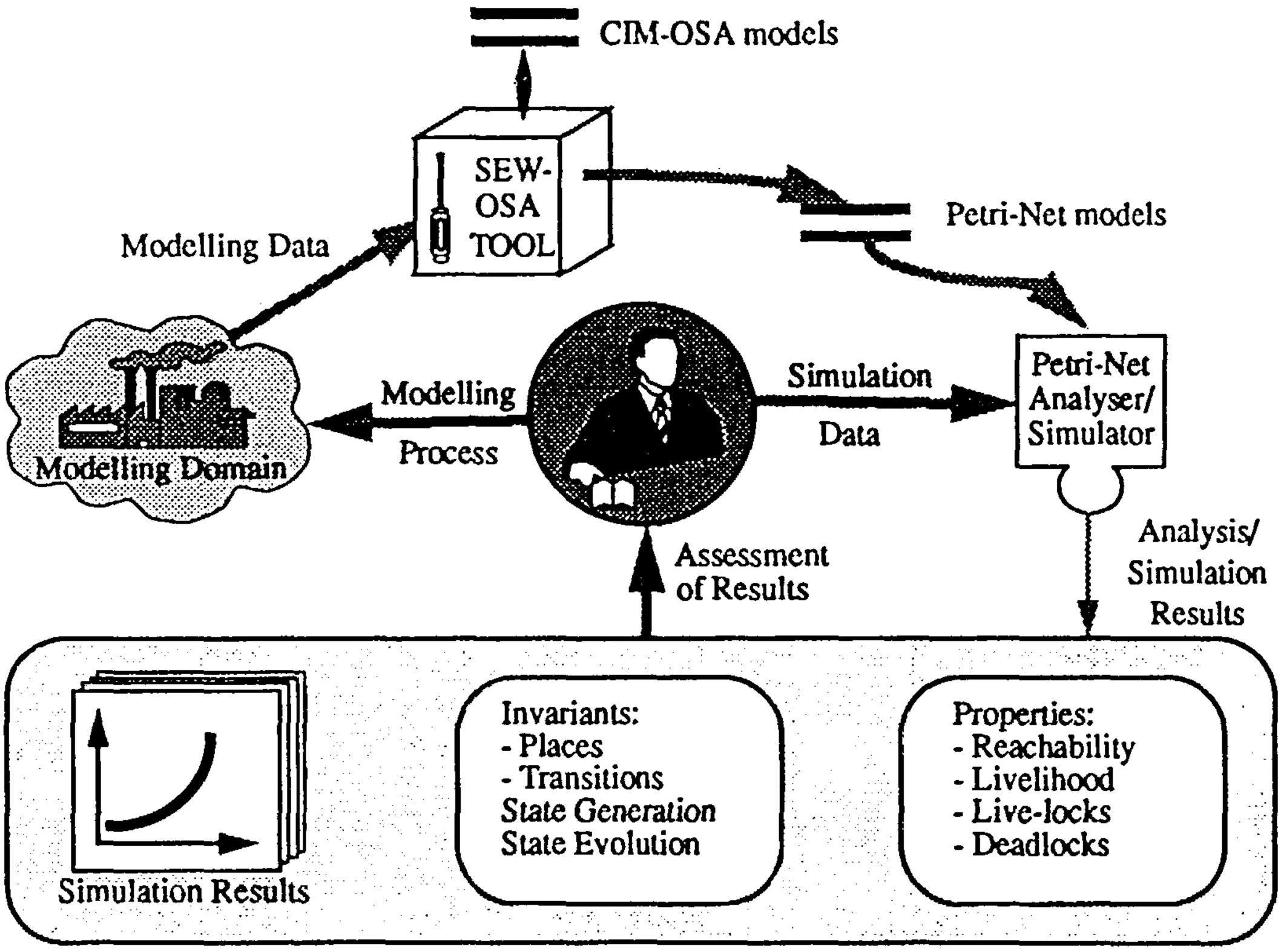


Figure 38 - Overall methodology for modelling and simulation

6.2. From CIM-OSA Models to Petri-nets

The concept of gracefully moving between modelling to simulation requires the dynamic aspects of the modelled system to be visible. In CIM-OSA, the dynamic properties of the modelled system are described by the behaviour model (which carries the "time-glue" that links other system models with the sequence of run-time happenings of system evolution). Execution of the behaviour model is a central aspect of simulation, whereas the remaining modelling aspects (i.e. information, organisation and resource, which are static representations in nature) can be gradually attached to the central description provided by the behaviour model; this in order to define all elements of a system. However, software tools are required to manipulate the models and, therefore, refine them in order to flesh out a complete system definition. This is the major role of the model-enactment capability of SEW-OSA.

As a first stage in the model-enactment process, a means of transforming behavioural aspects of a CIM-OSA model into a Petri-net representation were incorporated into the functionality of SEW-OSA [Aguiar 1993a]. As previously discussed, this transformation was required to take advantage of existing software tools which are capable of processing Petri-net models (e.g. ARP). As shown in Figure 38, such a transformation acts upon a business model created by using the model-building

capability described in previous chapters. The parts of the business model that are utilised in this transformation are constructs related to behavioural aspects of the function view, namely: domains, domain processes, business processes, enterprise activities, events, procedural rules, functional entities and functional operations.

An extension of ordinary Petri-nets, known as a generalised stochastic time Petri-net (GSTPN) [Juanole 1989], was selected as the simulation model (as discussed in Section 4.4). A process which translates from a CIM-OSA model to a GSTPN was implemented by associating a fragment of a GSTPN model (defined in terms of macro-Petri-nets) to each CIM-OSA construct relevant to the behavioural model. The nature of these macros changes according to where they are used in the modelling process. Table 3 shows the relationship between each CIM-OSA construct of interest and its equivalent GSTPN-macro. One should notice from Table 3 that the SEW-OSA CASE tool automatically handles the different configurations in which constructs are submitted within a model. For instance an event can be represented in three different ways, according to its position in relation to the domains that either produce or consume it. A similar consideration is valid for domains which work only as producers or consumers of events (i.e. they work as starters and terminators of activities).

Table 3 - Syntax translation between CIM-OSA and Petri-nets

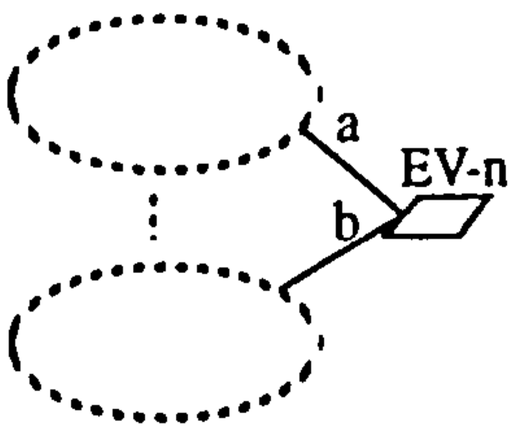
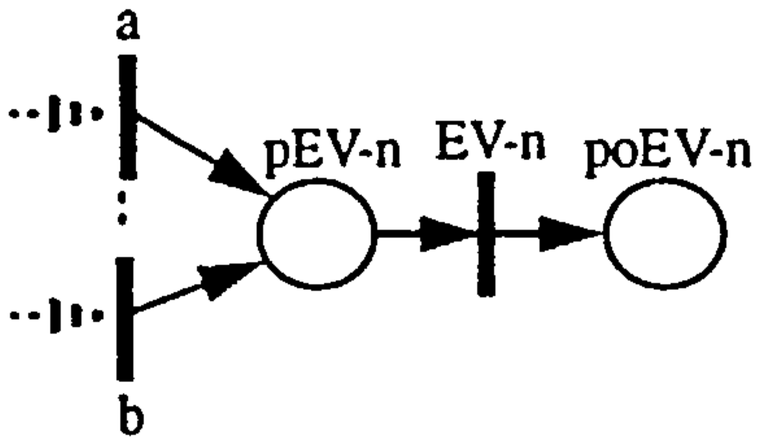
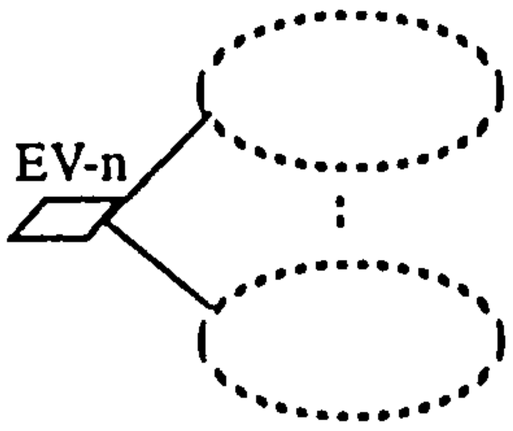
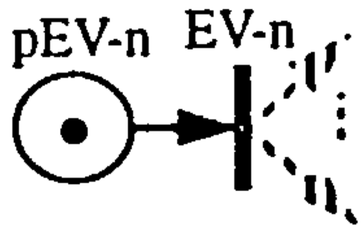
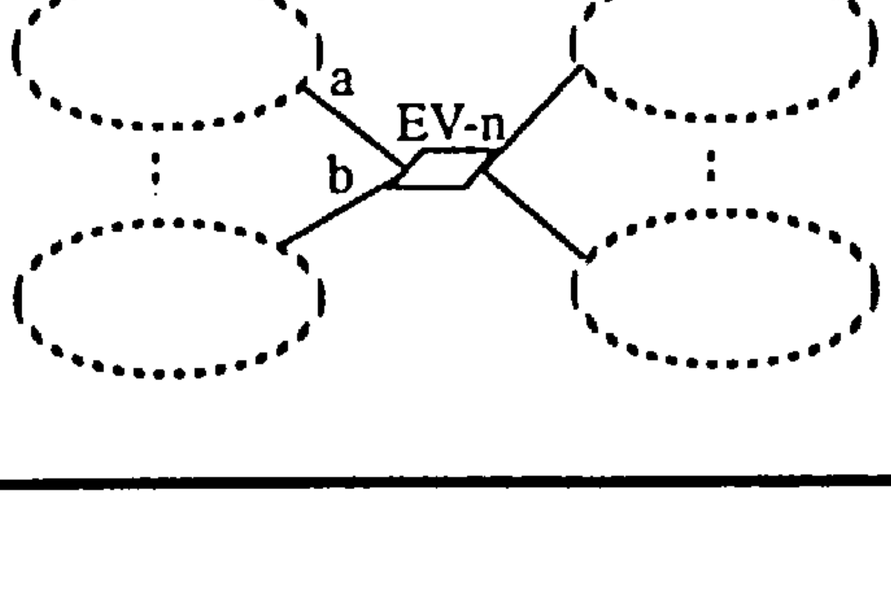
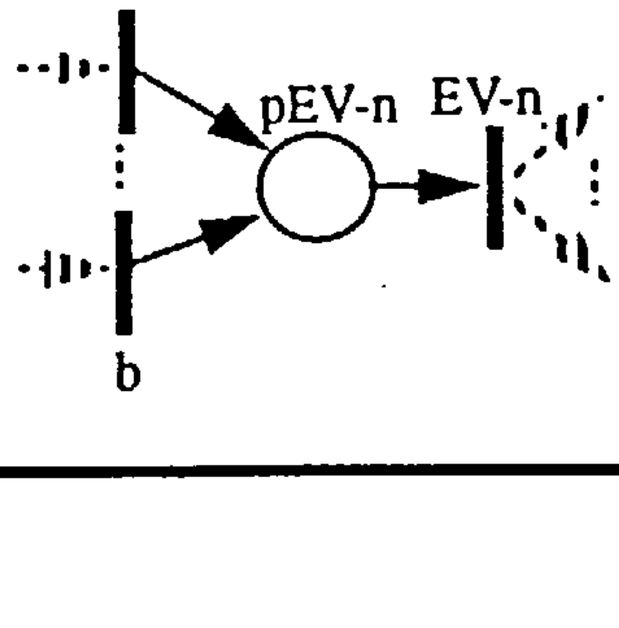
(row) Construct	CIM-OSA	Petri-net
(1) **EV - Enterprise Event** (without any triggering output)		
(2) **EV - Enterprise Event** (without any triggering input)		
(3) **EV - Enterprise Event** (propagating between Domains)		

Table 3 - Syntax translation between CIM-OSA and Petri-nets

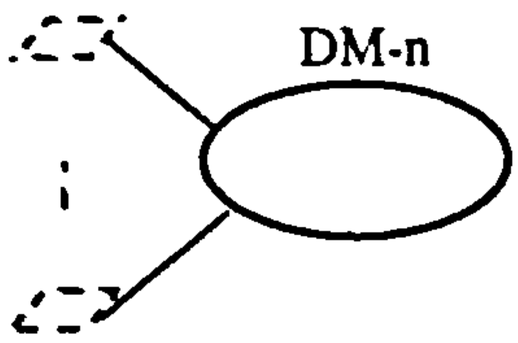
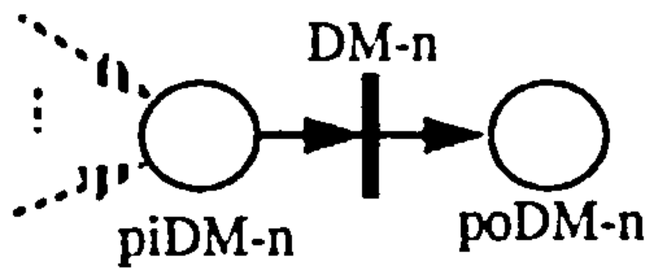
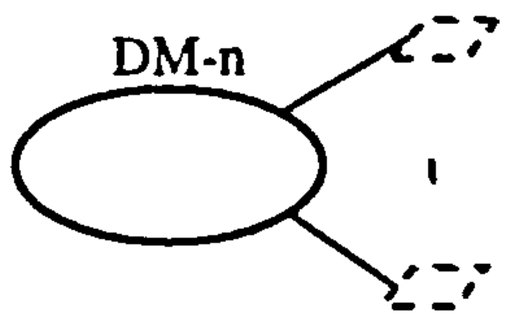
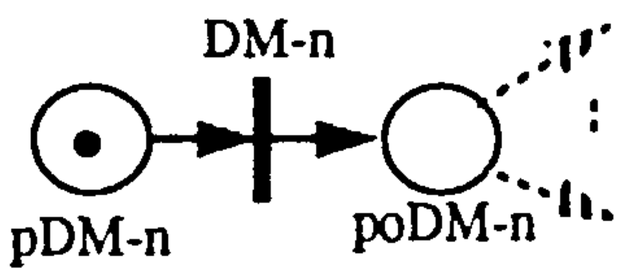
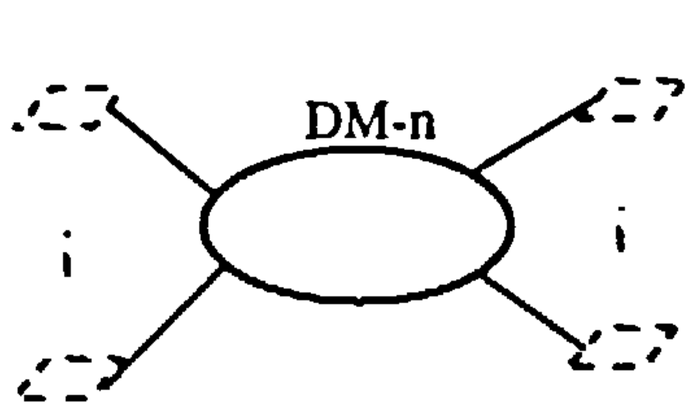
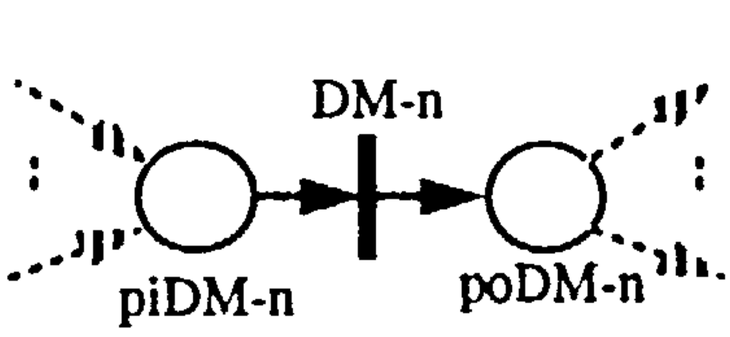
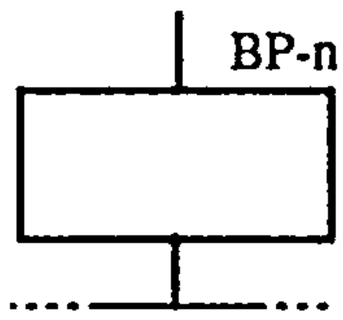
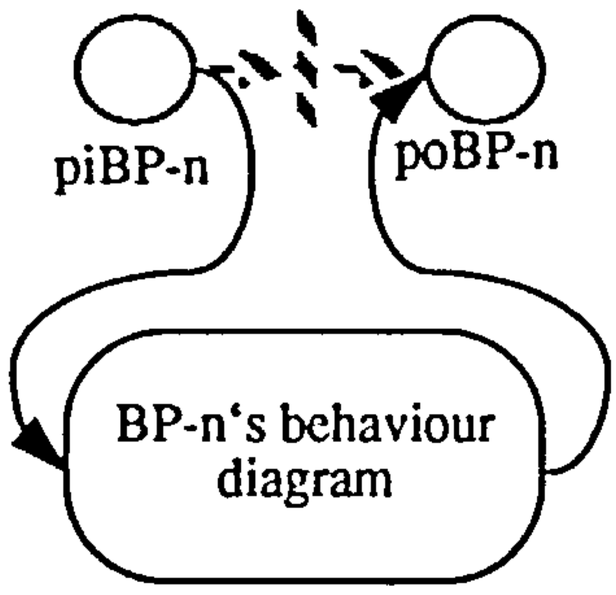
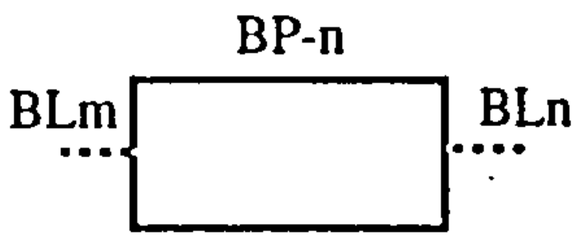
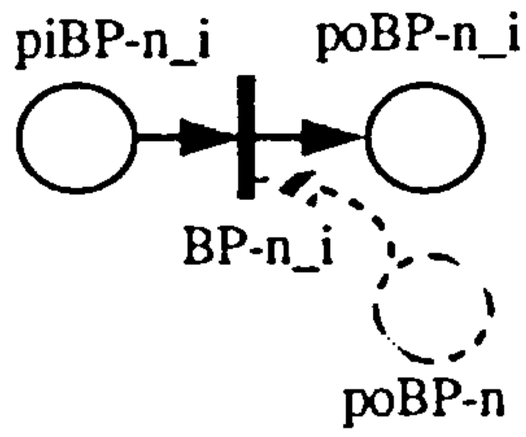
(row) Construct	CIM-OSA	Petri-net
(4) **DM - Domain** (that does not generate any event)		
(5) **DM - Domain** (that is not triggered by any event)		
(6) **DM - Domain** (producer and consumer of events)		
(7) **BP - Business Process** (represented in a structure diagram)		
(8) **BP - Business Process** (represented in a behaviour diagram)		

Table 3 - Syntax translation between CIM-OSA and Petri-nets

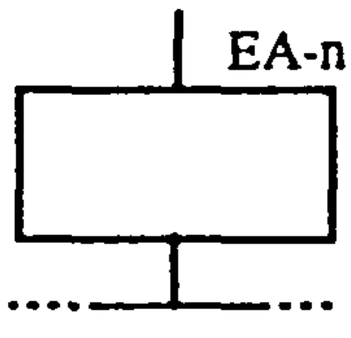
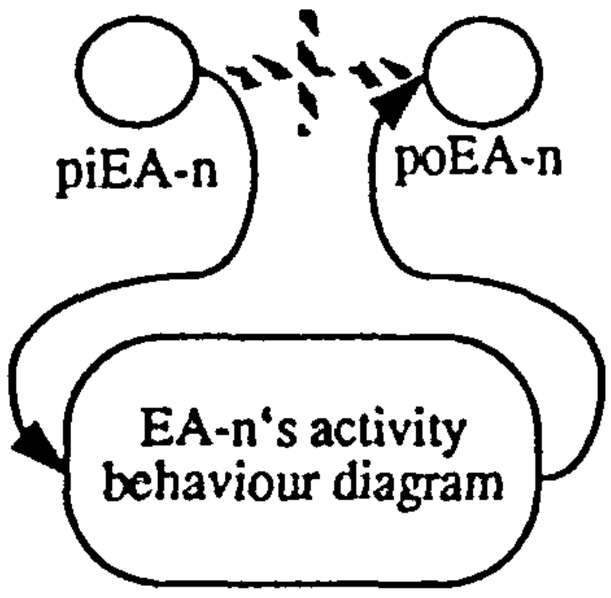
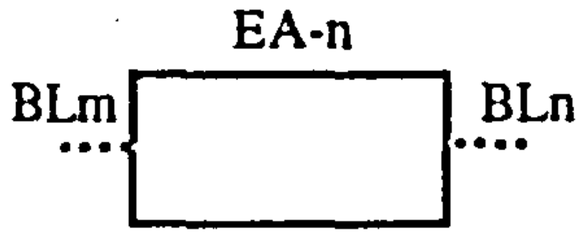
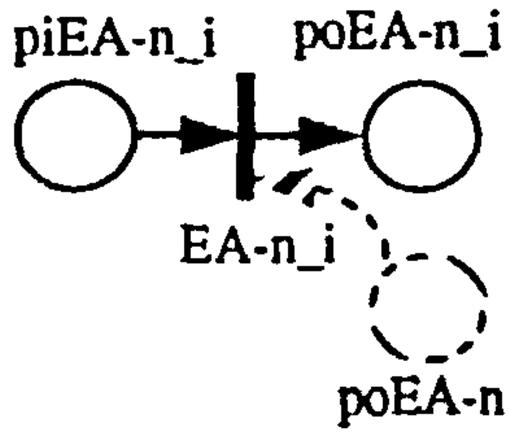
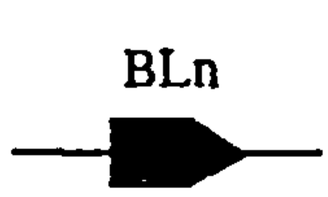
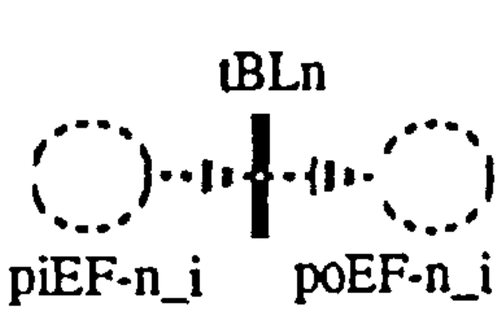
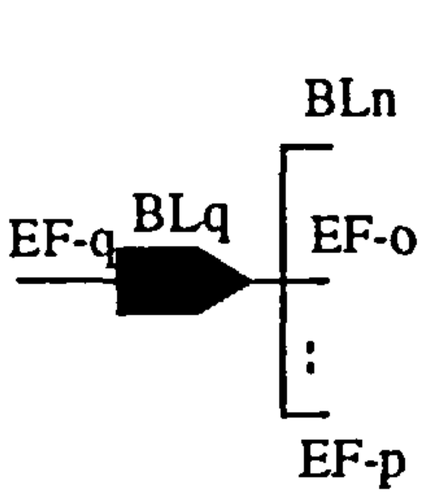
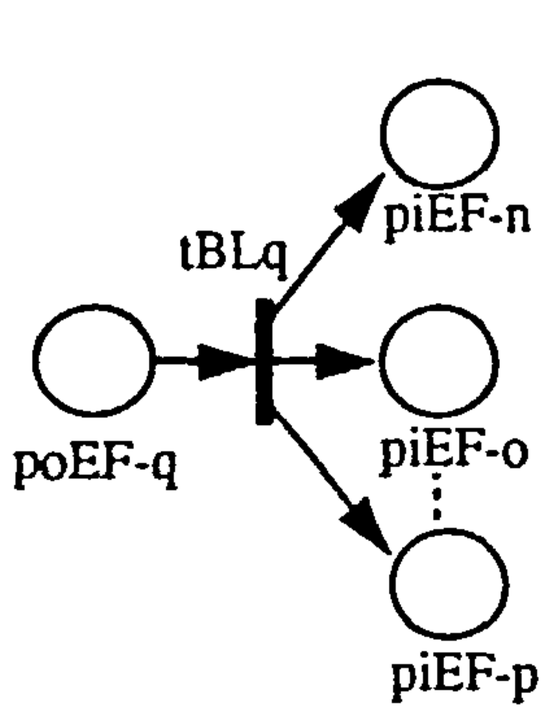
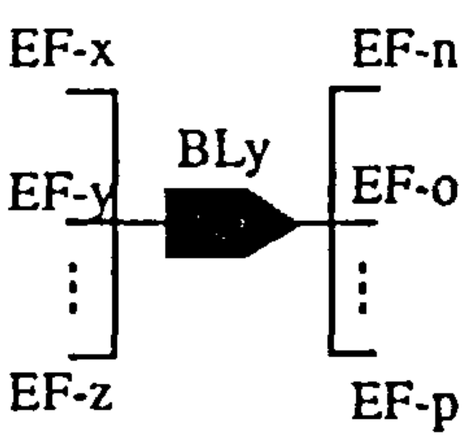
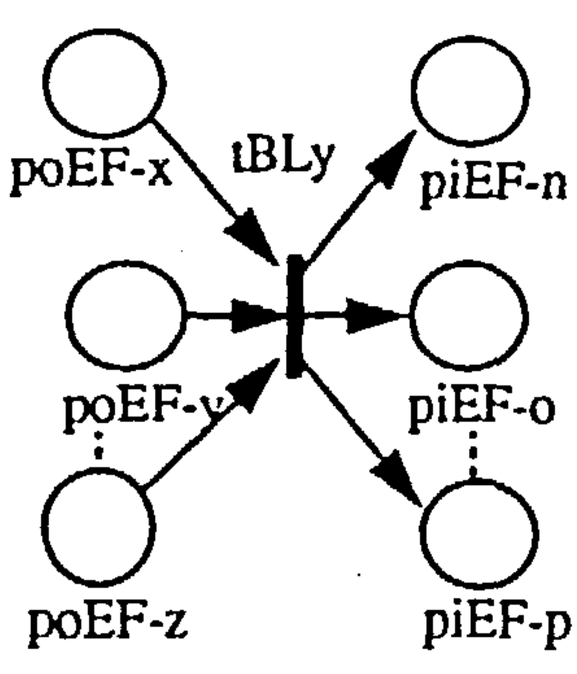
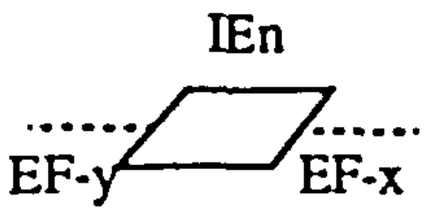
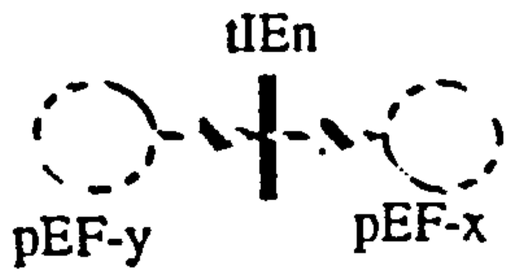
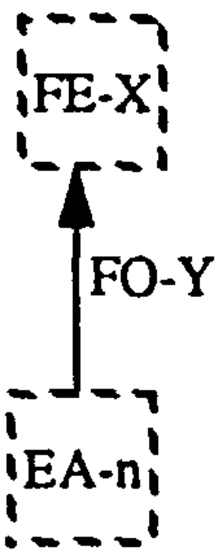
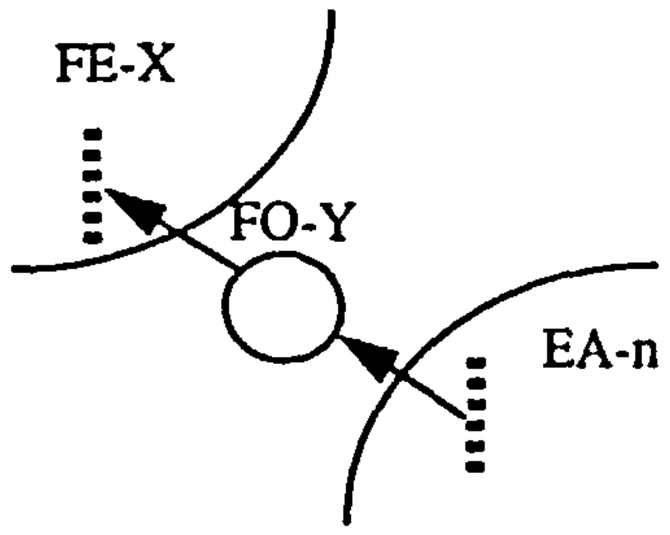
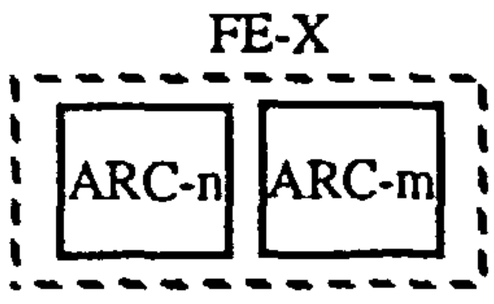
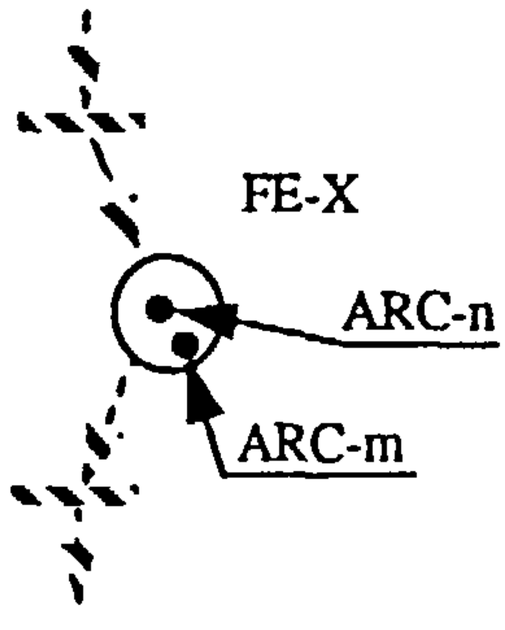
(row) Construct	CIM-OSA	Petri-net
(9) **EA - Enterprise Activity** (represented in a structure diagram)		
(10) **EA - Enterprise Activity** (represented in a behaviour diagram)		
(11) **Procedural Rules 'Forced' (FRC), 'Go/NoGo' (GNG) and 'Loop' (LP)**		
(12) **Procedural Rule Spawning (SP)**		
(13) **Procedural Rule Rendezvous (RE)**		

Table 3 - Syntax translation between CIM-OSA and Petri-nets

(row) Construct	CIM-OSA	Petri-net
(14) **Internal Events** (i.e. Start, Finish or Terminate)		
(15) **Functional Operation**		
(16) **Functional Entity and Active Resource Components**		

The CIM-OSA constructs described in Table 3 relate to the following parts of the model-building capability:

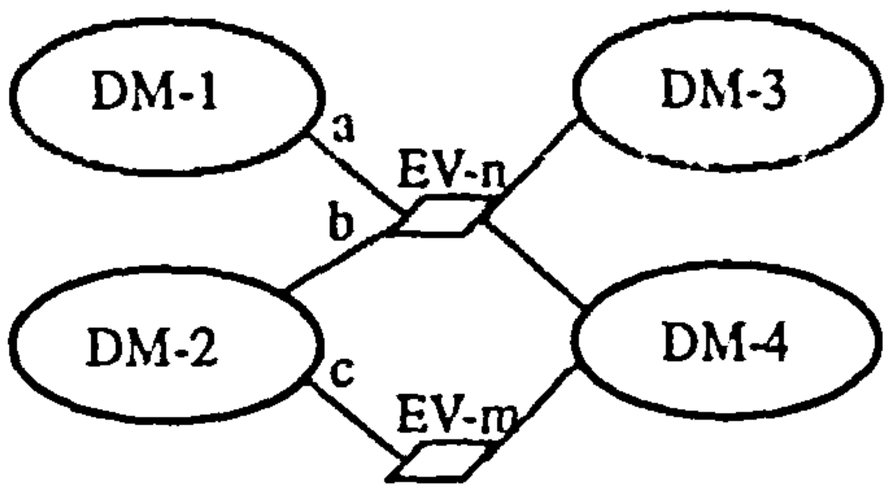
- the first six rows relate constructs manipulated in the context diagram (shown in Figure 23), namely: events and non-CIM-OSA-compliant domains. CIM-OSA compliant domains are not directly represented, for they are decomposed into domain processes which are, in turn, represented by the functions that they contain (i.e. business processes and enterprise activities);
- domain processes are represented directly by their behaviour diagrams (depicted in Figure 27);
- business processes and enterprise activities that appear in a structure diagram represented by Figure 26 (i.e. rows 7 and 9 in Table 3) comprise of the pool of functions which can be triggered by their parents and are described by procedural rules, internal events and other enterprise functions;
- business processes and enterprise activities that appear in a behaviour diagram exemplified by Figure 27 (i.e. rows 8 and 10 in Table 3) comprise of representations

of the parent utilisers of their functionality. An enterprise activity is a parent of an activity behaviour diagram (see Figure 31), whereas a business process is a parent of a process behaviour diagram (see Figure 27);

- rows 11 to 14 relate to the elements manipulated in the behaviour diagrams. In these rows, "EF" identifies either an enterprise activity or a business process being used by its parent (represented by the identifier BP-1EA-2);
- functional operations exchanged between enterprise activities and functional entities (as described in the object diagram shown in Figure 30) are represented by shared places (i.e. a communication channel) through which tokens (i.e. messages) can flow, as represented in row 15 in Table 3;
- active resource components, with which an enterprise activity occurrence interacts, are allocated every time the occurrence is triggered, provided that they are available to perform its functional operations. The allocation of a limited number of active resource components (defined in the diagram of Figure 33) to an 'a priori' unlimited number of occurrences of enterprise activities is represented in the manner indicated in row 16 in Table 3. Here, the number of tokens in the place "FE-X" indicates the number of active resource components available within the pool of resources represented by "FE-X";
- enterprise activities and active resource components are already represented in a predicate-action Petri-net form (as shown in Figures 31 and 32). In the translation process between predicate-action Petri-net to GSTPN, the predicates which are related to internal processing of the enterprise activities and active resource components have been disregarded.

The rationale behind these macros implies certain implementation decisions. For example, events linked to a number of domains, such as in Figure 39, should be interpreted as if an occurrence of EV-n could be generated independently by either of the producer domains (i.e. DM-1 and DM-2), whereas any occurrence of EV-n propagates to both its consumer domains (i.e. DM-3 and DM-4). However, whether an occurrence of EV-n propagated to a domain actually triggers functions within that domain, depends upon the following conditions:

- if the domain is non-CIMOSA compliant, whatever functions that domain embraces, a triggering condition should occur;
- if the domain is CIM-OSA compliant, the occurrence of EV-n is latched (i.e. memorised) within the domain and submitted to logical combinations of conditions that may exist between EV-n and any other event or ending-status of a function belonging to that domain. For example, if a procedural rule condition requires EV-n



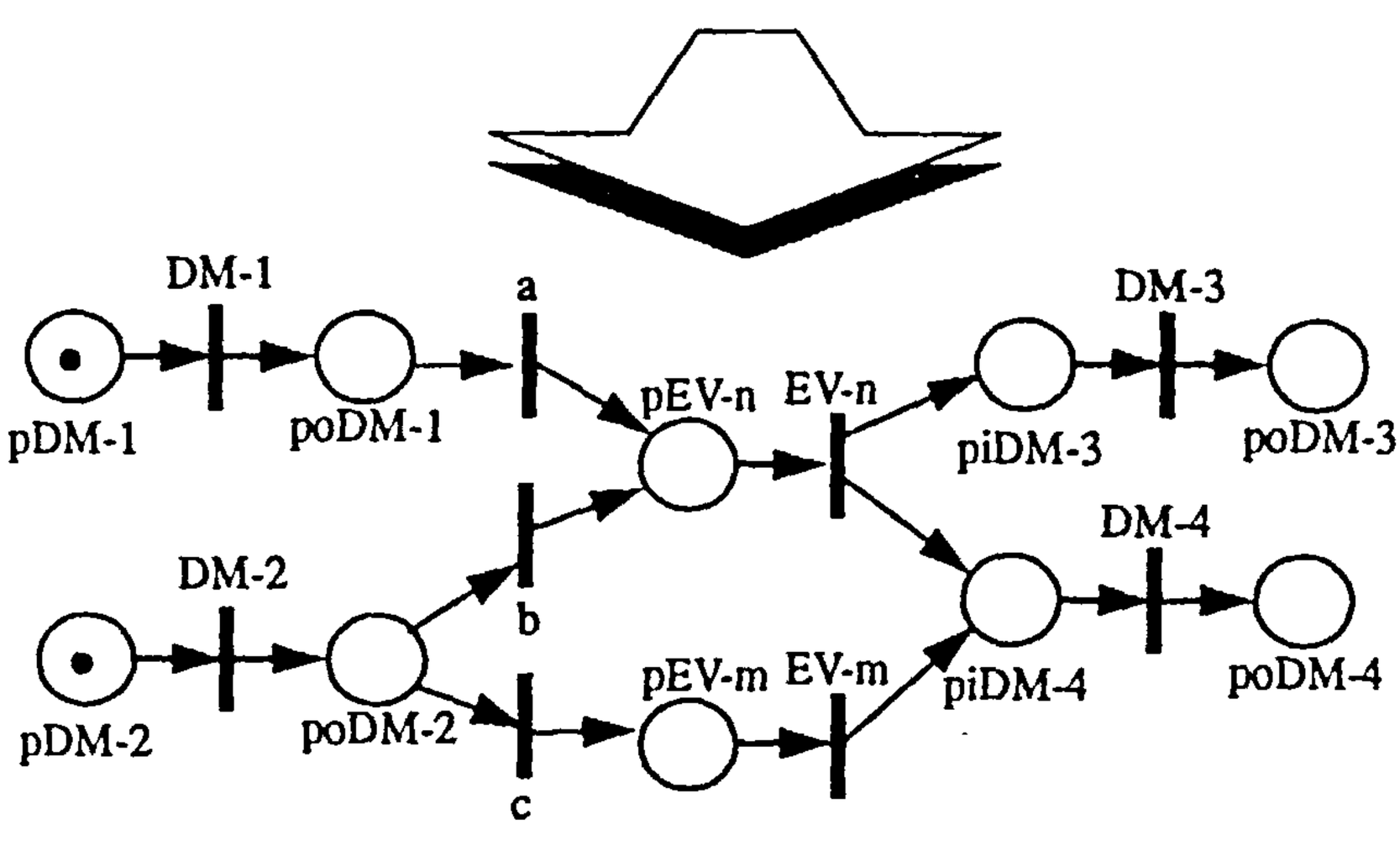


Figure 39 - Example of a Petri-net for a fragment of a context diagram

and EV-m to occur in order to trigger a function, any occurrence of either EV-n or EV-m will be latched in order to await for its counterpart to occur (so that a thread of functional execution may start). A number of occurrences can be latched to be combined with their event counter-parts.

Another important feature incorporated in the facilities for constructing CIM-OSA models (which further complicated the transformation process) concerns the sharing of enterprise functions by distinct procedural rule sets (i.e. the sharing of functions in the description provided by behaviour diagrams, as represented by Figure 27). As these functions can execute concurrently, they may compete for the same resources. Enterprise functions are represented as a "pool" of models that are used by the distinct procedural rule sets by means of shared places. Figure 40 illustrates how such a feature is implemented in SEW-OSA. Here, the model of a business process or enterprise activity is referred in the behavioural description of the enterprise function and uses them as sub-nets (i.e. child Petri-nets). The same sub-nets can be used in behaviour diagrams of other enterprise functions. The fact that the same model is shared by the procedural rule sets of distinct enterprise functions enables the designer to assess issues of utilisation level of certain enterprise functions which, ultimately, lead to means of determining the utilisation of the resources associated with them.

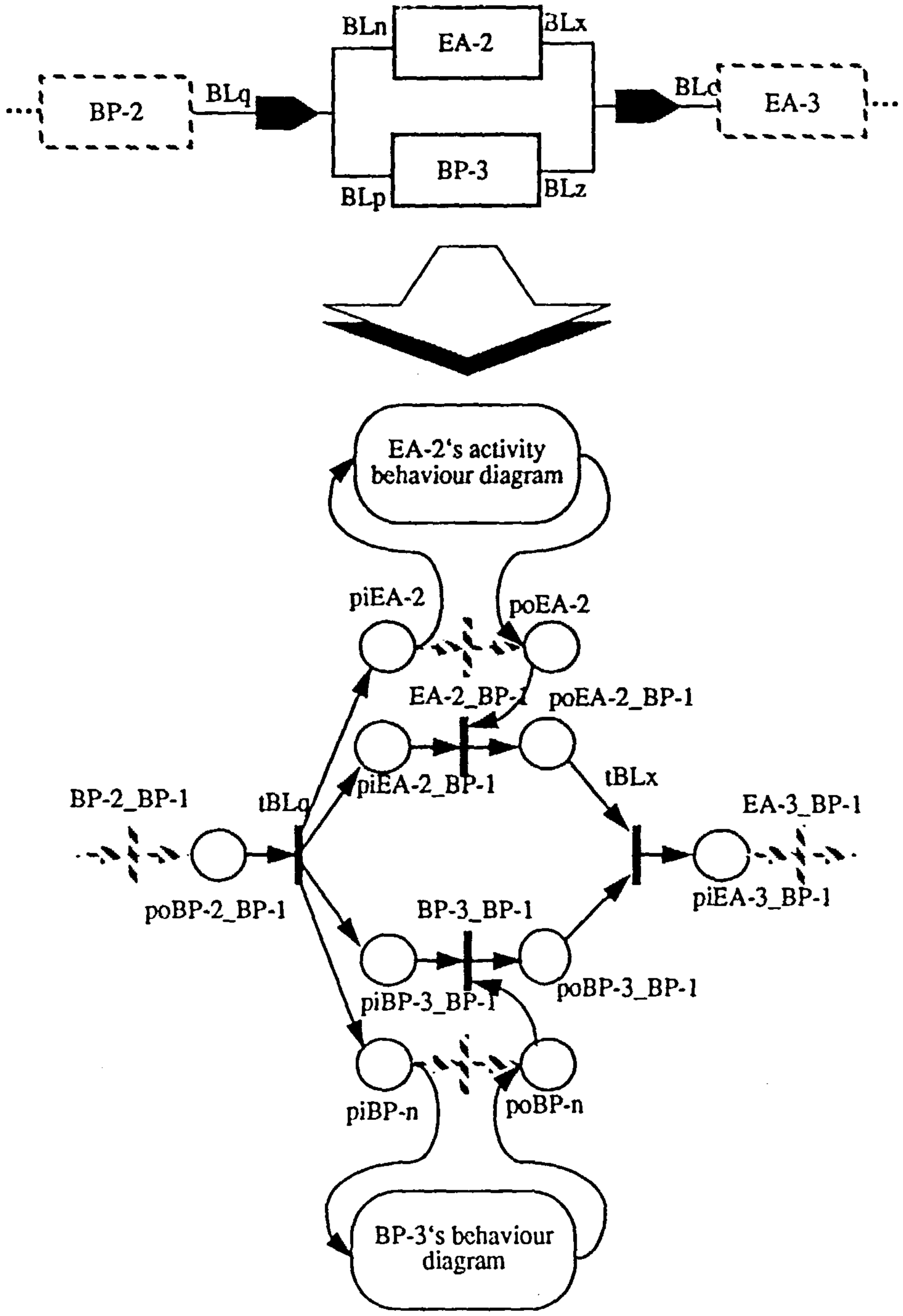


Figure 40 - Example of a Petri-net for a fragment of a behaviour diagram

Petri-net macros presented in Table 3 summarise qualitative information associated with the transformation process. However, GSTPN also incorporates data about time and probability functions associated with transitions (as explained in Appendix 4). Here, relevant data about time is extracted from attributes defined in the templates of functional operations (as exemplified by Figure 41). However, all

remaining constructs that are not decomposed down to functional operations (e.g. non-CIM-OSA compliant domains and events) have their time intervals set to zero. When no data is input by the designer in the "Simulation Data" attributes of a functional operation template, a null time interval is assumed by the SEW-OSA CASE tool. The same criterion applies to all other constructs that hold information relevant to the GSTPN models which have not been defined when the model is generated.

FUNCTIONAL OPERATION TEMPLATE:

TYPE:	Manufacturing
IDENTIFIER:	FO-8
NAME:	Populate PCB
DESIGN AUTHORITY:	Marcos Aguiar
DESCRIPTION:	Commands the placement machine to populate a batch of PCB

Simulation Data:

- Time Interval: [64, 72]
- Probability Distribution: NORMAL

Execution Data:

make(PCB)

Figure 41 - Example of a functional operation template

6.3. Analysis and Simulation

Based on the method implemented for transforming CIM-OSA models into GSTPN descriptions, code is generated in a format interpretable by ARP (see Figure 38). As stated in Section 4.4, this particular tool allows: editing of the model generated by SEW-OSA (so that minor changes can be made to suit simulation tests without re-generating the Petri-Net model), analysis of the basic properties of a Petri-net; analysis of invariants (of places and transitions); step-by-step simulation of the Petri-net's state evolution; and performance analysis of certain parameters of the system, based on its execution in simulated time (i.e. dynamic simulation of its state evolution).

6.3.1. Editing

A Petri-net model is generated by the SEW-OSA CASE tool in a structured text form (rather than graphically). This text defines places, transitions and how they are

inter-linked. The syntax that the ARP is able to interpret is shown in Figure 42. An example of such a description populated with data from a case study is presented in Appendix 5. New or altered nets require compilation (i.e. an internal ARP operation that checks the net for syntax errors) before any of the operations described in the following subsections can be performed on the net.

```
NET net_name; {comments are allowed between brackets}

CONST
        constant1, constant2,... = value;

NODES
        name1, name2,...: TRANSITION {[Tmin, Tmax]} {DISTRIBUTION (1000)};
        name1, name2,...: PLACE {(initial number of tokens)};

STRUCTURE                {input}                {output}
        trans_name: (place,weight*place,...),(place,weight*place,...);

ENDNET.
```

Figure 42 - ARP syntax

6.3.2. Analysis

ARP is able to perform the following analyses on a Petri-net: analysis of invariants of places and transitions; analysis of properties (i.e. limitation, conservation, livelihood, multi-sensibilisation, re-initialisation, livelocks and deadlocks); and analysis of state (i.e. generation of the table of markings associated with the net evolution). These analyses can be a powerful aid to checking the dynamic behaviour of a Petri-net for undesirable features. As an illustration, livelihood gives an indication as to whether all transitions can be triggered in a model. That is, a non-live transition indicates that a certain event in a system never occurs which could point to modelling error. Livelocks and deadlocks identify confined states in a system (which are essentially undesirable). Re-initialisation reflects the ability of the net to return to its initial state (after processing an incoming event).

6.3.3. Simulation

ARP also provides means of executing a Petri-net on a step-by-step basis. The simulation option enables the user to check for minor deviations in the behaviour of the system by controlling the execution of a net on a transition-by-transition basis.

6.3.4. Performance analysis

One of ARP's most important features is its capability to support performance analysis. This is based on computing the time the net takes to evolve from the

occurrence of an initial event to the occurrence of one or more final events. This allows certain dynamic characteristics of a manufacturing solution to be studied (e.g. lead-time, production rate, etc.). Additionally, by computing the average number of tokens in each place (during a simulation) ARP enables study of variables such as work-in-progress and level of utilisation of resources.

Figure 43 depicts an example of the type of results obtained from using this performance analysis capability. Here, the impact upon the manufacturing lead-time (of an assembly line) of two parameters (namely checking time and rate of PCB produced) is illustrated to provide a flavour of the issues that can be investigated. These results were obtained in an early study of D2D's shop-floor [Aguiar 1993a]. A more extensive discussion of results obtained from this case study is included later in this thesis.

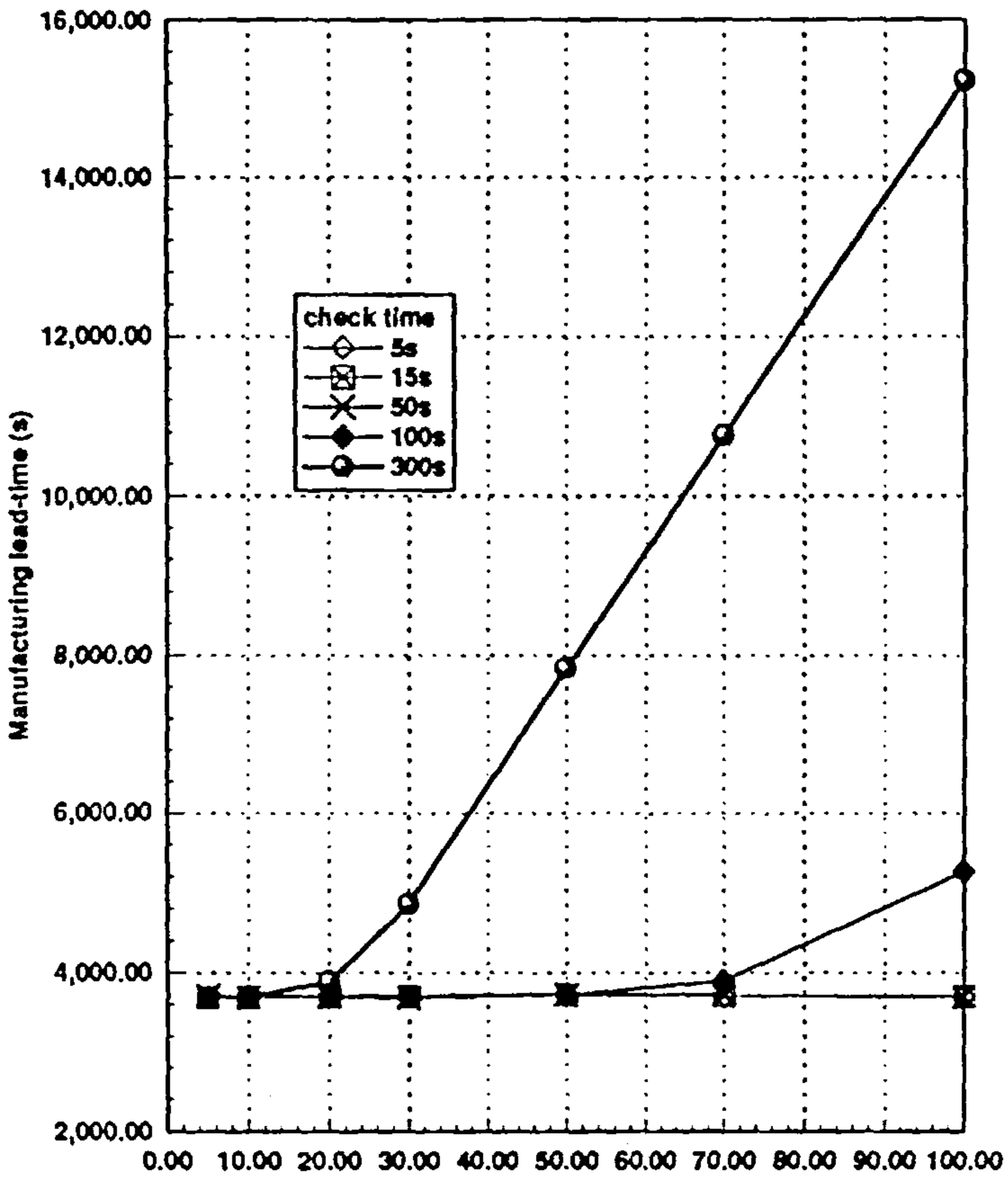


Figure 43 - Manufacturing lead-time as a function of checking rate and time

Corresponding to each point in Figure 43, a report is generated by ARP, thereby documenting simulation results for a particular configuration of the model. Example of reports are presented in Appendix 6.

6.4. Simulation using the Model-Enactment Capability

This section describes how ARP features were utilised to support design decisions within the design and implementation phases of the IMS life cycle. As discussed in previous chapters, model-enactment is proposed to be achieved by SEW-OSA in a gradual manner, the first step being the ability to perform simulation based on use of the models. In respect to simulation, this gradual process should be supported by an iterative process, which embraces:

- validation of requirements definition and design specification models, this includes support for evaluating performance considerations associated with the selection of candidate solutions (in terms of system configuration and its components);
- execution of the overall system model through simulation;
- assessment of performance results obtained from executions undertaken on alternative system components and configurations (i.e. alternative ways of organising the interactions amongst system components).

Such a process implies the execution of two levels of analyses on the business model, namely: analysis of behaviour and analysis of system configuration.

a. Analysis of behaviour

This stage consists of transforming the business model of a system and its components into a single Petri-net. The aim here is to analyse the behaviour of the system from a static perspective (by examining the properties of a Petri-net model), as well as from a dynamic viewpoint (by means of discrete event simulation). As part of the behaviour analysis, time constraints and concurrency issues are analysed.

This analysis is performed by creating simulated scenarios which are determined by: (1) how events that trigger activities in the business model are generated; (2) the values of particular time parameters associated with functional operations (see Figure 41); (3) the number and configuration of system components; and (4) the structure of the actual business model. When all scenarios of interest have been evaluated by ARP, the design specification model can be considered to be valid.

Analysis of behaviour is supported by the simulation features described in this chapter. Such features are particularly relevant at the early stages of the IMS life cycle, where emphasis is on gaining an understanding about how the system works based on the manipulation of its model.

b. Analysis of system configuration

The aim of this stage is to derive a robust business model, as well as an adequate specification of system components.

At this stage, the business model is executed in such a way as to interact with prototypes of system components distributed across an integrating infrastructure (as illustrated in Figure 9).

Thus, the business model is no longer described in a unique Petri-net. Rather the model contains refinements of the model manipulated during simulation (whereas models of system components are selected according to a methodology discussed later in the thesis). From the perspective of the business model, system components described by models should behave in the same way as actual physical components, in regard to: object view consumption and production, resource utilisation and execution time of functional operations.

Nevertheless, the overall behaviour of the business model should be the same as in the previous stage (i.e. analysis of behaviour). This allows execution of similar tests to those performed in the previous stage, as a means of comparing performance issues.

The main objectives of these two stages are to check the model for design errors and assess design decisions embedded in the model. Based on the results of these tests, changes in the model may arise which, in turn, may require further iterations between modelling and simulation. Iterations between model-building and model-enactment are repeated until satisfactory results are obtained. Simulation in the context of the model-enactment capability, is limited to support of the behaviour analysis stage. Analysis of system configuration is supported by the capabilities described in the next chapter.

6.5. Limitations

The limitations encountered in using the methodology depicted in Figure 38 for a behaviour analysis of the business model have two primary roots, namely:

a. CIM-OSA/Petri-net model transformation:

Some of the limitations inherent in the transformation macros proposed in Table 3 are as follows:

- no equivalent GSTPN representation was constructed for representing different "ending-statuses" of termination of enterprise functions.

  Basically, enterprise functions are modelled as if only one "ending-status" is possible. Such a limitation implies that decisions that are taken based upon

"ending-statuses" are not captured by the Petri-net models. In models where "ending-statuses" are intensively used, the Petri-net model may not reflect the actual behaviour of the system. Such a limitation is intrinsic of the type of Petri-net used in this model. In order to overcome this limitation, more powerful extensions of an ordinary Petri-nets should be used (e.g. Coloured-Petri-nets).

However, this limitation is not relevant to the analyses and simulations executed by the ARP, for one of its features is to enable descriptions of the probability of a certain ending status to occur (i.e. association of probabilities between the firing of transitions that are in conflict).

- passive resource components were not represented in the transformations, although they can be included in a similar manner to active resource components.
- the over-head introduced by an integrating infrastructure was not included in the Petri-net models, for they may vary considerably according to system configuration decisions and integrating infrastructure used. Indeed, studies of the impact of the over-head upon system performance is one of the issues that this thesis aims to investigate.

b. Simulation Tool

Certain limitations which are intrinsic to the tool adopted for simulation (i.e. ARP) are discussed as follows.

- Lack of model animation. Currently, only a text-based interface can be used to obtain results from ARP.
- Loose integration between the simulator and the SEW-OSA CASE tool. Interaction between the SEW-OSA CASE tool and the simulation tool is via file transfer between a Unix platform (on which the CASE tool is run) to an MS-Windows platform (on which the ARP is run). This limits the synergy that can be realised by combining the environments. This limitation could be overcome by re-implementing ARP on a Unix platform (or SEW-OSA on a MS-Windows platform).
- Although ARP suits the requirements of this research (specially in regard to qualitative analysis of systems, e.g. deadlocks, livelocks, etc.), it does not provide various facilities typically offered by discrete event simulators used for manufacturing system simulation (e.g. animation).
- The translation process incorporated into SEW-OSA generates large Petri-net models from fairly small CIM-OSA models. However, ARP was initially conceived to execute small Petri-nets (i.e. Petri-nets containing a maximum of 150 places and 150 transitions). Thus, the simulation exercises which were carried out had to use

models of limited size.

All these limitations can be readily overcome if SEW-OSA is interfaced with more powerful tools. These tools could either be more sophisticated Petri-net simulators (e.g. Unisson [Bonney 1992]) or simulation tools traditionally applied to logistic analysis of manufacturing systems. In either case, the code generated by SEW-OSA would have to be re-implemented in order to comply with the particular syntax used by these tools.

6.6. Concluding Remarks and Contribution

Arguably, the activities reported in this chapter, which were developed within the first of year of the author's Ph.D., comprise one of the first instance of an application of CIM-OSA to a problem of real industrial interest. They show how CIM-OSA models can be used to analyse and simulate issues which are relevant to design decisions in manufacturing systems. The results of these activities were published in a conference paper and a journal paper, the latter being awarded the "1993 Donald Julius Groen Prize" (i.e. "article of the year" in the area of "communication and control") by the Institute of Mechanical Engineers (see Appendix 7).

This work has also demonstrated how Petri-nets could be used in conjunction with CIM-OSA, by implementing and applying their formalism within a CASE tool. This was achieved by proposing and validating the use of a detailed mapping between the behavioural modelling constructs of function view and generalised stochastic time Petri-nets. Such a comprehensive mapping is both a novel and relevant approach to enabling modelling and simulation.

Chapter 7 - Model-Enactment for the Purpose of Rapid-Prototyping

This chapter describes the way in which the business entity of SEW-OSA was implemented and how it works. The business entity incorporates an associated model debugger as an aid to model-enactment. The most relevant features of the business entity are presented, highlighting the importance of supporting the rapid-prototyping[1] of manufacturing systems from model-based descriptions.

7.1. Business Entity Components

Once a business model is considered to be satisfactorily well tested through analysis and simulation runs, the rapid prototyping phase can start. At this stage, the same model used to generate Petri-nets for analysis and simulation purposes can be used to generate the various fragments of code required to define a system configuration and its components. Such fragments are then made available in an interpretable form to the business entity of SEW-OSA. The business entity is a layer of domain specific services sitting upon the CIM-BIOSYS integrating infrastructure which, in addition to executing the business model, serves as a model debugger (this structure is represented in Figure 9).

In order to understand the complexity embodied within the business entity, the following is worth noting. The business model encapsulates a description of the types of functions that need to be executed by the identified resources. However, at run-time, a number of occurrences of such function types may be executed concurrently. Execution of these occurrences (also referred to in this thesis as **threads of the business model**) must be managed so that they maintain their identity whilst being executed by a limited number of resources. The function of the business entity is to provide such a level of management, namely: to execute a number of threads of the business model simultaneously and to allocate system components to perform the functions captured in the business model.

Figure 44 presents the overall data flow of SEW-OSA which emphasises the set

---

1. *Rapid-prototyping characterises the engineering process by which a design description is* rapidly realised in order for it to be tested. In regard to the software development process, Brown [Brown 1993] proposes the "V" model to describe such a process, in which requirements definition and design/specification activities constitute one leg of the "V" and realisation and maintenance activities the other leg. Rapid-prototyping is characterised by Brown as cutting the "V" transversally, by enabling the progression from specification to implementation directly, thereby skipping tedious processes of generation and validation of paper specifications.

of models that are generated by the CASE tool. These fragments of models contain both the various facets of the business model that will be used to control the interactions among system components, and the code structure for the core functionality of the components. As depicted in Figure 44, the CASE tool generates five sets of models:

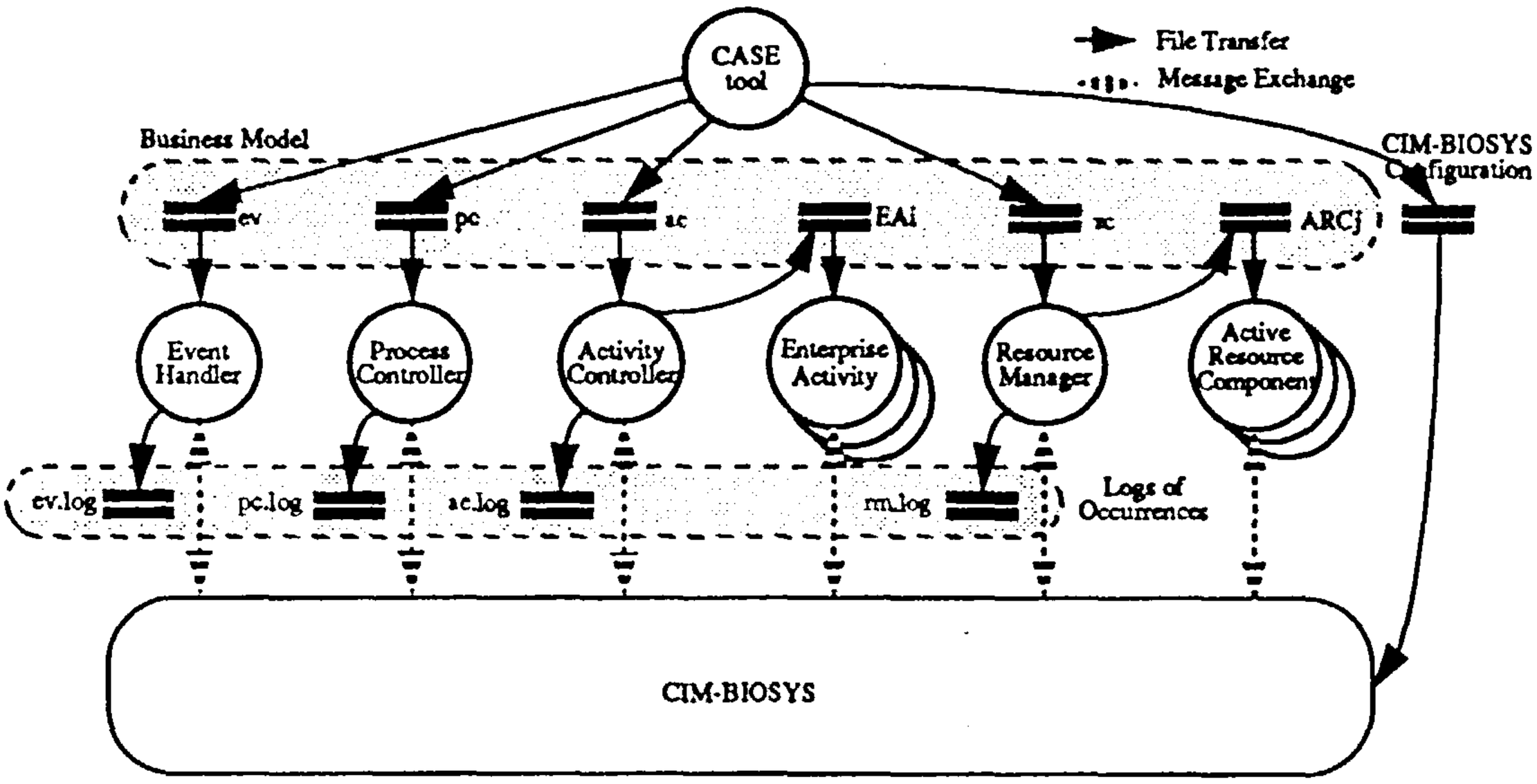


Figure 44 - Overall data flow in SEW-OSA

- ev: describes all events (i.e. asynchronous happenings) in the system.
- pc: lists all functions (i.e. domains, domain processes, business process and enterprise activities) and defines the procedural rule sets which inter-relate these functions according to the information formalised in behaviour diagrams (as defined in Figure 27).
- **ac**: lists all enterprise activities and functional operations used by them. This file also holds the code structure of enterprise activities generated from activity behaviour diagrams (as defined in Figure 31).
- **rc**: lists all active resource components and functional operations that they are able to execute. This file also contains the code structure of active resource components generated from entity behaviour diagrams (as defined in Figure 32).
- **cf**: consists of information required to configure CIM-BIOSYS in order to interact with applications associated with the business entity and those representing active resource components (as defined in Figure 34).
- **EAi** and **ARCj** is the code describing the internal behaviour of enterprise activities and active resource components.

An instance of pc is shown in Figure 45. This figure shows the information extracted from behaviour diagrams and converted to a format that can be interpreted by the business entity. This information is used to configure the components of the business entity of SEW-OSA which ultimately will run as CIM-BIOSYS applications[1] (as illustrated by the interactions in Figure 44). Each component is responsible for certain functions of the business entity which are associated with the normal operation of the modelled system.

As part of the debugging functions provided by SEW-OSA, each component interacts with the designer through appropriate interfaces. These interfaces provide functions which are standard for all four major components of the business entity (i.e. the Event Handler, the Process Controller, the Activity Controller and the Resource Manager), thereby including a capability to:

- check the list of constructs present in fragments of the model handled by the component (e.g. the fragment of the model is presented to the user as a list of events in the Event Handler and functions in the Process Controller);
- set the component to automatic mode so that its functions are executed without interference from the designer, although he (or she) can still monitor occurrences. The monitoring process in the debugging interface consist of scrolling lists of happenings, i.e. events that are continually updated;
- set the component to manual mode where the designer is able to intervene with processing functions of the business entity. Examples of such interventions are the generation of external events in the Event Handler, or enabling step-by-step execution of model occurrences in the Process Controller, Activity Controller and Resource Manager;
- generate a log of happenings concerning model execution at each business entity component (see Figure 44). Log files contain records of all happenings that occur within the component (i.e. all messages issued to the designer concerning major steps of model execution and control messages exchanged with the remaining business entity components). A time stamp (to the resolution of a micro-second) and an occurrence identifier are associated with each thread of execution of the business model. The record of time figures and occurrence identifiers obtained for each component can then be used to assess the overall performance of the system.

The differences between time figures obtained here with the log files and those

---

1. A CIM-BIOSYS application designates a software process which uses the services provided by the CIM-BIOSYS infrastructure.

```
CIM-OSA Design: icl

Process Model:


DM-6/PCB Assembly
  DP-1/Unique Line Type
    BP-1/Preparation
    BP-2/Populating
    BP-3/Finishing
    EA-1/Print
    EA-2/Monitor
    EA-3/Place
    EA-4/Inspect
    EA-5/Reflow Solder
    EA-6/Washoff

DP-1 {

ON (EV-1) DO BP-1
ON (ES(BP-3) = "any") DO EV-2
ON (ES(BP-1) = "done") DO BP-2
ON (ES(BP-2) = "done") DO BP-3

}

BP-1 {

ON (START) DO EA-1 & EA-2
ON (ES(EA-1) = "done" & ES(EA-2) = "done") DO FINISH

}
.
.
```

Figure 45 - Example of a Process Controller input file

obtained through the simulation runs described in Chapter 6 are that: (1) the former represents 'real time' whilst the latter relates to simulated time; (2) the latter does not consider the overhead imposed by the integrating infrastructure concerning message exchange and processing time whilst the former does; and (3) the former individualises each occurrence of model execution (even though it supports the execution of a number of occurrences simultaneously), whilst the latter cannot provide performance information about individual occurrences.

A sample log for the Event Handler is presented in Figure 46.

7.1.1. Event Handler

The Event Handler is responsible for initiating a thread of business model execution, this by channelling event occurrences (e.g. order release) initiated from

```
Log events generated by application 'pc':

18/05/1994-17:02:55.803 > DP-1/Unique Line Type triggered by: EV-1 0

18/05/1994-17:02:56.137 > Function(s) 'BP-1 ' triggered

18/05/1994-17:02:59.322 > Function(s) 'EA-1 EA-2 ' triggered

18/05/1994-17:03:18.819 > Killing Process Performed

18/05/1994-17:03:18.825 > Reset process performed
```

Figure 46 - Example of a Process Controller log file

outside the scope of the business model as well as monitoring event occurrences generated within the business model. A unique number is assigned to an event or a combination of events. Thereby, identification of a thread of model execution (i.e. an occurrence) is propagated to the rest of the business entity, so that the thread of model execution can be individualised even when it is competing with a number of other concurrent occurrences.

As an additional debugging feature, the Event Handler interacts with the designer through the interface depicted in Figure 47. This interface enables the designer to generate events in a "manual triggering mode" (these events being selected from the list at the top of the X-Windows user interface depicted in Figure 47). Events that are accessible to the designer are those that trigger a thread of model execution (i.e. events internal to a CIM-OSA-compliant domain, events generated by non-CIM-OSA-compliant domains or events that have no producers, only consumers).

7.1.2. Process Controller

The Process Controller is responsible for executing occurrences of domain processes, this based on underlying procedural rule sets of its component processes. The Process Controller is able to cope with concurrent execution of several occurrences of the domain process. This is achieved by the Process Controller through its spawning of local processes (i.e. Unix-based processes which do not communicate via the CIM-BIOSYS infrastructure) representing domain processes, which are responsible for the execution of their individual sets of procedural rules. In turn, these domain processes may spawn other processes (representing their component business processes, according to the structure defined in Figure 26) to execute their particular sets of procedural rules (as defined in their behaviour diagrams, as defined in Figure 27). All those processes are part of the Process Controller and they communicate with one

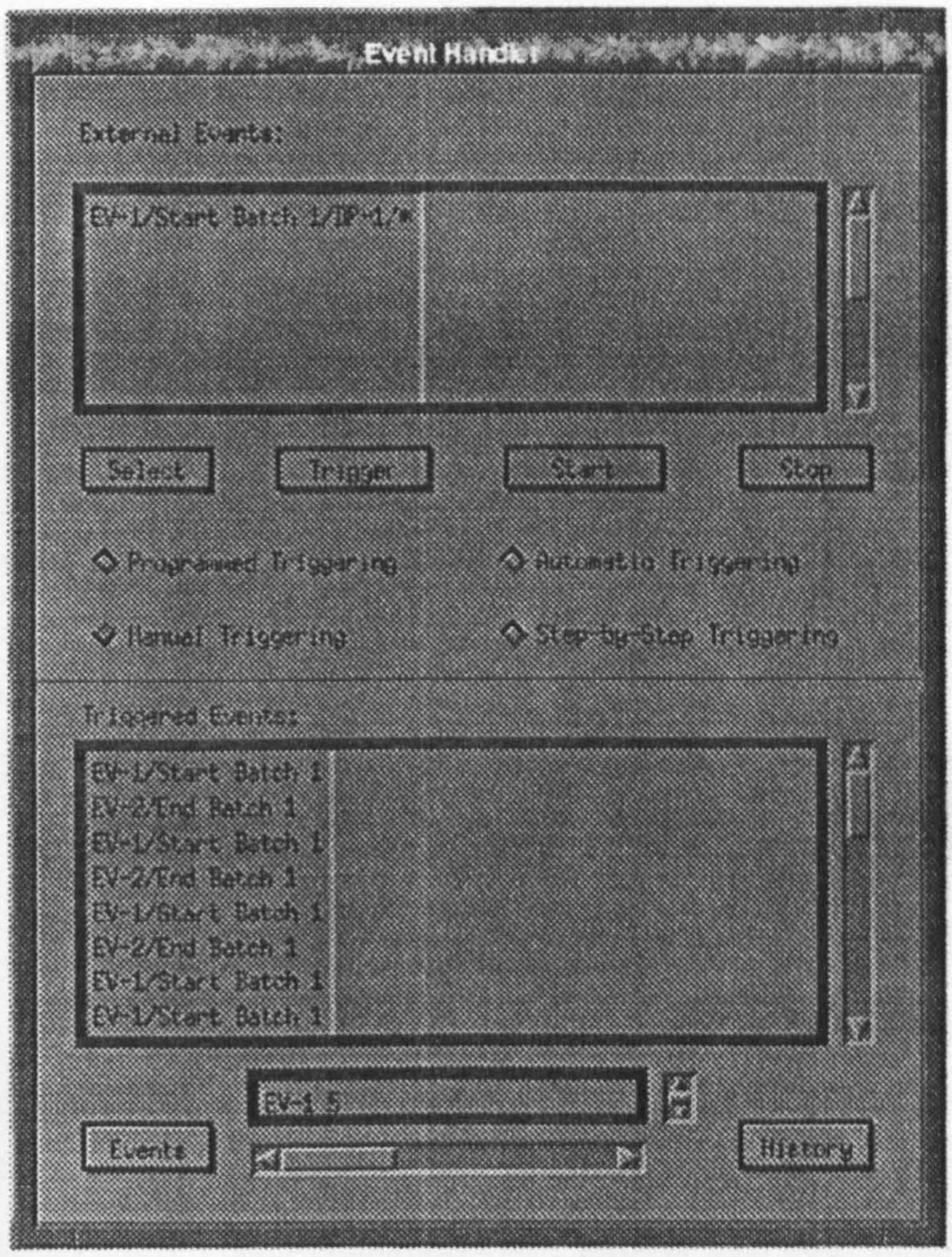


Figure 47 - Event Handler interface

another via local communication sockets (i.e. Unix sockets), in order to coherently execute several occurrences of process models.

As part of its additional debugging functions, the Process Controller interacts with the designer through the interface depicted in Figure 48. This interface enables the designer to: interrupt the process model execution while running in automatic mode; resume execution of the model; or start step-by-step execution from the interrupting point onwards. A formal specification of possible states associated with the control of debugging functions is shown in Figure 49[1].

7.1.3. Activity Controller

The Activity Controller is responsible for coordinating the execution of occurrences of enterprise activities. Like the Process Controller, the Activity Controller

---

1. In this figure, bubbles indicate states and bars crossed by arrows indicate state transitions.

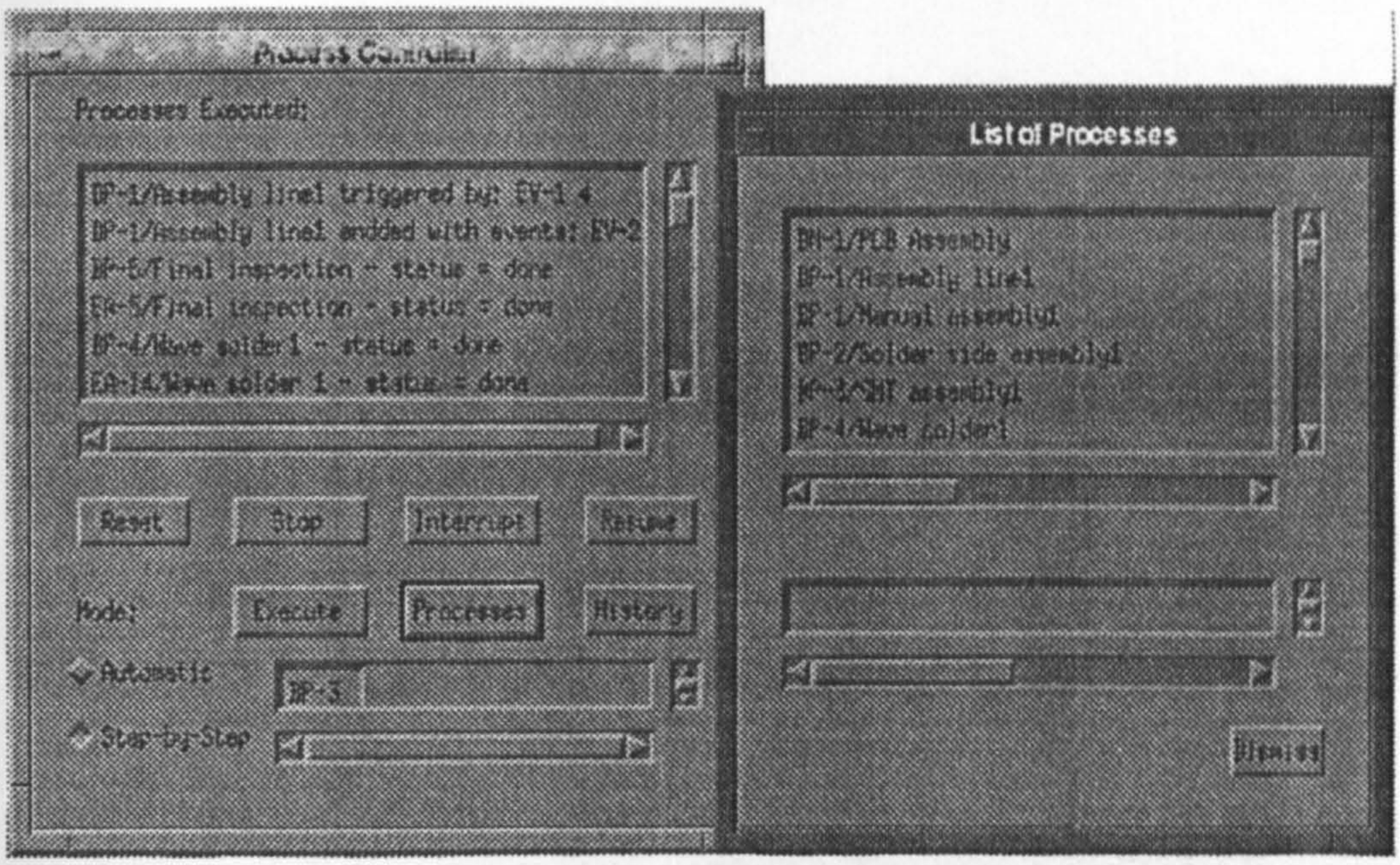


Figure 48 - Process Controller interface

can deal with the execution of several concurrent occurrences of enterprise activities. This is achieved through activating processes which represent enterprise activity occurrences and communicate via the CIM-BIOSYS infrastructure. In this way, they may share the same code structure described in activity behaviour diagrams (as depicted in Figure 31). The Activity Controller and enterprise activity occurrences communicate with one another via CIM-BIOSYS application services, this in order to coherently execute several occurrences of activity models.

As part of its additional debugging functions, the Activity Controller interacts with the designer through a similar interface to the one defined by the Process Controller (depicted in Figure 48).

7.1.4. Resource Manager

The Resource Manager is responsible for controlling the allocation of active resource components to execute threads of the business model (i.e. performs a task dispatching function). In the Resource Manager, the unfolding of a business model is constrained by the resources allocated to execute the model. However, the actual resource scheduling function is not supposed to be provided by the Resource Manager. In fact, the Resource Manager operates as a bridge amongst the three parties involved in resource allocation, namely: the remaining components of the business entity (i.e. Process Controller, Event Handler and Activity Controller); active resource components that need to be scheduled; and a special type of active resource component

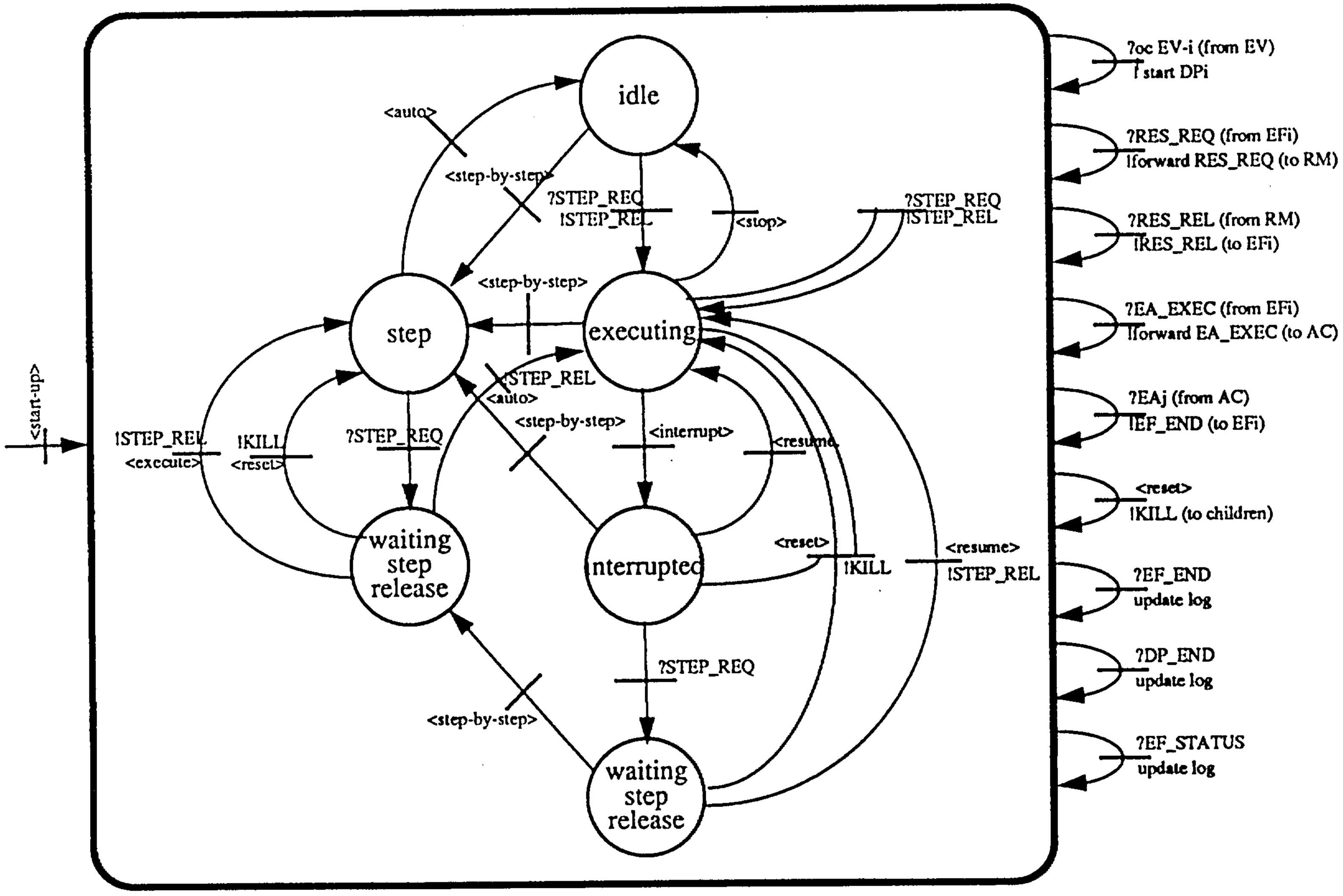


Figure 49 - State-transtion diagram for the Process Controller

(which provide the scheduling functionality). As for other components of the business entity, the Resource Manager can handle the concurrent execution of several occurrences of model threads. At the rapid-prototyping stage, the Resource Manager interacts with active resource components represented as CIM-BIOSYS applications. These applications are similar in nature to those representing occurrences of enterprise activities.

As part of its debugging facility, the Resource Manager interacts with the designer through a similar interface to the one defined by the Process Controller (depicted in Figure 48)

In addition to its four main components, the business entity also enables the user to view internal events occurring within two other classes of components, namely: enterprise activities and active resource components.

7.1.5. Enterprise Activity

Run-time occurrences of enterprise activities are mapped onto CIM-BIOSYS applications, each of which is initiated dynamically every time an enterprise activity is triggered by procedural rules (executed by the Process Controller). Achieving control over creation and interaction between processes is a function of the Activity Controller. The Activity Controller also creates a file from which each enterprise activity obtains its fragments of code (represented by **EAi** in Figure 44). This code is then interpreted by a Petri-net engine, which is constructed specifically to execute its functionality.

This code is generated from a Petri-net model (created using activity behaviour diagrams exemplified in Figure 31). The transition at the top of Figure 31 (i.e. trans_in) is triggered when the enterprise activity is created by the Activity Controller; whilst when the transition at the bottom of the diagram (i.e. trans_out) is fired, it triggers the enterprise activity termination procedure. An enterprise activity has a transitory life. It is created at run time every time its functionality is required and it lasts for as long as "trans_out" is not triggered. When "trans_out" is triggered the enterprise activity ceases to exist.

A number of occurrences of enterprise activities (that share the same description defined in the activity behaviour diagram) may exist at same time. However, they are uniquely identified by an identifier generated by the Activity Controller. Such an identifier is related to the thread of model execution in which the enterprise activity is used. For example, at a particular time, the second occurrence of the enterprise activity EA-1 will cause the creation of a CIM-BIOSYS application named EA1.2, which will be executed simultaneously with application EA1.1 (which was previously created).

An enterprise activity occurrence does not provide the standard debugging functions present in the four other components of the business entity. Its means of

interacting with the designer is through a shell tool [Back 1986] on which information associated with the firing of each transition are displayed (see Figure 50)[1]. This enables a designer to monitor the evolution of a Petri-net associated with the enterprise activity and to assess repercussions with respect to the remainder of the business model. In this way, designers can identify any consistency in a model.

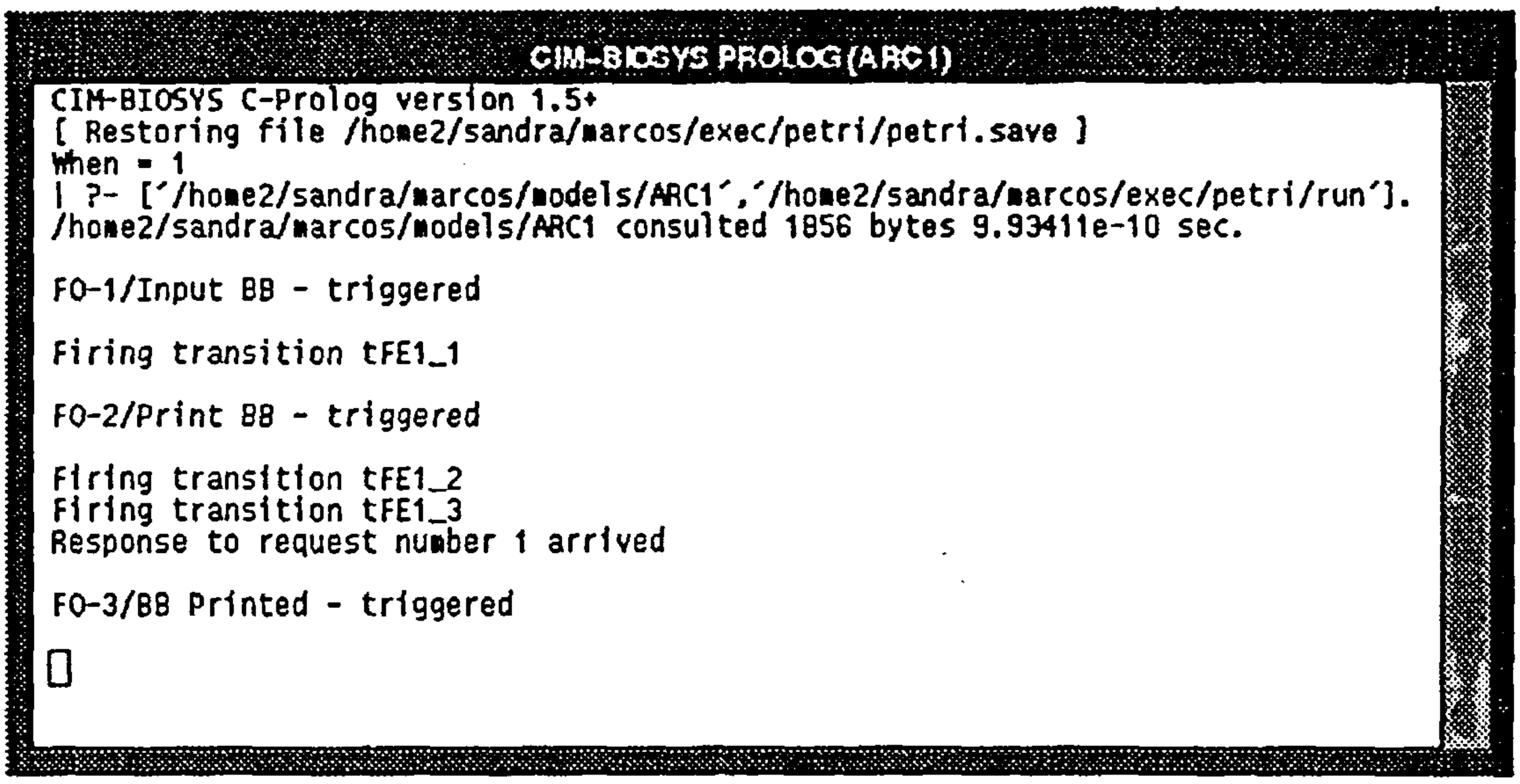


Figure 50 - Enterprise activity and active resource component interface

7.1.6. Active resource components

As previously indicated, at the design specification modelling level the behaviour of active resources components is emulated by a Petri-net model (which itself is defined by an entity behaviour diagram, as shown in Figure 32). Likewise for an enterprise activity, a model is generated by the SEW-OSA CASE tool and passed to the active resource component engine by the Resource Manager as a file (i.e. **ARCj** in Figure 44). This model is executed by the active resource component, which exists as an independent CIM-BIOSYS application, and interacts with enterprise activity occurrences through message exchange. These messages are not exchanged directly between enterprise activities and active resources, but via use of a communication protocol established between the Resource Manager (which interacts with active resource components) and the Activity Controller (which interacts with enterprise activities).

Similarly to an enterprise activity occurrence, an active resource component

---

1. This interface was built upon an interpreter developed by the University of Edinburgh [Pereira 1982].

interacts with the designer through a shell-tool similar to the one used by the enterprise activity (see Figure 50).

7.2. Business Entity Structure

For the business entity to execute a model, co-operation is required between the components discussed in the previous section. Each component provides certain services required for model execution, this by acting upon a certain view or fragment of the business model. Execution of a complete model is achieved via a communication protocol developed to use the application services of the CIM-BIOSYS infrastructure. This layer of protocol allows components to offer their services to remaining sections of the business entity, thereby facilitating cooperation through message exchange.

The type and format of the messages defined to be exchanged by the components of the business entity are listed in Table 4. These messages include interactions between the Event Handler, the Process Controller, the Activity Controller and the Resource Manager, as well as interactions between processes local to the Process Controller (i.e. occurrences of domain processes and business processes).

It is not a trivial task to gain an understanding of how the components of the business entity interact. However, this is necessary to ensure that the model correctly reflects properties of the real system. Much of the complexity involved in this process stems from the considerable number and types of messages required to enact multiple occurrences of a model.

In order to illustrate how a model is executed by the business entity, a business model example will be examined. Figure 51 illustrates a very simple business model. This model consists of a single domain process (DP-1), triggered by an event EV-1 which, on its completion, generates an event EV-2. The domain process comprises a single business process (BP-1) which, in turn, uses only one enterprise activity (EA-1). The enterprise activity uses two functional operations (FO-1 and FO-2) provided by one active resource component (ARCj). Furthermore, in this example functional operations are to be executed sequentially (i.e. FO-1 first then FO-2).

Figure 52 shows a scenario diagram of the business entity. This diagram illustrates how execution of the example model depicted in Figure 51 is achieved. The round boxes in Figure 52 represent CIM-BIOSYS applications whereas square boxes represent Unix application processes (which do not communicate via the CIM-BIOSYS application services). All these applications communicate through exchanging messages, as defined in Table 4. The messages are numbered sequentially, according to the order at which they occur (this number being displayed between parenthesis in Figure 52). The fields transmitted in the message appear in the order depicted by their labels in Figure 52[1].

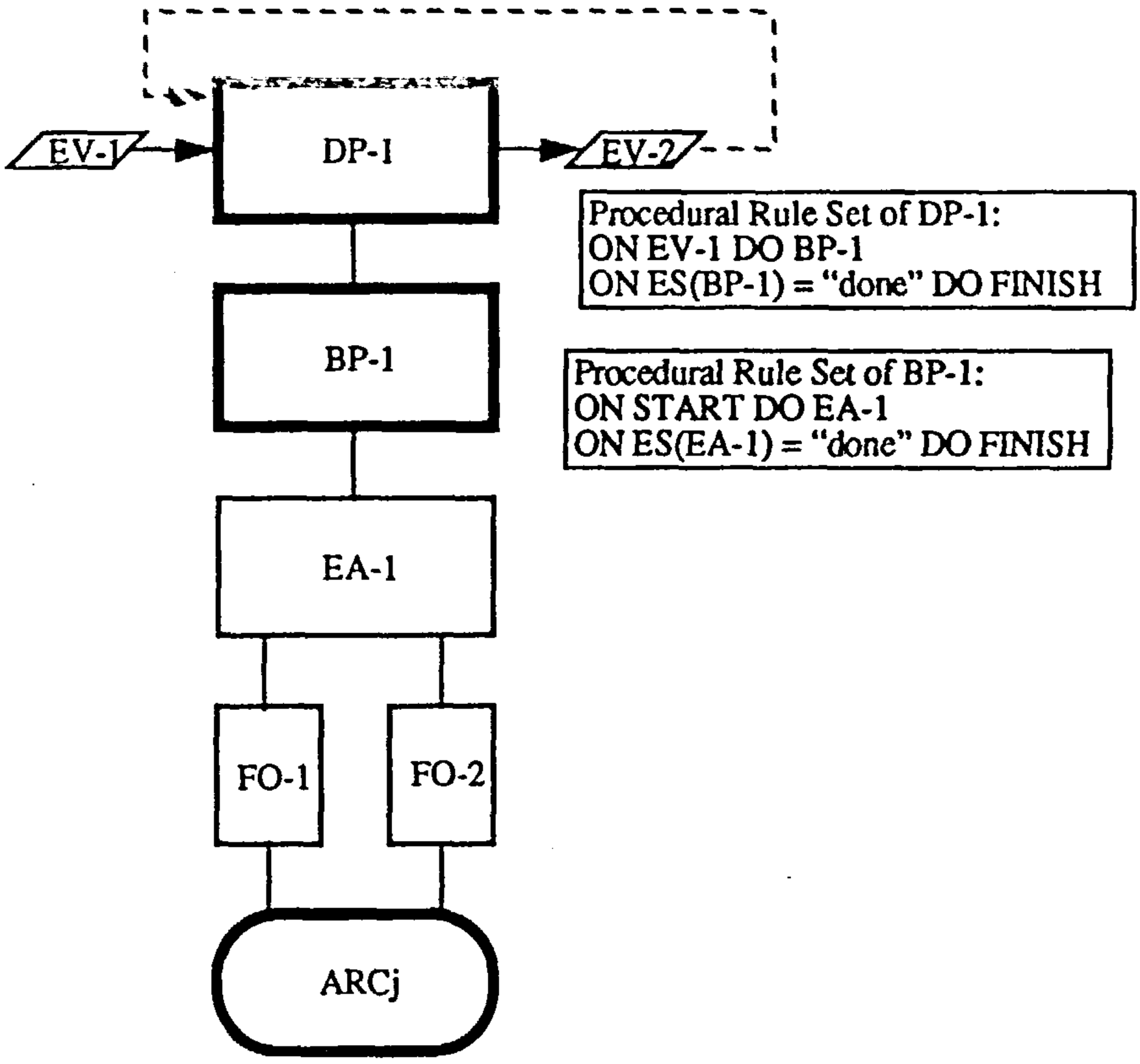


Figure 51 - Model example for describing business entity interactions

A step-by-step description of the scenario diagram shown in Figure 52 is presented in Appendix 8. This includes a description of the meaning of each message and its fields. However, the following points are worth highlighting:

- for the example of Figure 51, the execution of a thread of the business model starts when EV-1 is generated, which triggers DP-1 and, in turn, BP-1 and EA-1. EA-1 requests the execution of the operations FO-1 and FO-2, and these are executed by ARCj. On completion, EA-1 generates EV-2 which marks the end of a thread of this business model;

- the separation between functionality and behaviour is evident in Figure 52 through the interactions between ARCj and EA-1. These interactions occur via the Activity Controller and the Resource Manager, so that neither ARCj nor EA1.i are visible to one another;

- ARCj is allocated to serve EA1.i as long as EA1.i exists. That is, ARCj will not execute functional operations of an enterprise activity occurrence other than the one

---

1. Fields in square brackets are not transmitted, but implied by the context in which each message occurs.

Table 4 - Types and formats of the messages within the business entity

identifier	**message**	**format**
PID	child reporting pid number to the parent	oc <pid> PID <ef-id>
EF_STATUS	process reporting its child ending status to PC	oc <pid> EF_STATUS <ef-id> <es>
EF_END	reporting end of execution	oc <pid> EF_END <ef-id> <es>
DP_STATUS	DP reporting an event it generated to PC	oc <pid> DP_STATUS <ef-id> <ev-id1> <ev-id2>...
DP_END	DP reporting end of its execution to PC	oc <pid> DP_END <ef-id> <ev-id1> <ev-id2>...
STEP_REQ	process requesting permission to execute a step to PC	oc <pid> STEP_REQ <nef> <ef-id1> <ef-id2>...
STEP_REL	PC releasing execution of a step	oc <pid> STEP_REL <nef> <ef-id1> <ef-id2>...
RES_REQ	requesting a resource	oc <pid> RES_REQ <nef> <ef-id1> <ef-id2>...
RES_REL	PC releasing a resource	oc <pid> RES_REL <nef> <ef-id1> <ef-id2>...
EA_EXEC	requesting EA execution	oc <pid> EA_EXEC <ef-id>
FO_EXEC	request FO execution	oc <pid> (FO_EXEC) >fo-id>
KILL	process and PC killing their children	oc <pid> KILL

Legend:

oc: occurrence of a thread of execution of the business model

pid: process identification number

ef-id: enterprise function identification number

es: ending status

ev_idi: event identification number "i"

nef: number of enterprise functions to follow

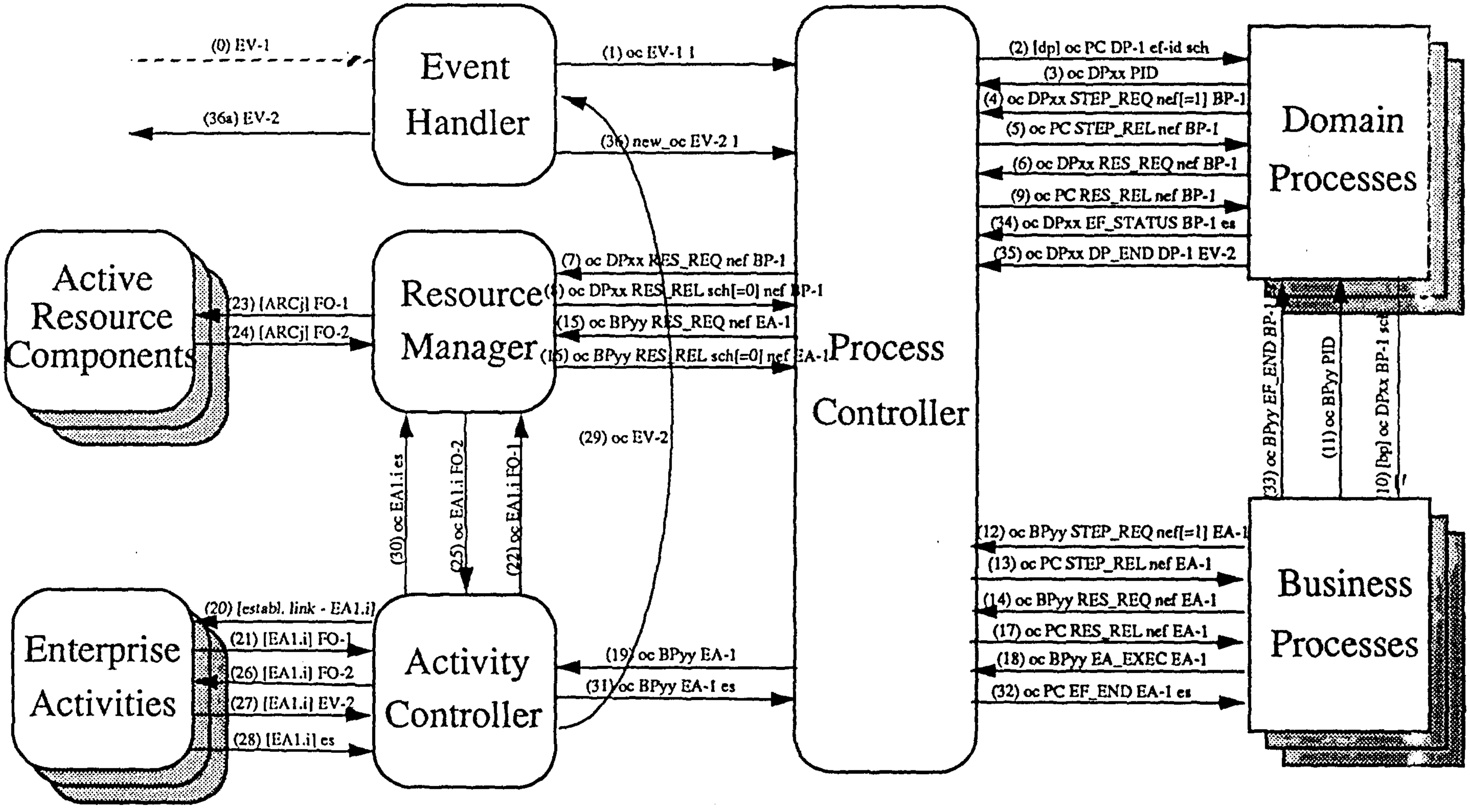


Figure 52 - Business entity of SEW-OSA

it is currently serving. This means that when an enterprise activity finishes executing its functionality, it must report its completion to the Resource Manager.

- finally, Figure 52 depicted the execution of a very simple business model which, nevertheless, makes use of all messages defined in the business entity protocol (see Table 4). More complex situations arise when the relationships between the components are based on larger and more complex models, with repercussions to the number, content and sequence of the messages exchanged amongst components. For instance, if BP-1 in Figure 51 owned another business process, an additional sequence of messages similar to the ones defined for itself would be required for the execution of procedural rules of the new business process. This business process could, in turn, own another business process and so forth.

A possible situation which is not illustrated by Figure 52 can occur when a domain process owns an enterprise activity. In this case, the same set of messages used by **bp** to execute the enterprise activity can be used by **dp**. The only difference between a domain process and a business process at run time is that domain processes are triggered by the Process Controller directly (via events) whereas business processes are triggered either by domain processes or by other business processes.

7.3. Business Entity Implementation

Essentially, the business entity functions as an interpreter of the models generated by the CASE tool. Each of its component is responsible for interpreting and executing a segment of the model (e.g. the Event Handler manipulates all the events and the Process Controller all the process descriptions).

Two classes of components were produced to enable the creation of a business entity, dependant on the way in which they would need to be executed. The two classes produced are: (1) components that are executed as CIM-BIOSYS applications (and, thus, are capable of accessing the CIM-BIOSYS application services) and (2) components that are executed directly as Unix applications (in which case interactions with other processes are via local sockets). Components included within the first class were: the Process Controller, the Activity Controller, the Resource Manager, the Event Handler, and 'run-time' occurrences of active resource components and enterprise activities. This class of component is represented by round boxes in Figure 52. 'Run-time' occurrences of domain processes and business processes comprise a second class of components. The second class of components is represented as square boxes in Figure 52. Within the first class, another subdivision exists between (1a) processes that interpret the CIM-OSA models directly (e.g. Event Handler, Process Controller, Activity Controller and Resource Manager) and (1b) those that interpret models

represented in a Petri-net form (e.g. 'run-time' occurrences of enterprise activities and active resource components).

A brief description of how each of these components was implemented is presented below. In each case, the following building blocks of functionality were used: user interface element, model interpreter, communication element, data logging element and core functionality element.

7.3.1. Class 1: CIM-BIOSYS applications

1.a. CIM-OSA Models

Components within this class share the following features:

- Their user interfaces have been built through using the BuilderXcessory interface generator (discussed in Appendix 3);
- They interpret models in a common way. That is, a model (in an structured text form) is read in by means of yacc and lex tools [Sun 1990b] and transformed into a "C" structure which is manipulated directly by the functionality of the component;
- They use a common layer of functions devised and developed to interact with CIM-BIOSYS application services (i.e. the library of services produced by I.A. Coutts [Coutts 1994]);
- They include a capability to display information about the status of the component and to record such information in a log file (as shown in Figure 44) in order to facilitate subsequent analysis;
- Much of their code has been written in "C" [Sun 1990a] and incorporates a number of library functions produced or purchased by the MSI Research Institute (e.g. libraries of list manipulation, event handling, local socket communication and the CIM-BIOSYS application process interface).

The different capabilities of particular components are realised via use of different algorithms within their core functionality. Indeed, the algorithms devised and implemented by the author for each component are illustrated by Figures 53 through 56.

1.b. Petri-Net Models

This class of components is very similar to those of (1.a), except that a greater (than class (1.a)) proportion of their functionality is defined by the business models. Their core functionality demonstrates the following properties:

- Waiting for an event to occur (i.e. a button pressed by a designer or an incoming message from the Activity Controller) - see Figure 52;
  - When an event is generated, the Event Handler checks whether the event should trigger a domain process;
    - If so, the Event Handler generates a business model occurrence and an event identifier, and sends the event identifier to the Process Controller;
    - If not, the Event Handler passes the event to the external world (i.e. to non-CIM-OSA-compliant domains - Figure 23);
- Handling any request associated with the display of information to the designer via its appropriate interface (see Figure 47).

Figure 53 - Algorithm of the Event Handler

- Waiting for an asynchronous occurrence (e.g. incoming message from CIM-BIOSYS applications or incoming message from **dp** and/or **bp**);
  - When an event is received from the Event Handler, the Process Controller searches for domain processes that are triggered by the event and spawn them;
  - When a STEP_REQ is received from a dp/bp, the Process Controller sends back an STEP_REL if it is in 'idle state' or 'execution state' or when commanded by the designer (i.e. pressing of one of the following buttons, namely: <execute>, <auto> or <resume>, as shown in Figure 48, according to Process Controller internal state);
  - When a RES_REQ is received from dp/bp, the Process Controller forwards it to the Resource Manager;
  - When a RES_REL is received from the Resource Manager, the Process Controller forwards it to the appropriate occurrence of dp/bp;
  - When an EA_EXEC is received from dp/bp, the Process Controller forwards it to the Activity Controller;
  - When an EF_END is received from AC, the Process Controller forwards it to dp/bp;
  - When an EF_STATUS, EF_END or DP_END is received from dp, the Process Controller updates appropriate the information in its interface;
- Handling any request associated with the display of information to the designer via its appropriate interface (see Figure 48).

Figure 54 - Algorithm of the Process Controller

- Waiting for an asynchronous occurrence (e.g. incoming message from CIM-BIOSYS applications);
  - When an EA_EXEC is received, the Activity Controller spawns the corresponding enterprise activity occurrence;
  - When a request to execute a functional operation is received from the enterprise activity occurrence, the Activity Controller forwards it to the Resource Manager;
  - When a report of execution of a functional operation is received from the Resource Manager, the Activity Controller forwards it to the appropriate occurrence of the enterprise activity;
  - When an event is received from the enterprise activity, the Activity Controller forwards it to the Event Handler;
  - When an ending status is received from the enterprise activity, the Activity Controller updates its internal variables and forwards it to the Resource Manager and the Process Controller;
- Handling any request associated with the display of information to the designer via its appropriate interface.

Figure 55 - Algorithm of the Activity Controller

- Waiting for an asynchronous occurrence (e.g. incoming message from CIM-BIOSYS applications);
  - When an RES_REQ is received from the Process Controller, the Resource Manager checks the pertaining scheduling information and sends a RES_REL back to PC;
  - When a request to execute a functional operation is received from the Activity Controller, the Resource Manager checks the status and capability of active resource components and forwards the functional operation to an ARC capable of executing it;
  - When a functional operation is received from an active resource component, the Resource Manager forwards it to the Activity Controller;
  - When an ending status is received from the Activity Controller, the Resource Manager updates its internal variables and releases the appropriate active resource component;
- Handling any request associated with the display of information to the designer via its appropriate interface.

Figure 56 - Algorithm of the Resource Manager

- Their user interfaces are very simple, consisting basically of a shell-tool [Back 1986] with information reported to the designer on scrolled windows (as depicted by Figure 50);
- They use the same means of interpreting a model as that for (1.a) components. Hence, a model stored in a file identified by a name derived from the component's name is read in and continually scanned by a Petri-Net engine;
- They use the same layer of functions as (1.a) components to interact with the application services of CIM-BIOSYS;
- They can display information about their internal status, although they do not include a capability to record such information in log files. This limitation is inherent to the Petri-net engine adopted;
- They have been programmed by implementing a surface syntax [Aguiar 1994b] in Prolog, in order to develop a predicate-action Petri-net engine, whose usage is described in the following.

Figure 57 shows an example of the type of syntax that these components support. In this Petri-net description, a variable is assigned to every place, transition or actual internal variable manipulated in the predicates and actions of the net. Basically, two types of predicates are acceptable: **variable()** and **transition()**.

variable() is used to initialise variables and define the initial marking of the Petri-net. The first argument of this predicate is the variable name and the second is its value.

```
Activities belonging to: DM-6
Behaviour of: EA-1 {

variable(ptrans_in, 1).
variable(pEA1_13, 0).
variable(pEA1_14, 0).
variable(pEA1_15, 0).
variable(pEA1_16, 0).

transition(trans_in,(ptrans_in = 1),(pEA1_13 is pEA1_13 + 1 @ ptrans_in is ptrans_in - 1)).
transition(tEA1_7,(pEA1_13 = 1),(pEA1_14 is pEA1_14 + 1 @ pEA1_13 is pEA1_13 - 1 @
send_app(ac, "FO-1"))).
transition(tEA1_8,(pEA1_14 = 1),(pEA1_15 is pEA1_15 + 1 @ pEA1_14 is pEA1_14 - 1 @
send_app(ac, "FO-2"))).
transition(tEA1_9,(pEA1_15 = 1 & recv_app(ac, "FO-3")),(pEA1_16 is pEA1_16 + 1 @
pEA1_15 is pEA1_15 - 1)).
transition(trans_out,(pEA1_16 = 1),(pEA1_16 is pEA1_16 - 1 @ send_app(ac,"done") @ halt)).
}
```

Figure 57 - Example of a Petri-net model processable by the Prolog interpreter

transition() defines conditions that must be met for the transition to be enabled, as well as actions that will be taken when the transition is fired. This predicate has three arguments. The first is the transition name or identifier. The second is a list of enabling conditions for the transition, this includes the number of tokens that must be present in the input places of the transition. The third argument is a list of actions that will be executed when the transition is fired.

Predicates and actions are either associated with internal and external happenings. An internal happening consists of any operation on variables (e.g. comparisons, arithmetic operations, updates, etc.), including the progressive updates of Petri-net markings. An external happening is any operation that involves interactions with other components of the business entity. For instance, in Figure 57, the transition trans_EA1_9 requires a message to be received from the Activity Controller (i.e. ac) with the content set to FO-3 in order for it to be enabled. Likewise, the triggering of transition trans_EA1_7 activates the sending of the message FO-1 to the Activity Controller.

The surface syntax consists of a set of pre-fabricated predicates which are used to build the predicates variable() and transition(). Such a set was built using Prolog to manipulate facts (i.e. elements of knowledge in a Prolog knowledge base), and "C" to implement the functions which enable the component to interact with CIM-BIOSYS. Further information on how such a surface syntax was created is presented by Aguiar and Coutts [Aguiar 1994b].

7.3.2. Class 2: non-CIM-BIOSYS applications

The components of this class share the following features:

- They do not possess a user interface. Their user interface is indirectly provided by the Process Controller which filters the relevant information that they provide;
- They use the same means of model interpretation as Class 1.a components. Hence, the model is generated in a structured text form, parsed by yacc and lex tools and transformed into a "C" structure which is manipulated directly by the functionality of the component;
- They do not interact with CIM-BIOSYS. Unix sockets are their basic means of communicating with their child processes (i.e. processes that have been spawn by them) and with the Process Controller, which is the only CIM-BIOSYS process that interacts with them;
- They do not possess the ability of either displaying information about what is occurring within the component or recording such information. Again, this function is indirectly performed by the Process Controller based on status reports that it

receives from these components;

- They have been programmed purely in "C" and incorporate some of the library functions used by the CIM-BIOSYS applications (e.g. libraries of list manipulation, event handling and local socket communication).

The elements of this class of components practically implement the same algorithm, differing only in the way their processes interact. That is, these elements can either represent domain processes or business processes. Their differences arise from the inherent particularities of these two processes.

Once these processes have been spawned by either the Process Controller or a parent process, the algorithm that they execute is basically related to the execution of the procedural rule set of its functions. That is, one component is required for the execution of each procedural rule set, where each rule follows the format: "ON (condition) DO (functions)". The algorithm for such component consists basically of the steps shown in Figure 58.

Implementing these components was perhaps one of the most complex activities involved in the SEW-OSA implementation. Even for very simple models (as the one illustrated in Figure 52), a number of occurrences of domain processes and business processes may be initiated which compete for system resources. The model execution process then involves the exchange of a considerable volume of messages among these occurrences and the Process Controller, in such a way that debugging their algorithms becomes very complex.

A common feature of all components of the business entity is their maintenance of state variables associated with every interaction that they execute in association with their peer components. That is, to every identifier or message exchanged between the components, variables identifying the message are stored in internal state tables, so that when a response to a certain command or request arrives, the component has a record of what caused the request in the first place. Hence, it can determine what to do next. These occurrences generate a considerable volume of data which in most cases must be dealt with as separate threads of business model execution.

7.4. Rationale for the Approach Adopted to Realise the Business Entity

This chapter described the approach taken to realise the business entity, as well as an explanation of how it works. Following is a brief discussion of considerations which led to the adoption of this approach.

As explained in Section 4.8, the rationale for the development of the business

i. to find the first rule (i.e. the one identified by the condition START or the one triggered by the events indicated by the Process Controller) in the procedural rule set (i.e. the enabled rule);

ii. to send a STEP_REQ to the Process Controller when the enabled rule is found, in order to request permission from the Process Controller to execute the functions listed in the rule (see Figure 45);

iii. to send a RES_REQ to the Process Controller, when a STEP_REL is received from it;

iv. to spawn processes in the list of functions and/or request the execution of activities to the Process Controller (EA_EXEC), when a RES_REL is received from the Process Controller;

v. to indicate the completion of a function to the designer via a appropriate interface, by storing its identifier on a list of completed functions and test the list against the conditions of every procedural rule ("condition" part of the procedural rule), when an EF_END is received from either the Process Controller (this relating to execution of enterprise activities) or a child process (this relating to execution of business processes);

vi. to enable a procedural rule, when its condition matches the ending statuses of some of or all the functions on the list of completed functions; to go back to step "(ii)" and carry on executing the algorithm, If the list of functions to be triggered is not equal to FINISH or TERMINATE; otherwise, to finish executing procedural rules, find its ending status, report it back to its parent or Process Controller, and terminate.

Figure 58 - Algorithm of a domain process and business process

entity is that software was not yet available in the market to realise its purpose. In the context of its development, a number of design decisions were made which could have led to the selection of alternative solutions. The most relevant decisions include:

a. Business entity structure

The business entity was constructed with reference to a general specification of an integrating infrastructure, as provided by CIM-OSA [ESPRIT/AMICE 1993a]. This specification was limited to defining (at a very high level of abstraction) the interfaces between the entities of the integrating infrastructure, as well as the behaviour of some of these entities. With respect to the business entity, this specification (1) identified the need for a Process Controller, an Activity Controller and a Resource Manager; and (2) broadly suggested what they should do, without detailing either their functionality or how they should be implemented.

As these definitions were not sufficiently detailed to realise a working business entity, this research had to extend the specification to yield the solutions discussed in this chapter. Prime additions to the specification emerging from this research involved the detailed design of the business entity, based on an implementation of the two classes of components previously defined.

Pre-requisite properties which shaped the design of component classes in the current version of the business entity were:

- **class 1**: components that can be easily distributed across the CIM-BIOSYS infrastructure (as CIM-BIOSYS applications), where: (1) **class 1.a** describes components which are: permanent constituents in the business entity (i.e. they always exist, regardless of the system configuration) and have a fixed number of occurrences; and (2) **class 1.b** describes enterprise activity occurrences and active resource components which are not permanent constituents nor do they occur on a pre-defined and fixed number of times;
- **class 2**: components that relate uniquely to the Process Controller and therefore can be realised as spawned local Unix processes.

The motivation for implementing these classes of component was the need to rapidly realise a working solution for the business entity. In principle, if debugging functions of the business entity components are not implemented and various instances of permanent components are generated at run-time for each computer host in the system, the business entity can be implemented completely using only components of *class 1*. In this case, processes that represent occurrences of domain processes and business processes are also realised as CIM-BIOSYS applications, facilitating their distribution across the integrating infrastructure.

Finally the use of *interpreted code* (as opposed to *compiled code*) was adopted to facilitate the debugging of the first prototype of the business entity. Discussions within the "Model-Driven CIM" project are leading to future versions of the business entity being OMG compliant [OMG 1991] and based on compiled code.

b. Development tools

As previously discussed, development of the business entity was realised using programming utilities commonly used in the MSI Research Institute (e.g. X-Windows Motif [Heller 1991], BuilderXcessory [ICS 1991a] [ICS 1991b], Unix programming tools [Back 1986], etc.). This context also determined the choice of programming languages (i.e "C" and "Prolog").

Alternative approaches which could have been used include use of a formal description language to specify business entity protocols and object-oriented utilities to realise it. However, their use was ruled out by their unavailability and the author's lack

of experience with them.

7.5. Limitations

Constructing the business entity of SEW-OSA was quite a complex process and required a considerable implementation effort. Even then, the implementation realised is only a sub-set of the business entity envisaged by CIM-OSA. In this respect, the resulting business entity presents certain limitations, namely:

a. Interruptions

CIM-OSA specifies that the execution of an individual thread of the business model should be allowed to be interrupted either by a human functional entity or through the conclusion of the procedural rule set of a business process with a TERMINATE event.

In the current version of SEW-OSA, the only interruptions allowed are ones generated manually by a designer through the interfaces provided by each of the components of the business entity. These interruptions stop execution of all threads of the business model (being dealt with in that component). Each interruption is generated locally and separately in each component. This implies that aspects of the model that are dealt with in each of the business entity components cannot be interrupted at the same time.

Although, the type of interruption implemented in SEW-OSA was not specified in CIM-OSA, interruptions prescribed by CIM-OSA can only be emulated in SEW-OSA by incorporating points of interruption in the models (i.e. adding alternative ending statuses to the processes involved).

b. Secure communication

The level of communication security assumed by the protocol of the business entity is the one provided by CIM-BIOSYS (i.e. no acknowledgement protocol at the application level is assumed [Gilders 1991a] [Gilders 1991b]). Additionally, the layer of services provided by the SEW-OSA business entity is not secure against losses of messages exchanged between its components (such as in the scenario illustrated in Figure 52).

Such a limitation can cause dead-locks in the business entity, which could be overcome by incorporating a secure message passing protocol into CIM-BIOSYS at the application level. Efforts in this direction are under way [Coutts 1994].

c. Ending-status

Ending-statuses are used by the business model to decide what the next process or activity must be triggered based on the ending-status of previous processes or

activities. Ending statuses of activities are determined by their behaviour diagram execution. Ending-statuses of processes should be obtained as a logical combination of the ending-statuses of other processes and activities that they use.

In the current implementation of the business entity, only two ending-statuses are allowed for processes: "done" and "any", whereas any status is allowed in an enterprise activity. Therefore, coordination decisions made on the basis of ending-statuses must be concentrated within enterprise activities.

d. Interface with information view

At the rapid-prototyping stage, the decision points in the business model are purely based on the flow of execution within the function and resource views. Decisions based on the value of an information element stored in a data-base or manipulated by a functional operation are not included at the rapid-prototyping stage. These decisions can only be considered at the system execution stage (discussed later in the thesis).

e. Communication between enterprise activities

Enterprise activities occurrences may communicate with one another at 'run-time', for the sake of synchronisation or data exchange. The current implementation of the business entity does not support this form of cross communication. However, means of including such a facility exists which, nonetheless, would have to be constrained to communications between enterprise activities belonging to the same thread of the business model.

In summary, fully implementing the complete functionality of the business entity, which is the most complex entity in the CIM-OSA integrating infrastructure, requires a great deal more effort and resources than this research project possessed. However, experiencing the complexity of such implementation was extremely useful in understanding the limitations involved in building tools to realise model-enactment.

7.6. Concluding Remarks and Contributions

The key contribution of the business entity of SEW-OSA is the provision of an instance of a model interpreter that enables the transformation of business models into service transactions executed by the CIM-BIOSYS integrating infrastructure. In other words, the implemented business entity is a key component in enabling model-enactment.

In addition, debugging facilities associated with the business entity play an essential role in checking the business model for any errors not identified at the modelling, analysis and simulation stages. The business entity associated with the SEW-OSA CASE tool also provides the first instance of an integrated enterprise

engineering environment [ESPRIT/AMICE 1993a] which actually works upon an industrially proved integrating infrastructure (i.e. the CIM-BIOSYS infrastructure).

It is a major thrust of this research that these two elements (i.e. the SEW-OSA CASE tool and the business entity) establish a stable basis for incorporating services of the remaining entities of CIM-OSA into the CIM-BIOSYS infrastructure. The outcome of such an incorporation is the realisation of a complete engineering environment for enterprise integration.

Chapter 8 - Case Study on the Application of SEW-OSA

This chapter brings together the SEW-OSA capabilities discussed in previous chapters and applies them in a case study of an industrial system. Hence the case study involved modelling, analysis, simulation, rapid-prototyping. configuration and execution with focus on a specific shop-floor system at D2D[1]. The case study embraced:

a. **Case study activities**, these included:

- an overall description of all activities performed within the case study boundary;
- a description of the 'as-is' situation and issues identified for improvement;
- a proposal for tackling integration problems and coordination issues between the resources involved in D2D shop-floor activities, this as a means of improving performance;
- a description of the model resultant from the application of SEW-OSA;
- the specification of a 'should-be' system based on the combined application of CIM-BIOSYS and SEW-OSA, this in order to realise integration and coordination[2].

b. **Relationship between the case study system and other D2D systems of concern.** This included a preliminary analysis of how a SEW-OSA-based shop-floor control system could be integrated with information systems in current use at D2D, with particular emphasis on an ongoing company project under the banner of SPEAR (Strategic Planning Environment for Assembly Routes). SPEAR is an initiative aimed to develop systems to support automated process planning.

Case study activities (described in this chapter) focus on the application of SEW-OSA. The relationship between the case study system and other systems at D2D is described in some detail in Appendix 9. This appendix is included to illustrate the context in which the case study was carried out.

---

1. Design to Distribution (former ICL, International Computers Ltd.), a major UK computer manufacturer. Prime focus of the case study work was on their printed circuit board manufacturing plant.
2. Quantitative results associated with the enactment of 'should-be' models are discussed in Chapter 11.

8.1. Context of the Case Study: Domain Definition

An evaluation of the applicability of SEW-OSA was envisaged with respect to three classes of domain within D2D. They would serve as the test-bed for much of the work of this thesis and, indeed, for the work of the "Model-Driven CIM" project.

a. Model based integration of the shop-floor:

This initiative tackles integration issues related to a manufacturing shop-floor. In the case study, models of how the shop-floor is currently organised (i.e. 'as-is' situation) within the target company were produced, in association with on-going modelling initiatives aimed at improving the current situation (i.e. identification of possible 'to-be' scenarios).

b. Software inter-operability:

This initiative sought to address problems faced by D2D when integrating heterogeneous software packages, focusing on the means of realising structured interaction between threads of functionality embedded within these software packages. The software packages involved are used as an integral part of the "product introduction business process" of D2D.

c. Business analysis:

This initiative sought to investigate means of comparing alternative strategies by which improvements in enterprise performance could be realised, this within the context of helping to develop an IT strategy for D2D.

As part of this research, early attempts were made to apply CIM-OSA to certain modelling domains, this prior to the realisation of SEW-OSA[1]. The final application domain was defined having conducted this early work and identifying the following requirements:

- a domain which would facilitate experimentation over different phases of the IMS life cycle, through system modelling to system operation;
- a well understood domain which would enable the research to concentrate on evaluating architectural issues, rather than investigating re-engineering issues connected with the domain;
- addressing an area of interest to D2D, as important resource inputs would be

1. Such an early study embraced a modelling and simulation exercise documented in three articles [Aguiar 1992e] [Aguiar 1993a] [Aguiar 1993b] and three internal reports [Aguiar 1992a] [Aguiar 1992b] [Aguiar 1992c].

required from their personnel in order to obtain case study data.

Although prime focus was defined to be the integration of the shop-floor domain (item "a" above), in due course, SEW-OSA can also provide support to address the two other fields of application (i.e. software interoperability and business analysis). Indeed, the author will take this opportunity in his forthcoming research.

8.2. Case study activities[1]

In December 1991, an effort was initiated to gather and organise data associated with the PCB assembly shop-floor. Initial focus of attention was on SMT assembly lines, and with particular emphasis on assembly lines 2 and 3 as (at that time) they represented typical SMT processes. This effort required a number of interviews with D2D personnel knowledgeable about the shop-floor processes. However, due to their limited availability, initial case study activities which required their direct involvement had to be constrained to around forty man-hours. Forced to cope with such limitations and in view of the fact that the case study was instrumental rather than central to the research objectives, the following strategy formed:

a. Detailed study of available documents provided by D2D about their manufacturing processes:

Before time was spent attempting to understand the electronic manufacturing processes more generally, a thorough study was undertaken of documents describing D2D processes [BSL1991] [Roberts 1991] [Longman 1990] [ICL 1988] and SMT technology in general [Pawling 1987] [Siemens 1987] [Hinch 1988].

b. Preparation of documents to structure contacts with D2D personnel:

These documents aimed to structure interviewing processes, so that best use could be made of the available time of D2D personnel. These documents provided:

- templates identifying key data items (see [Aguiar 1992b] for further detail). These templates were particularly useful during the first interviews in clarifying the type of data required for model-building.
- tables to be filled in by D2D personnel. These tables are organised in a top-down manner and list relevant data. At the top level, data about the whole shop-floor is organised, thereby identifying its structural organisation (into lines or segments of lines). Then, a table for each assembly line (or segment of line) provides more

---

1. The activities described in this section were performed in a joint effort which involved I. S. Murgatroyd in the gathering and analysis of data obtained of D2D shop-floor.

information about its processes. Finally, more detailed data about each process is organised into separate tables. Sets of these tables were distributed in a form which could be filled out by people in different areas of D2D. Subsequently these were compared to identify inconsistencies.

- questionnaires used during interviews, this in order to gather data that could not be captured in templates or tables which was, nevertheless, relevant to gaining an understanding of certain activities on the shop-floor.

c. Actual data gathering through interviews with engineering and production personnel, and tours through the shop-floor:

About seven visits were made to the D2D plant, during which the documents prepared in previous phases were used intensively. The majority of data gathered was through use of these documents. Additionally, samples of the order book and the SPEAR database were obtained, which provided data about manufacturing processes for each PCB type (such as: sequence of operations, time and motion data, etc.). This data was essential for simulation studies.

d. Processing of the data gathered through modelling, analysis and simulation:

The data gathered was formalised in models for a twofold purpose. On the one hand, these models were used to visualise static aspects of the system. On the other hand, simulation was performed in order to study system behaviour. At the early stages of the research, SEW-OSA was not available to create and enact models. Thus, an IDEF0 representation of the assembly line was developed and enacted by translating it manually to a Petri-net model. The tools used for such an exercise were Design/IDEF [Meta 1990] for model-building and ARP [LCMI 1989] and for model-enactment. This exercise and the results generated with it are discussed in a separate set of documents [Aguiar 1992e] [Aguiar 1993a] [Aguiar 1993b] [Aguiar 1992a] [Aguiar 1992b] [Aguiar 1992c]. These documents present samples of IDEF0 models; an equivalent model of a segment of the SMT assembly line using the CIM-OSA functional syntax; and a translation of such a syntax to GSTPN. In this early study, these models were used as a means of illustrating the usefulness of model-enactment (i.e. simulation) geared towards identifying means of improving certain aspects of the assembly line.

e. Checking the accuracy of the models built based on the data collected, this through feedback interviews with D2D personnel:

These activities enabled a review of the data gathered from D2D personnel, as well as provided valuable experience with difficulties in communicating the information manipulated by the modelling tools to the average engineer on the shop-floor. Thus, in some cases, in order to obtain any feedback on the accuracy of the models, a

simplification of the models created had to be made, so that the engineers could understand them and participate in the validation process.

f. **Populating SEW-OSA with part of the data gathered at D2D:**
After the functionality of SEW-OSA presented in previous chapters was made available, the SEW-OSA CASE tool was populated with part of the case study data. The development of SEW-OSA took place in parallel with the work described in previous phases. Therefore, model-building in SEW-OSA could only start when the data gathering phase of the case study was already completed. Although, this parallelism of activities limited the kind of models that could be produced in the early stages of the study, it provided a great deal of information about requirements to be addressed, particularly as a tool for modelling and simulation.

g. **Analysis of the results obtained with the creation and enactment of the models:**
This phase sought to attempt to re-engineer part of the D2D shop-floor, so that it could operate in a model-driven manner (i.e. to realise the design of a 'to-be' system), in order to overcome the limitations encountered in the current shop-floor configuration (i.e. the 'as-is' system). The aim of this phase was to evaluate the way in which the shop-floor was operating and to specify necessary changes in order to improve its performance. It was understood that the level of detail of such a specification should be sufficient to outline the architecture of the proposed ('to-be') system.

Details of the activities and findings of phases "f" and "g" (which constitute the essence of the case study) are discussed as follows, in terms of: an overview of D2D shop-floor, aspects in need of improvement, SEW-OSA models produced and an outline of a 'should-be' system.

8.3. Overview of the PCB Assembly Shop-Floor

The D2D manufacturing cycle is comprised of three stages: (1) Printed Circuit Board (PCB) manufacturing, (2) PCB assembly and (3) final product (computer) assembly. Resources to perform the first two of these stages are located at Kidsgrove. The latter stage takes place at Ashton. The case study concentrates on activities and processes involved in the second stage.

8.3.1. The complete shop-floor

When the case study started, the PCB assembly shop-floor comprised of seven production lines composed of manufacturing devices, such as placement machines, printers, reflow-solder, etc., these operated and supervised by some fifty people.

Figure 58 depicts a simplified view of the PCB assembly shop. This highlights possible flows of PCB types through line segments required to produce them. In terms of the domain under consideration, a line segment comprises a set of manufacturing process whose basic operations do not change drastically from one PCB type to another. Line segments may also encompass a number of stages in the manufacturing cycle (such as, SMT assembly, manual assembly, track side assembly, test, etc., as shown in Figure 58).

Figure 59 shows an example of a SMT line segment in which a printer, a placement machine, reflow solder and wash-off machines are physically inter-linked by a conveyor and buffers. This means that normally when a job is allocated to the printer, it is also automatically allocated to the remaining downstream operations of that particular assembly line. Hence, no major diversion of flow occurs within a line segment, and this fact can be implied for scheduling purposes.

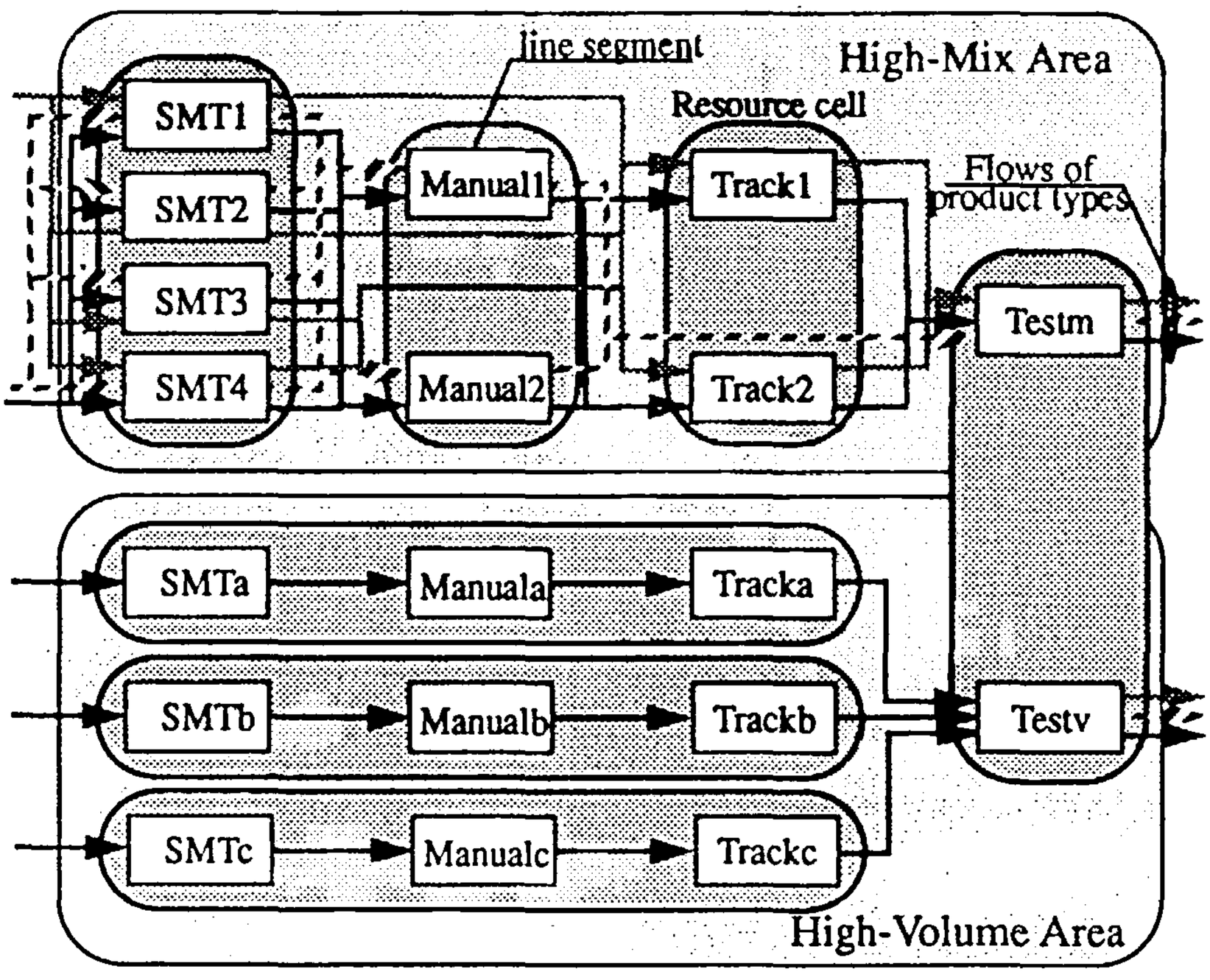


Figure 58 - Overview of the printed circuit board assembly shop-floor

A number of possibilities exist in regard to material flow amongst line segments, according to the type of PCB that need to be produced (as illustrated in Figure 58), namely:

- single-sided PCB's may require only one pass through the SMT stage;
- certain double-sided PCB's may require one pass through the SMT stage and

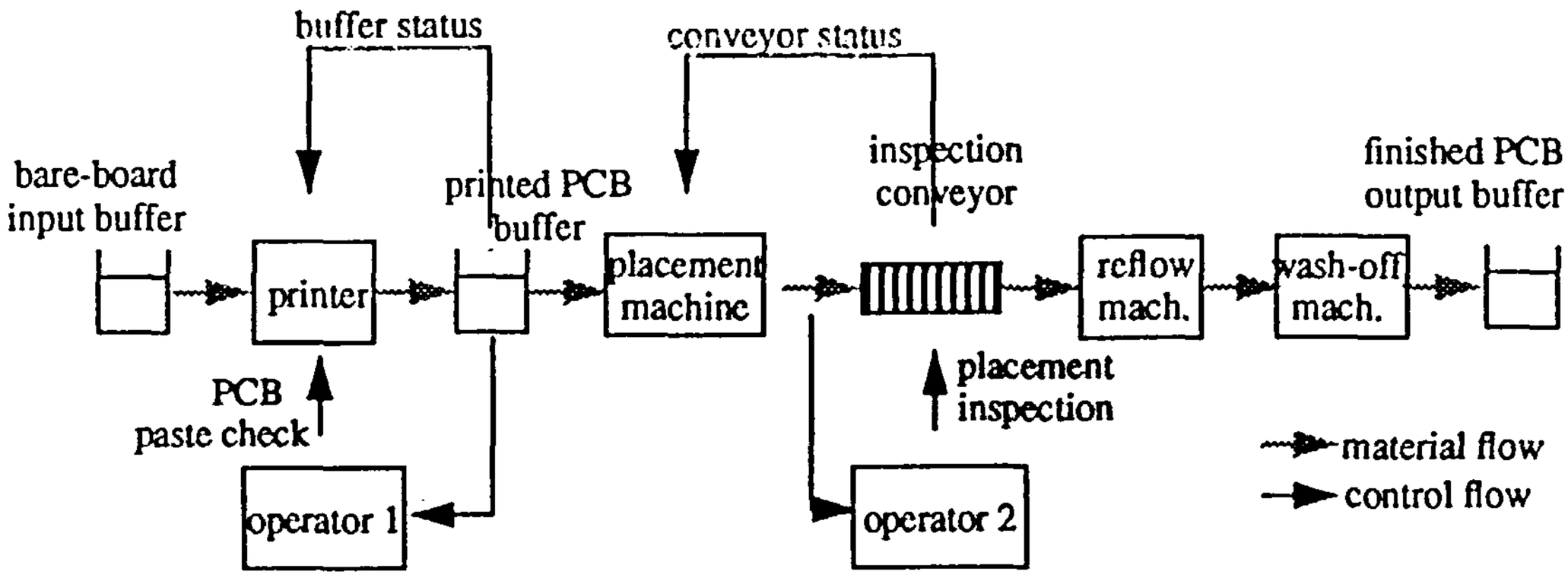


Figure 59 - Surface mount technology assembly line

another through the track side assembly;

- other double-sided PCB's may require two passes through the SMT stage and one through the track-side assembly. Track-sided components usually require use of to wave-solder technology, specially if through-hole components are used on the board;
- the manufacture of most PCB's requires some form of manual assembly stage between SMT and track-side assembly stages. This requires the configuration of two sub-types of the three main types of flows (i.e. with and without an intermediate manual assembly stage);
- all PCB's must go through a test stage.

Each line segment needs to provide distinct capabilities. For example, certain types of PCB may require the capability provided by the line segment "SMT1" (see Figure 58) or may require the capabilities provided by a certain combination of line segments (e.g. SMT1, then Manual1, then Track1 and then Testm, as indicated by one of the black flows of material in Figure 58). When allocating a job to a certain combination of line segments, these requirements need to be considered during the process planning stage, and should reflect: (1) the use of recommended process routes, this to assemble certain PCB types with minimum cycle times and set-up times[1]; and (2) the use of a recommended sequence of jobs to minimise change-over times.

In principle, many different flows are possible between the line segments indicated in Figure 58. The range of choices of a line segment to serve a particular flow (at a certain stage in the manufacturing cycle on the shop-floor) is represented in Figure 58 by cells or lines (such as, the SMT cell and assembly line "a"). In the case of

1. One of the aims of SPEAR is to generate these process routes.

lines, if the use of either assembly line "a", "b" or "c" is required to produce a given PCB batch, the complete assembly line must be allocated to assemble that batch. In that case, physical constraints apply (e.g. assembly line "a" is a fully conveyorised and dedicated to assemble only PCB's for Sun Workstation).

In the case of cells, line segments (within them) can be selected, according to the technical considerations (1) and (2) above, as well as considering scheduling constraints. For instance, a PCB assembled by the line segment "SMT1", which also requires manual assembly, could pass through one of the two line segments available within the manual cell (represented in Figure 58). A decision concerning the allocation of a particular line segment (within the manual cell) is currently the responsibility of a shop-floor supervisor.

Another important feature of D2D shop-floor is an intended separation of lines dedicated to produce boards in different batch sizes, that is (see Figure 58): (1) at high-volume with a reduced product mix, and (2) at low-volume with a high product mix. This separation is useful as certain problems faced by each of these classifications are distinctive in nature. In the high-volume area, stability is a major concern, whereas for high-mix, production flexibility is essential. Stability can lead to an improvement in the ability to repeatedly produce products more quickly and with improved quality, this in terms of avoiding rework and scrap which can impact significantly on throughput and cost. Flexibility results in an ability to re-schedule and rapidly re-allocate resources in order to cope with frequent changes in product types, although quality issues may also constrain opportunities in this respect.

This study was initially concerned with the problems faced in the high-mix area.

8.3.2. A line segment

Figure 59 shows example flows of material and control which are encountered in a typical SMT line segment. This figure highlights the form of interactions amongst components of the line segment for it to realise inspection, quality control, and buffer size control functions. One should notice that the interactions are performed in a "hard-wired" manner, in as much that links between components are implemented directly through establishing direct physical connections between components.

8.4. Aspects in Need of Improvement

In addition to surveying details of how existing D2D manufacturing processes are organised, the case study also identified (in conjunction with the D2D personnel) potential aspects in the shop-floor organisation which can be improved. A summary of the resultant findings is presented in the cause-effect diagram in Figure 60. Here, a chain of causes and their effect leads to sub-optimal utilisation of available shop-floor

capacity and longer lead-times than could be achieved. It was found that the list of causes includes two primary ones, namely: the effect of recession and limited business integration[1] (i.e. coordination of the activities on shop-floor).

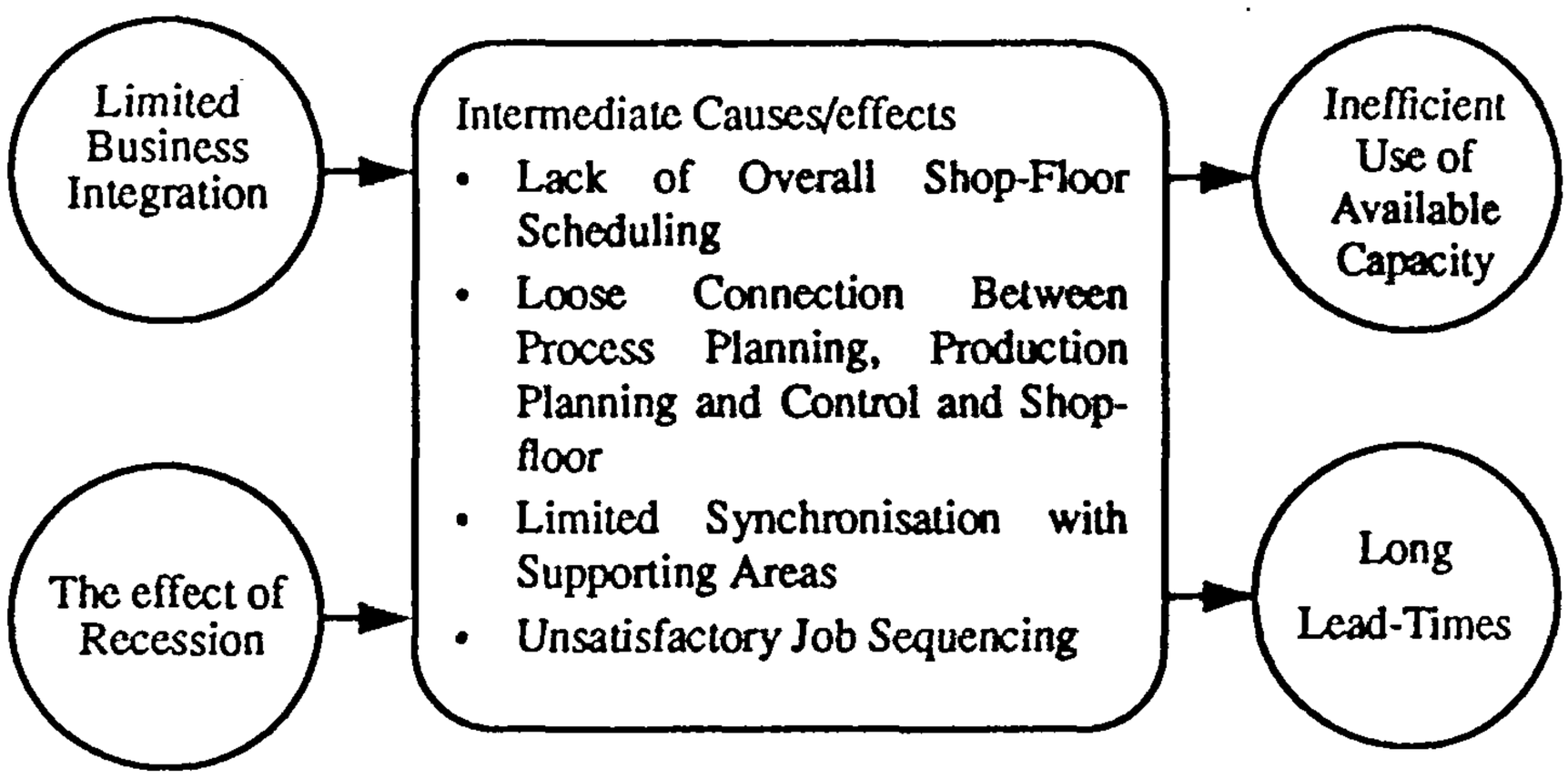


Figure 60 - Cause-effect diagram: 'as-is' situation of D2D shop-floor

It is difficult to assess the effect of recession upon the overall performance of the shop-floor. Improvements in this area would likely reduce redundancies, improve morale and ultimately enhance the overall productivity of the shop-floor. However, it is not clear whether the current economic scenario represents a temporary situation or is one driven by the environment in which manufacturing enterprises (such as D2D) must learn to survive. In either event, it was established that improved coordination of processes should substantially impact on the intermediate factors depicted in Figure 60. Here, the term coordination is used to imply the control of the interactions and relationships amongst the entities involved in realising shop-floor production.

Defining how better integration (and, as a result, improved coordination) can be achieved is by no means a simple task. Improvements could arise simply by changing the way the shop-floor is organised and managed. For instance, typically supervisors are required to control the work of individual assembly lines, whereas during the period of the case study three supervisors were independently charged with controlling the work of eight assembly lines. However, as previously discussed (see Figure 58), operation of the assembly lines is not independent so that organisation and management issues traverse the complete shop-floor.

Additionally, when scheduling the use of shop-floor resources, all combinations of possible line segments that could be used to produce a certain batch of PCB's should be considered before releasing the batch for production. To date, however, no overall

---

1. The term business integration is used here as defined in Chapter 1 and illustrated by Figure 2.

shop-floor scheduling system is employed at D2D. Consequently, technical recommendations with respect to job sequencing made during process planning are not always considered when a job is released to a particular assembly line, this quite commonly leading to unnecessarily long change-over times.

Another issue identified is the need to synchronise the shop-floor with respect to its supporting areas, such as ware-houses. For example, when a line segment is made available, following the completion of previous jobs, components and materials required to set up the line segment should already be in place. This may not occur when an operator or an assembly line supervisor fails to fetch the material in advance of job completion.

Staff training is of great importance in respect of realising any change in scenario. This research does not challenge such a premise. However, the extent to which training in itself can lead to significant improvements is not clear. In fact, an adequate combination of initiatives of an organisational and an information technology nature is a topic that transcends the scope of this research.

In any case, the aim of this research is to assess use of SEW-OSA in facilitating a re-engineering effort on the shop-floor, this in order to determine whether benefits can actually be drawn from IT based integration and hence improve coordination. Stated differently, the aim is to investigate whether an improved level of business integration can bring about significant improvements on the shop-floor.

8.5. SEW-OSA Models Produced

The application of SEW-OSA consisted of modelling the current D2D shop-floor situation, making the assumption that every resource component on the shop-floor conforms to the basic CIM-OSA paradigm (i.e. it operates as a server of functionality to the business model). The primary aim was to improve coordination by capturing within business models the means of overcoming problems which caused poor coordination. Part of the approach would be the use of CIM-BIOSYS as a means of realising integration of the active resource components on the shop-floor, so that enactment of the business model could be achieved. A basic assumption here is that opportunity for improvement would arise if all relevant interactions among shop-floor entities became driven by models.

The modelling exercise required the application of the SEW-OSA model-building capability to a domain termed as "smt line" (as shown in Figure 25). Examples of diagrams and templates produced as part of this exercise were presented earlier in this thesis, whereas a more complete description of the models produced is contained in an internal report [Aguiar 1994h]. Indeed, these models were later used as a basis for assessing the features of modelling, analysis, simulation, rapid-prototyping,

configuration and operation of the re-engineered system on the shop-floor.

Basically, the SEW-OSA model-building started by defining the boundaries of the model (i.e. the context diagram shown Figure 23). Here, the context diagram identified "PCB assembly", as the only CIM-OSA compliant domain, which is required to interface with other D2D domains through the exchange of object views and events. The other domains can be either internal or external to the Kidsgrove site or indeed to D2D itself.

The "PCB assembly" domain comprises a number of domain processes which characterise the types of functions that can be asynchronously started in this domain. However, a simplified version of a domain diagram (shown in Figure 25) limited the scope of this domain to one of the SMT lines[1]. In Figure 25, it is important to emphasise that it is not within the scope of this diagram to define whether a certain domain process represents the work of a particular assembly line. One should also note in Figure 25 that a number of occurrences of the domain process "produce board" may occur in association with one occurrence of the domain process "prepare line" (the cardinality of relationships being defined later on during the modelling process).

At the model-building stage, two levels of abstraction were introduced into the modelling process, namely:

i. A **line segment domain**, embracing the coordination amongst component elements within an SMT assembly line (i.e. a line segment embracing the scope illustrated in Figure 25).

ii. The **complete shop-floor domain**, embracing the coordination amongst different line segments on the shop-floor (see Figure 58).

8.5.1. Coordination within an SMT assembly line (i.e. line segment domain)

Coordination within the scope of an SMT assembly line involved the development of models (at the requirements definition modelling level) represented in a simplified form in Figure 61[2]. In this model, the level of granularity of the functionality coincides with that required to control interactions amongst machines and people within the scope of an assembly line. That is, functionality is described to a level at which functional operations can be individually executed by a component of the assembly line (i.e. without having to interact with any other component).

The behaviour associated with the domain process "assemble" (in Figure 61)

---

1. A complete version of the "PCB assembly" domain is presented in an internal report [Aguiar 1994h].
2. Shorter names were given to the constructs in this figure (in relation to those used in the diagrams presented in Chapter 5) to improve its readability.

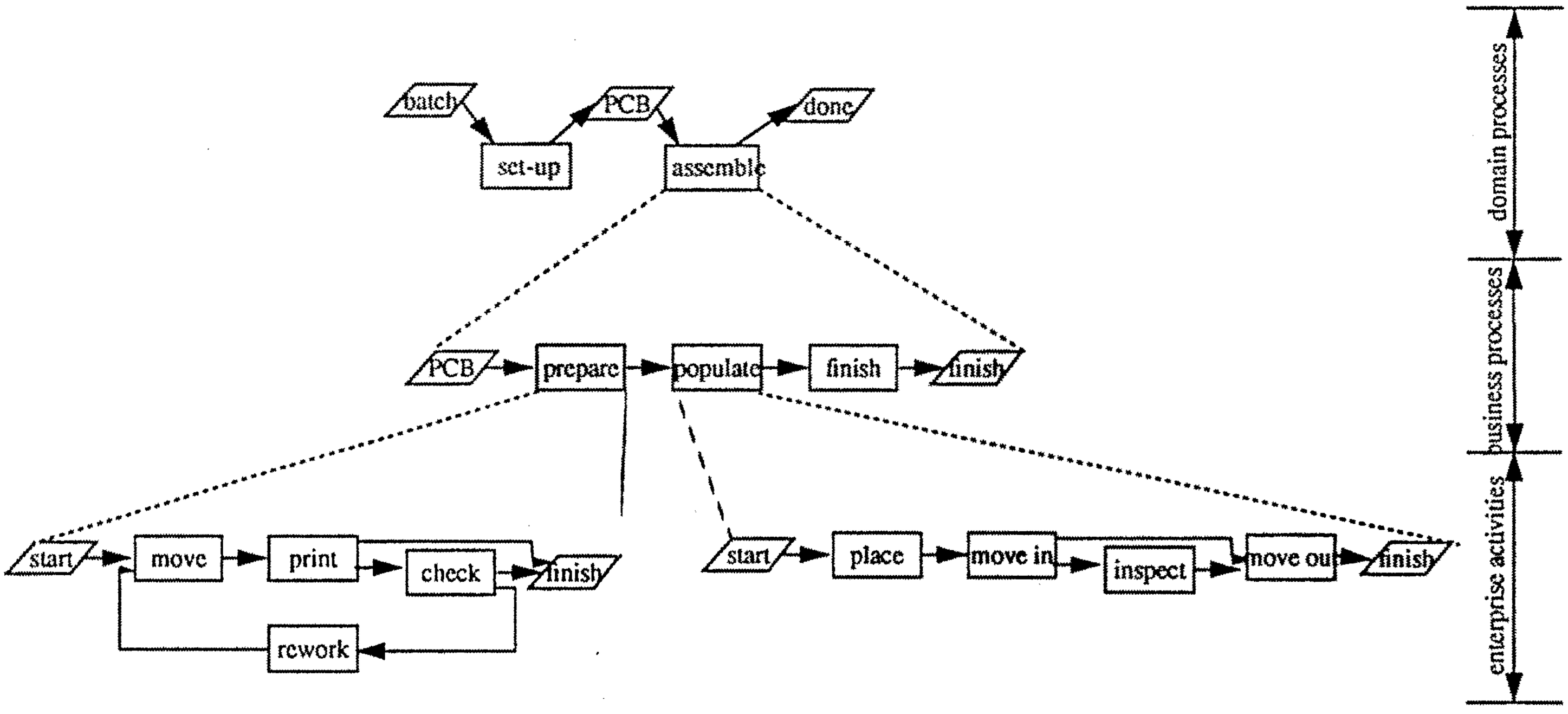


Figure 61 - Coordination within the SMT assembly line

was described as a hierarchy of behaviour diagrams. Hence, each diagram encodes a description of how a process uses its component business processes or enterprise activities. The business processes "populate" and "prepare" (in Figure 61) are of particular interest as they encapsulate control considerations requiring tighter coordination.

a. Business process "populate"

In the business process "populate", a decision is related to the frequency of inspection of PCB's, these being inspections based on: a programmed frequency of inspection or on the results of previous inspections (i.e. use of a statistical process control algorithm). Usually, the former approach is used at D2D with one in every ten boards inspected.

In respect to synchronisation of operations, when an inspection is triggered, the inspection conveyor belt (shown in Figure 59) is stopped so that an inspector can retrieve a PCB or inspect it on the conveyor. Meanwhile, upstream and downstream operations carry on. Upstream operations will only stop if there is no space left in a set of internal slots which are located between the placement machine and the inspection conveyor. Downstream operations will only stop when they run out of PCB's. When the inspector completes an inspection operation, he (or she) will inform the business model and the conveyor will be reactivated, re-enabling upstream and downstream operations (should they be stopped). Inspection operations will impact on manufacturing lead times, particularly if the inspector takes a significant time to perform them.

b. Business process "prepare"

In the business process "prepare", two functions are of particular importance, namely: (1) quality control of the printing process and (2) management of the buffer that separates printing and placement machines. These functions are relevant as typically there is an unbalance between the printing process and the placement process. The former usually requires shorter cycle times than the latter. At D2D, this unbalance is managed by the operator of the printer. This is done by printing a sufficient number of boards to keep the placement processes busy, when the interface buffer with the placement process is nearly empty (see Figure 59). Hence, the level of work-in-progress in the buffer is determined by the manner by which the operator manages the printing process.

Additionally, the operator continually monitors the quality of the printing process, by checking one in five PCB's for paste height and shape. In extreme cases, when a defect occurs, it may be necessary to re-set up the printer and wash-off of mis-printed boards. This exceptional case was not modelled in the case study. In the case of defects that affect only one PCB, the operator can wash it off and place it back on the

input conveyor. This operation is represented by the enterprise activity "rework" in Figure 61.

The examples emphasised by the two business processes described above represent only a sample of the types of control that can be exerted by SEW-OSA on the SMT assembly line. A variety of other coordination decisions are also required, namely: coordination between functions performed by the operator and those carried out in the supporting areas to provide components (i.e. paste, components in reels and sticks, etc.) and bare-boards.

Analysis of the effect that a particular configuration of the integrated system (envisaged to control the assembly line) can have upon the performance of the line is demonstrated later in the thesis.

8.5.2. Design specification stage

The design of the domain "smt line" was based on the functional decomposition of the domain formalised in functional diagrams, which led to the definition of an object diagram (such as the one depicted in Figure 62). Enterprise activities identified in the functional diagram are executed by means of interactions with active resource components via an integrating infrastructure. These interactions are modelled as messages exchanged between objects representing enterprise activities, functional entities and the information entity. Here, enterprise activities (identified within Figure 61) realise a functional transformation (i.e. act on their information and material inputs, in order to produce desired outputs), by requesting the execution of functional operations (i.e. FO's) which are carried out by the functional entities (i.e. classes of components) of the system. Functional operations can also relate to data access operations (see Figure 62).

The internal behaviour of enterprise activities is described by predicate-action Petri-nets. Figure 63 shows an example of such a description for the enterprise activity "print". Here the top and bottom transitions respectively represent initialisation and termination of a run-time occurrence of the enterprise activity. At initialisation, the Petri-net is enabled, i.e. a token is included in pEA-3_1 which enables tEA-3_2[1]. When tEA-3_2 is fired, the functional operation FO-5 is transmitted to FE-2 (see Figure 62). Then, EA-3 waits for the receipt of FO-6.

FO-21 and FO-22 represent an operation in which data is read from a data-base. This particular piece of data is a variable which records the number of PCB's produced between inspections (i.e. number of executions of the enterprise activity "check" in

---

1. The reader should refer to Appendix 4 for a description of how a Petri-net describes system evolution.

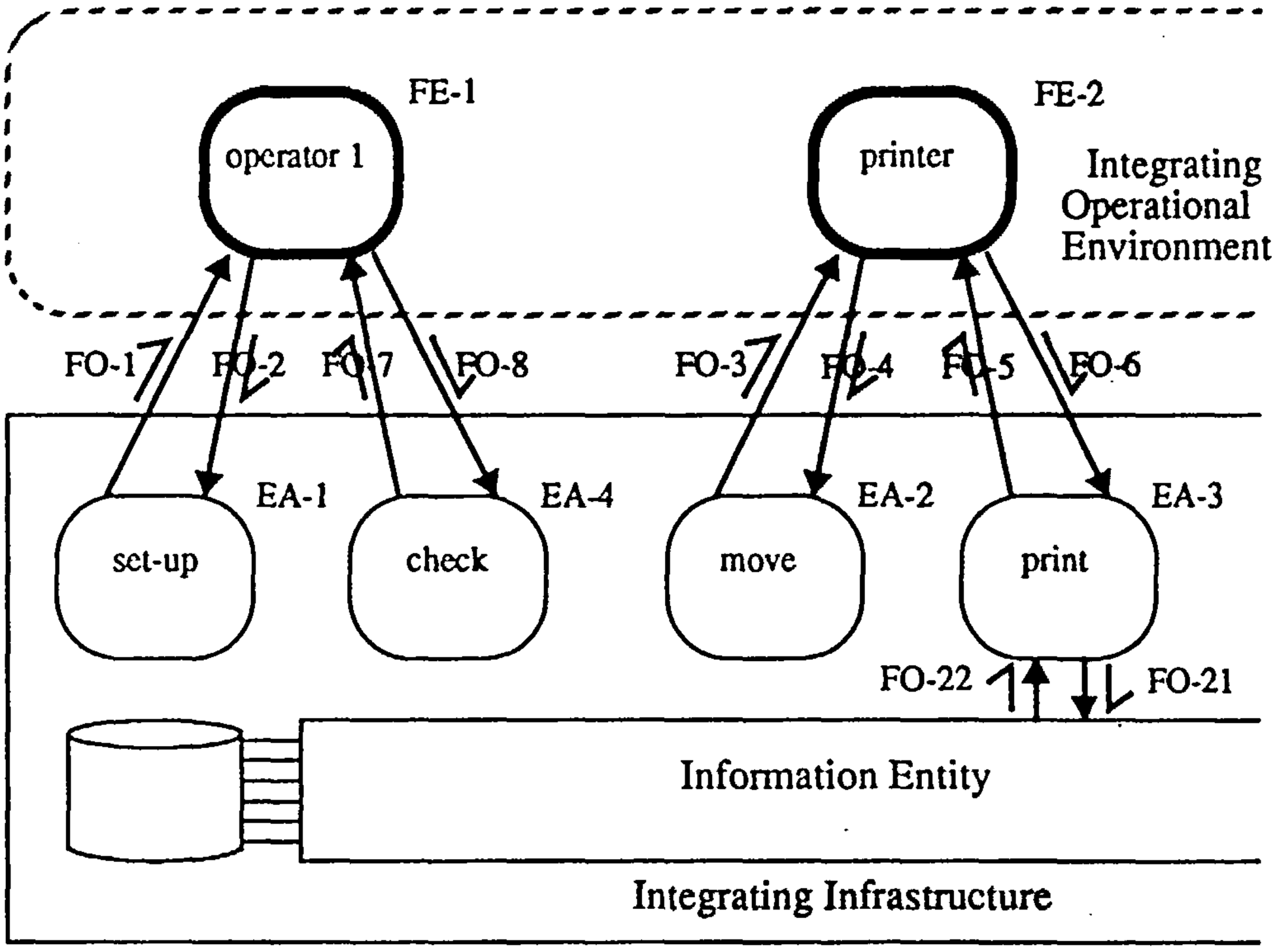
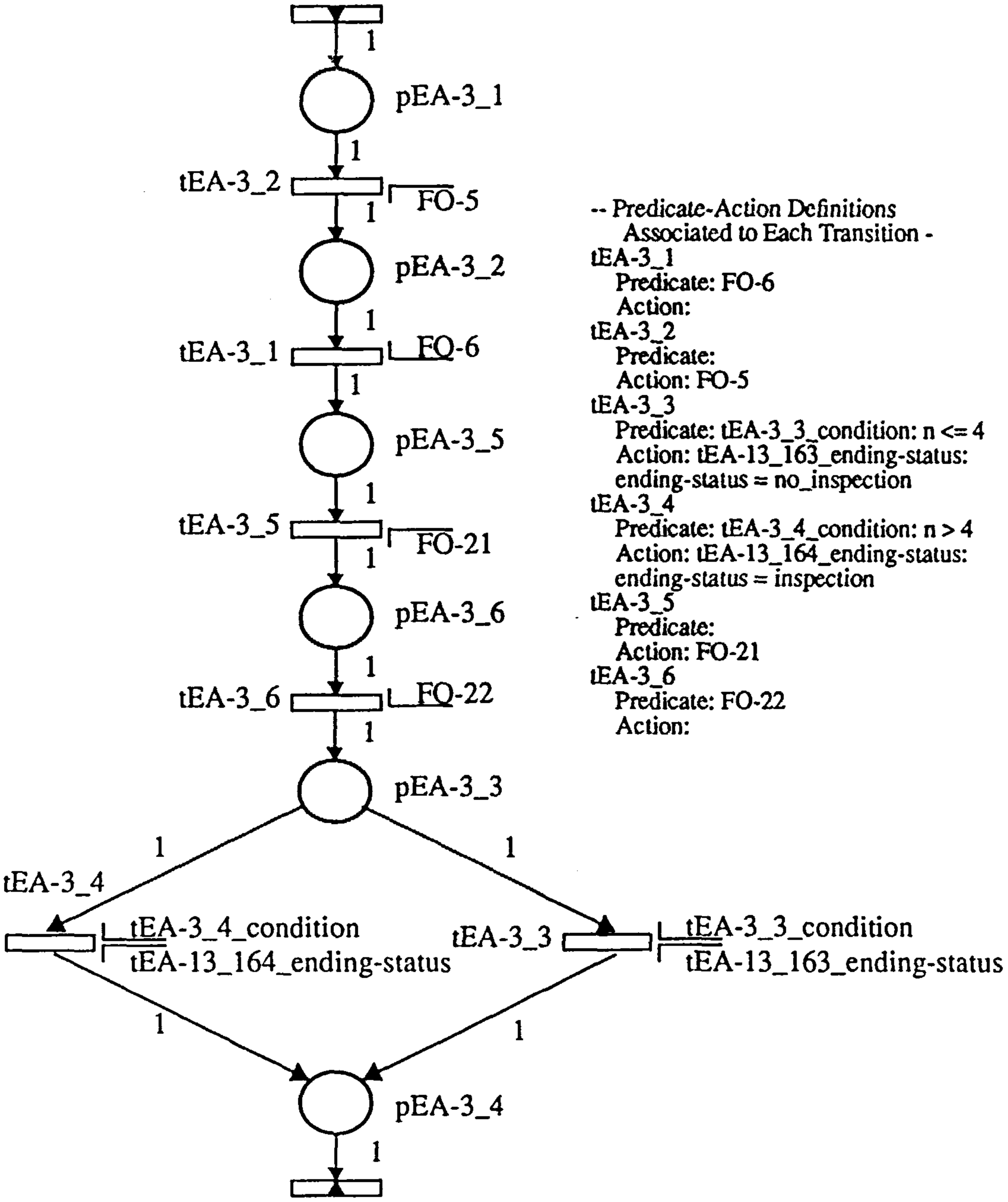


Figure 62 - Fragment of the object diagram

Figure 61). Conditions tEA-3_3 and tEA-3_4 in Figure 63 indicate alternative ending statuses for this enterprise activity. This status condition is used to make a decision as to whether to check the PCB or to finish executing the procedural rules of the business process "prepare" (hence, implementing the frequency of checking previously discussed). Similar behavioural descriptions have been defined for all remaining processes and activities in the business model. Once the business model is complete, interpreted code can be generated for the system (i.e. model-enactment can be facilitated to enable rapid-prototyping of the system) in order to test the solution designed. This code along with other code fragments for the remaining constructs of the business model can then be executed by the business entity to coordinate interactions amongst system components.

At this stage, the functional definition of the system being designed is complete. However, two issues remain, namely: (1) the specification of physical resources to provide such a functional content; and (2) the definition of an appropriate topology for the computer system to integrate those physical resources. The resource diagram (depicted in Figure 33) shows a resource specification for the domain processes "set-up" and "operation". This specification was produced through using the resource modelling facility (described later in Chapter 10). Additionally, the configuration diagram (see Figure 34) shows how the active resource components of the assembly line are integrated via a network of computers.



-- Predicate-Action Definitions
Associated to Each Transition -
tEA-3_1
Predicate: FO-6
Action:
tEA-3_2
Predicate:
Action: FO-5
tEA-3_3
Predicate: tEA-3_3_condition: n <= 4
Action: tEA-13_163_ending-status:
ending-status = no_inspection
tEA-3_4
Predicate: tEA-3_4_condition: n > 4
Action: tEA-13_164_ending-status:
ending-status = inspection
tEA-3_5
Predicate:
Action: FO-21
tEA-3_6
Predicate: FO-22
Action:

Figure 63 - EA-3: behaviour diagram: print

Having populated SEW-OSA with data from the D2D assembly line, simulation runs were carried to study the performance of the system under various operating conditions. More detailed information about these studies is presented later in Chapter 11.

Based on findings of the simulation study, an assembly line configuration in terms of resource components was prototyped using the SEW-OSA model-enactment capability. The replacement of emulated components by physical ones was also demonstrated by integrating SEW-OSA with a separate simulation model of the

assembly line (i.e. a shop-floor animator developed by I. S. Murgatroyd and discussed later in Chapter 9). In the animator, assembly line components behaved towards the business model in a similar manner as would physical components. Details of the results obtained at the rapid-prototyping stage are presented later in Chapter 11.

8.5.3. Coordination amongst different line segments (i.e. the complete shop-floor domain)

In order to provide support for coordination amongst different line segments on the shop-floor, the level of granularity of the functionality modelled was defined to coincide with the scope of functions provided by each line segment shown in Figure 58. Hence, Each line segment was made to relate to enterprise activity classes, being each type of material flow indicated in Figure 58 related to one enterprise activity instance. Similar diagrams to those obtained in for the "line segment domain" were also obtained for the "complete shop-floor domain". However, constructs associated with each domain process (i.e. enterprise activities, functional entities and active resource components) were defined so that they encapsulate tasks performed by a given line segment, as opposed to task performed by each component of a line segment.

The difference in purpose between the models obtained for the "line segment domain" and for the "complete shop-floor domain" is the fact that, in the latter, modelling was carried out only for the purpose of studying the organisation and associated performance of the system whereas, in the former case the aim was to generate a prototype of a system configuration and study its performance. This difference led to the creation of models at different levels of granularity, whereby the "line segment domain" represents a more detailed description of functionality, as compared to the model of the "complete shop-floor domain".

These two levels of granularity were defined to facilitate the execution of simulation with models of the "complete shop-floor domain". A more detailed model of the shop-floor to the same level of granularity of the model of the "line segment domain" could have been obtained. However, the number of constructs in the model would have been so large that simulation runs would have required excessive computer power to enact the model (see later discussion on this issue in Chapter 11). Furthermore, additional detail would not have provided information that was not already encapsulated in the domain processes of the "complete shop-floor domain".

In regard to coordination issues on the shop-floor, motivation for the simulation study included: (1) identification of critically constrained resources (i.e. bottlenecks); and (2) tests of the impact of different shop-floor configurations (and associated scheduling strategies) upon performance metrics, such as: throughput, level of work-in-progress and level of utilisation of the line segments. However, as explained later in Chapter 11, limitations in the simulation tool did not allow simulation studies to be

performed.

8.6. Outline of a 'should-be' System

An outline of a 'should-be' system was proposed based on the application of the model-building and model-enactment capabilities to the "line segment domain".

Figure 64 depicts the computer architecture proposed for integrating existing components of the SMT assembly line. Each machine component is physically linked to a CIM-BIOSYS host by means of the interface and protocols provided by its controller. These protocols can be implemented by appropriate device drivers and related functional operations, according to the methodology described in the two next chapters. An interface is assigned to each operator to enable he (or she) to receive instructions from the business model, as well as to allow them to feed back data and status.

With regard to extending this configuration to the complete shop-floor, it should be noticed that there should be an instance of SEW-OSA model-enactment services serving each line segment. This would confine coordination of activities within the scope of a line segment, whilst providing a link to the remaining line segments via a line segment scheduler.

Any scheduling decisions required within the scope of a line segment, are assigned by the business entity of SEW-OSA to this line segment scheduler. The line segment scheduler does not operate as a cell controller. Essentially, CIM-OSA eliminates the need for conventional cell controllers as such[1]. Indeed, this function is performed by the business entity when it executes a business model. However, as previously discussed, CIM-OSA requires an external scheduler (and associated a scheduling algorithm) which make decisions about the allocation of physical resource components. A discussion of scheduling issues is presented in Appendix 9.

Associated with the system outlined in Figure 64, a considerable amount of implementation issues must still be added by further work (e.g. realisation of hardware and software interfaces and software development for functional operations). These implementation issues are beyond the scope of this research.

---

1. A cell controller is an element responsible for coordinating the interactions amongst manufacturing devices within a manufacturing cell by dispatching jobs to be executed by these devices.

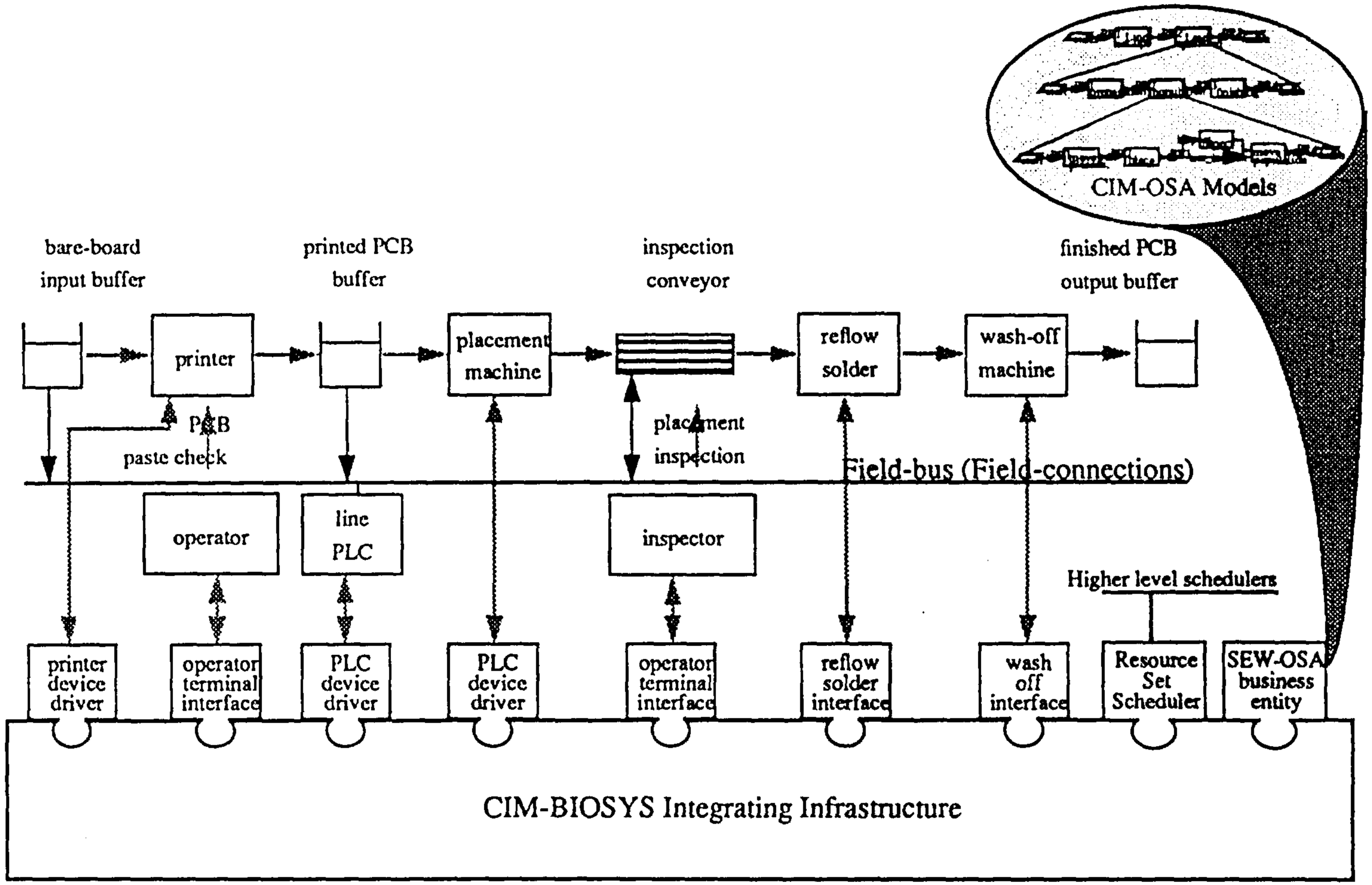


Figure 64 - SMT assembly line integrated by SEW-OSA

8.7. Benefits of the Case Study to D2D

The case study involved understanding the wider-picture of D2D's systems and processes, modelling D2D shop-floor and enacting models of one of its assembly lines. These activities have led to attaining the following benefits (by D2D):

- reports (by an external and independent person, working as a consultant) on an evaluation of some of their systems and processes, with particular emphasis to their shop-floor;
- the proposition and evaluation of a 'should-be' system derived from model-enactment exercises, the results of which can offer the basis for decisions as to whether further investments can be made on a model-driven operation of the shop-floor;
- the availability of models over which further studies can be carried out;
- an understanding of how a model-driven approach could be used to support changes in the manner by which current systems and processes are organised;
- transfer of knowledge (from the MSI Institute) about methods and architectures. This is now motivating them to apply modelling technology to other problems within the company, such as the definition of business and IT architectures for the whole organisation.

8.8. Limitations

Unfortunately, it did not prove practical to realise an implementation of SEW-OSA-driven solutions on the D2D shop-floor. The results did reach a level of initial specification (as to how the shop-floor should be organised), but without realising a physical implementation it is clear that validation was at best, partially achieved. Implementation would have required a level of resource, effort and time that was not available during the period of this research, either to MSI or D2D.

Indeed, any decision by D2D to apportion resources to support implementation depends upon preliminary results from SEW-OSA modelling and analysis work. Additionally, certain aspects of detailed work necessary to realise a commercially viable and practical solution may not have added value to the research results obtained.

A further limitation of this case study is related to the fact that the quality and quantity of the results obtained will be constrained by the data gathered to support it. Here, time and resource constraints at D2D, allied to the dynamic nature of D2D shop-floor operations, limited the accuracy of the 'as-is' description. In this respect, the quality of the data gathered in the case study can also be a measure of the effectiveness of the methodology proposed in Section 8.2 (to overcome such limitations).

8.9. Contributions from the Case Study

Three primary contributions of knowledge have come from this case study work, as follows.

(1) This is the first case study (known to the author) on the application of CIM-OSA in an electronic manufacturing company. It is also one of the most complete CIM-OSA case studies carried out in the sense that it sought to validate the use of a CASSE environment for systems engineering (i.e. SEW-OSA).

(2) The case study has also demonstrated the usefulness of CIM-OSA (when realised as part of SEW-OSA) in addressing contemporary manufacturing systems design issues. At D2D, this made the importance of a model-driven approach to tackling systems design well known to key staff of the company and, in addition, enabled them to understand the concepts of CIM-OSA.

(3) Finally, as a result of the modelling, simulation, analysis, rapid-prototyping and configuration activities, possible improvement scenarios have been identified for the D2D shop-floor.

Chapter 9 - 'Run-Time' Execution of the Physical System

This chapter outlines how a business model refined from a rapid-prototyping exercise can be put to work to coordinate activities of actual physical components. This is achieved by supporting a methodology which structures the processes involved in producing simulated and emulated descriptions of a system, as well as producing a more complete description of the physical system.

Three aspects of enabling 'run-time' execution[1] of a physical system are discussed in this chapter, namely:

- the organisation of functionality and behaviour required from the system in order to comply with a SEW-OSA environment (Section 9.1);
- a preliminary specification of a presentation entity for SEW-OSA (Section 9.2);
- observations in respect to the manner in which coordination issues impact on the structure of active resource components (in order for them to be integrated into the SEW-OSA environment) - Section 9.3; and
- the strategy to overcome the inexistence of either a presentation entity or a physical implementation of CIM-OSA-compliant resource components (within the context of the "Model-Driven CIM" project - Section 9.4).

Here, it is important to note that, although CIM-OSA defines an overall specification of a presentation entity, it does not define how active resource components should be structured. The absence of this definition was one of the main motivation for including the discussions presented in this chapter. Hence, this chapter seeks to make observations and recommendations (which have yet to be formally proven) regarding the organisation and integration of resource components within the CIM-OSA architecture (as adopted in SEW-OSA).

Hence, this and the next chapter present the author's interpretation of the impact of using SEW-OSA on: (1) the structure of resource components and (2) a framework for integrating modelling tools and services produced by other researchers working in the "Model-Driven CIM project". In this context, whilst previous chapters (and Chapter 11) characterise the essence of this Ph.D (i.e. its body of proven research), this and the next chapters do not directly add to the body of the author's research. Rather they help clarify the context of the study. They also illustrate the impact of underlying

---

1. The term "system execution" means the execution of a business model in support of the operation of a physical system.

architectural definitions of SEW-OSA on supporting tools for system integration, as well as on the final product resulting from applying these tools.

9.1. Support of System Execution in SEW-OSA

Key issues involved in supporting the execution of a physical system include:

- a view of the way in which SEW-OSA supports the implementation description modelling level of CIM-OSA;
- the proposition of a structure, with which active resource components should comply, in order to enable integration into a SEW-OSA compliant environment;
- an analysis of the impact that such a structure imposes upon coordination issues within the environment;
- a discussion of repercussions arising from coordination issues in respect to the level of granularity with which the functionality of active resource components should be defined.

9.1.1. The implementation description modelling level in SEW-OSA

As previously discussed, associated with each model of an active resource component (handled during the rapid prototyping stage) there should be a physical implementation of the resource component which can be physically integrated. Thus, activities involved in the implementation description modelling level should be focused on replacing emulated models of resources by their physical counterparts. This process should also correspond to the last phase in the design process, so that a system emerges out of the design activities of modelling, analysis, simulation, rapid-prototyping (including use of emulated components), and the gradual replacement of emulated components by physical ones. Hence, as later described in Chapter 10, from this process emerges a need for a library of models which represent physical components. Models of resource components from this library can then be selected and integrated into the SEW-OSA environment and tested.

Once the designer is satisfied with the performance of a certain configuration of resource models, instances of physical resources can replace their counterpart models and be tested in site by means of a refined version of the system prototype. This final stage will, henceforth, be referred to as "system execution". The replacement process should be performed gradually (i.e. on a component-by-component basis), if physical considerations permit (i.e. certain physical resource components may depend upon one another for proper operation). Should instances of physical resource components not be available, tests on the models can provide a more rational basis for decisions regarding the acquisition of physical instances. Within this context, models provide a strong basis

for the decision making process.

9.1.2. CIM-OSA-compliant active resource component

Further to the provision of models which describe how components work, also required is the provision of CIM-OSA-compliant resource components. As can be observed from Figure 65, CIM-OSA-compliant active resource components operate as servers of functionality to the level of granularity required by their client (i.e. in the form of functional operations). Here, the client is the business model executed by the business entity. In this respect, the business model is the only decision maker with regard to coordinating the execution of functionality. If the level of granularity at which an active resource component is defined coincides to that of an individual human being, a machine or an application program, each atomic piece of functionality provided by such a component must be related to the work that the component is capable of executing without interacting with other components (i.e. in such cases interactions can be encapsulated by the business model).

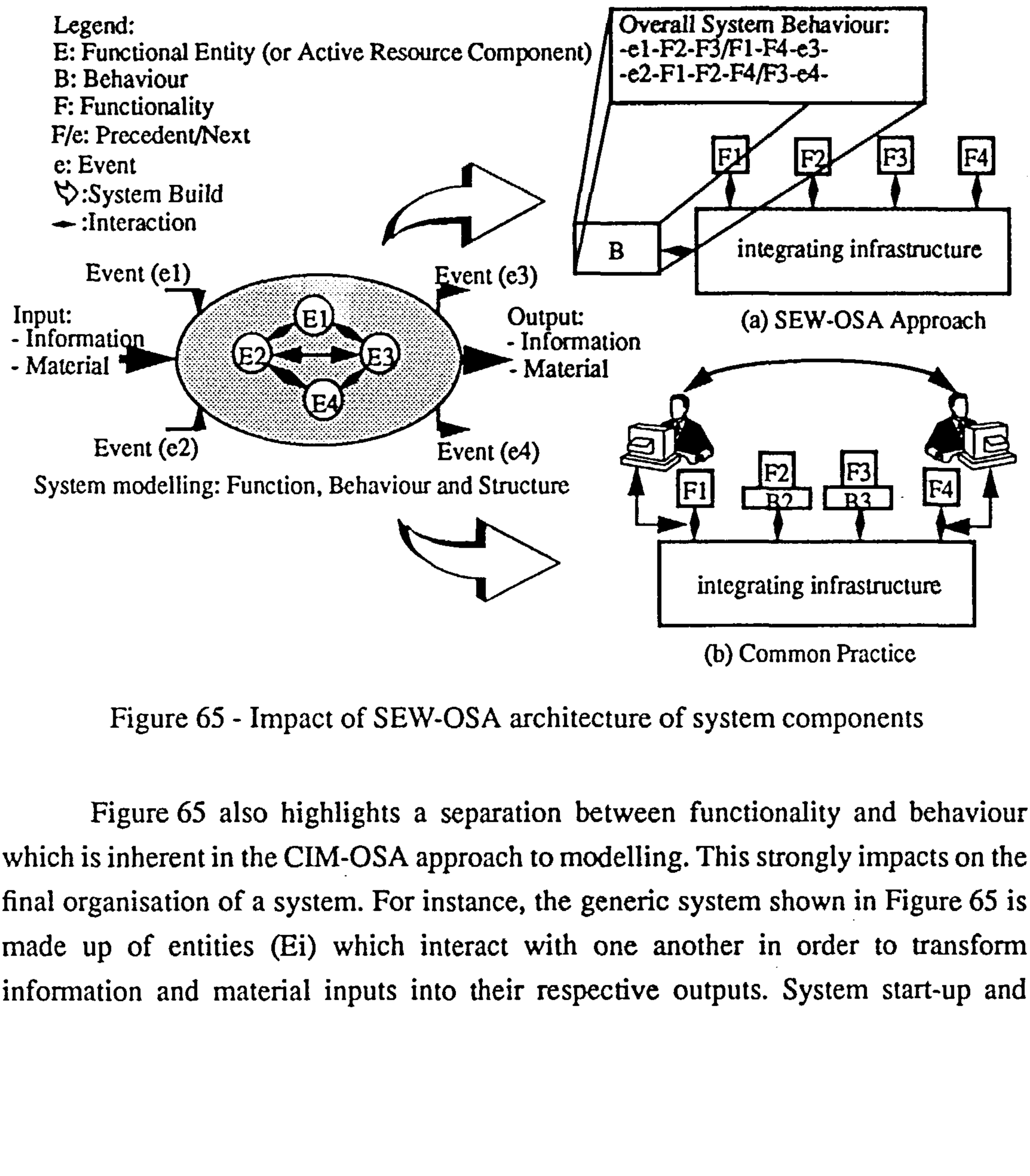


Figure 65 - Impact of SEW-OSA architecture of system components

Figure 65 also highlights a separation between functionality and behaviour which is inherent in the CIM-OSA approach to modelling. This strongly impacts on the final organisation of a system. For instance, the generic system shown in Figure 65 is made up of entities (Ei) which interact with one another in order to transform information and material inputs into their respective outputs. System start-up and

termination are defined by events (ei). Enterprise modelling focuses on engineering the relationships among its components in order to enable the system to achieve its goals. Such an engineering process includes the definition of an adequate selection of components and an appropriate design for the interactions among them (see Figure 65).

Regardless of the separation between functionality and behaviour, when CIM systems are built (i.e. as in Figure 65.b) entities are modelled as monolithic elements. Coordination among the elements of functionality (i.e. system behaviour) may be either intrinsically designed in the components or implemented via human interaction. That is, aspects of the system behaviour are embedded within the functionality of its elements (e.g. such as within software code or as part of the operations performed by a person).

In SEW-OSA, overall system behaviour is captured by the business model (as illustrated in Figure 65.a). Subsequently, "centralised"[1] execution of that model drives the system operation. Any change required in system behaviour needs to be reflected in this unique model at engineering time, in order for it to be effected at run-time in a consistent way.

Figure 65 also highlights that the CIM-OSA approach offers improved flexibility over conventional practice, in respect of making changes to the way the system operates (i.e. changes in behaviour or functionality, because they are de-coupled). In the CIM-OSA approach, decisions regarding the coordination of system functions are captured in the business model. In the traditional approaches, these decisions are distributed among applications that perform system functions. Thus, difficulties arise when changing either the various elements of system functionality or behaviour, due to the need to re-implement hard-coded hidden coordination functions contained inside each application.

The idea of extracting coordination decisions from the components stems basically from the fact that essentially those decisions characterise the manner in which an enterprise is organised at the level of business integration (as early illustrated in Figure 2) and hence there is an implicit need to facilitate their change.

9.1.3. CIM-OSA-based coordination

Model-based coordination of the interactions among system components requires:

- access to the operations (i.e. the functionality) of system components;

---

1. Here, the term centralised does not necessarily mean that model execution takes place on just one computer host. Centralisation is related to the existence of a unique model for the complete system, which can be executed across a number of computer hosts.

- predictability of all possible outcomes of the execution of such operations.

Therefore, in the context of a CIM-OSA modelling approach, ideally system components should be developed in such a way that:

a. they operate as servers of functionality which can be encapsulated into functional operations;

b. when they operate as clients, their requests can be transformed into events which trigger the execution of appropriate threads of the business model, this to process the request;

c. they do not directly access information elements relevant to integration (as identified in the information view), for this type of information access should be encapsulated by the business model and performed as part of the occurrence of an enterprise activity.

Hence, it is quite straightforward to envisage a CIM-OSA-compliant component when dealing with shop-floor machines. After all, a business model does precisely what traditional cell controllers are expected to provide, in terms of coordinating the execution of jobs on the shop-floor. Human beings when playing the role of mere executers of decisions (made elsewhere) can also be made to comply with requirements of a CIM-OSA business model. However, re-engineering legacy software, so that they can inter-operate with the remaining components of the system may be far more complex.

Commonly, legacy software is not developed to provide ready access to internal functional operations, so that they can be triggered by a business entity. Likewise, as human beings commonly function as creative components of a system, they are not easily encapsulated as a CIM-OSA-compliant component. Indeed, a prime difficulty with such an encapsulation is that it requires the ability to associate and establish a defined level of predictability and causality in the relationships among system components.

Two alternative approaches can be used to facilitate modelling (and hence integration) of components which do not readily comply with requirements "a", "b" and "c" above, namely:

(1) To model such components as non-CIM-OSA-compliant domains;

(2) To encapsulate the functionality of such components within a single functional operation.

Both approaches facilitate integration of components by providing adequate

interfaces for their constituent components to interact with a CIM-OSA-compliant environment. These interfaces can be provided by creating "wrappers"[1] for the functionality concerned.

In the case of approach (1), components represented as a non-CIM-OSA-compliant domain (see Figure 23) interact with other components by exchanging object views and events, in the form of message exchange or data sharing. In this case, components interact with the rest of the system in a completely asynchronous manner.

For the approach (2), component interfaces with the rest of the system must comply with requirements "a", "b" and "c" above. If adequate "wrappers" are developed to encapsulate components, requirement "c" is the one that imposes the greatest constraint. Indeed, requirements "a" and "b" can be emulated by event exchanges between domains defined via approach (1).

Based on the assumption that system components will offer their functionality by complying with requirements "a", "b" and "c", next sections present a discussion of the issues involved in the integration of the three classes of active resource components (i.e. machines, human beings and application programs).

9.2. Presentation Entity in SEW-OSA

Interaction with physical resource components as opposed to emulated ones requires important changes in the operation of SEW-OSA. At the rapid-prototyping stage, the Resource Manager is required to interact with application shells which emulated the internal behaviour of active resource component. When physical instances of active resource components are used, the Resource Manager will interact with three basic classes of component belonging to the presentation entity, namely:

- machine drivers, to facilitate interaction with machine functional entities[2];
- user interfaces, to facilitate interaction with human functional entities;
- application enablers, to facilitate interaction with application functional entities.

As previously stated, the main purpose of the presentation entity is to provide remaining entities of CIM-OSA with a uniform means of interacting with inherently heterogeneous resource components. If resource components were to provide their

---

1. According to the working group 3 of the ICEIMT workshop III [Petrie 1992b], a wrapper is "a [set of] instructions that conform to a defined interaction protocol".
2. The term "functional entity" when referring to a physical instance of a resource (coined by the ESPRIT/AMICE consortium [ESPRIT/AMICE 1989]) is used here as a synonym for active resource component.

functionality directly in the form of functional operations, there would be no need for the presentation entity. However, that is not the case.

CIM-OSA does not prescribe a detailed structure for implementing the presentation entity. Within the context of this research, it is envisaged that SEW-OSA should support the automatic generation of an occurrence of a presentation entity (PE) component to interact with each active resource component in the system. Figure 66 shows the structure of a PE component proposed in this research. Such a component works as a translator between a language understood by active resource components (which is technology dependent) and that of CIM-OSA (which is technology independent). In order to perform this function, the PE component uses a script file which relates each command in the technology independent language into a sets commands in the technology dependent one (see Figure 66).

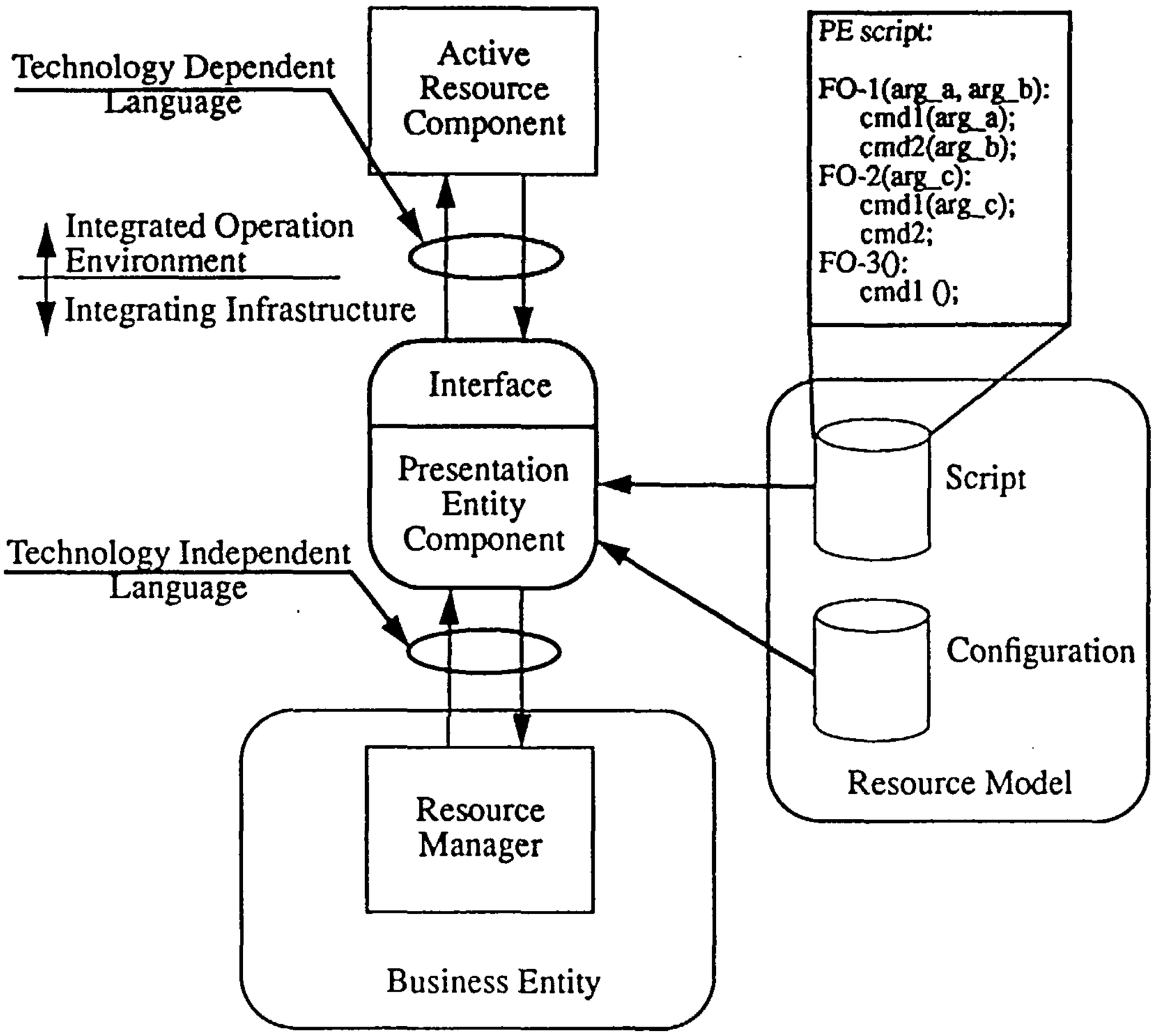


Figure 66 - Architecture of a generic presentation entity component

Here, an underlying assumption is made that the level of detail encapsulated by the technology independent language is less or equal to that of the technology dependent one.

In SEW-OSA, it is envisaged that a component would be selected from a resource model (as discussed later in this thesis), which encapsulates information about the component configuration, namely:

- the interface required by the PE component in order for it to interact with the active resource component (such as the functionality of a CIM-BIOSYS device-driver, enabling it to interact with a CNC-machine);
- the identifier of a script (i.e. a translation table) which enables mapping of technology dependent commands (used by the active resource component) into the functional operations that are manipulated by the business entity.

SEW-OSA was conceived so that when rapid-prototyping with physical resource components is carried out an occurrence of a PE component is started. The occurrence of this PE component is automatically configured so that it can execute on an identified host and be shared with the active resource component, this according to the data represented in the configuration diagram (as illustrated in Figure 34).

9.3. Integration of Physical Resource Components in SEW-OSA

The structure depicted in Figure 66 implies that at run-time a CIM-BIOSYS process is generated to carry out the role of the PE component, whereas another entity implements the functionality of an active resource component (where this entity can be either a CIM-BIOSYS application or an alien application [Weston 1993]). It was proposed that this architecture should be adopted for every type of resource component (i.e. machine, application program or human being)[1]. Nonetheless, variations occur due to the particularities of each resource component, as discussed in the following sub-sections.

9.3.1. Machine functional entity

The structure depicted in Figure 66 can be adopted to facilitate interfacing with machine functional entities. This is specially so as in this type of resource the separation of technologies implied in Figure 66 coincides with a physical separation between the integrated operation environment and the integrating infrastructure.

Thus, the structure shown in Figure 66 can be populated with a device driver in the PE interface, which implements the protocol engine to converse with the machine via a physical interface (e.g. an RS-232 serial link). An example of a script file which relates the technology independent commands for machines and technology dependent

---

1. The types of resource components are referred in this thesis as functional entities, as coined in the CIM-OSA terminology.

commands for a CNC machine is shown in Figure 67. Here, whenever a functional operation make(Part_A) is received from the Resource Manager, the PE component executes the sequence of commands listed in Figure 67, namely: down-load program "**Part_A.program**" followed by "**start**" [program]. In order for this to occur, the PE component reads the program stored in a database (via the information entity services); starts the program; and monitors program termination.

```
FO-1: make(Part_A):
    download(Part_A);
    start;
FO-2: made(Part_A);
    end_of_program(Part_A);
```

Figure 67 - Example of a script for a machine functional entity

"Part_A.program" is an CNC program built and tested (in advance of modelling activities) by the engineering area of the enterprise. The only relationship between "Part_A.program" and SEW-OSA is through the identification of "Part_A.program" as a necessary information object view, required to execute the functional operation make(Part_A), referred to in the business model.

When the program is completed, the machine will feed back a status report (i.e. "end_of_program") this being translated into the functional operation made(Part_A) and transmitted back to the Resource Manager.

One should observe that if interactions are performed using a machine that facilitates control of a finer level of granularity (at which operations hard-coded in the program "Part_A.program" are formally defined in the script, e.g. move part to the assembly envelope, place component A on position x, etc.), a greater level of flexibility would be allowed. However, conversely, an undesirable level of detail would have to be dealt with.

Finally, it is important to notice that, although it is envisaged that SEW-OSA controls logical connections between the software components involved, appropriate hardware connections should be in place to interact with physical components (which may reside in hosts alien to CIM-BIOSYS). For example, in order to integrate to the placement machine, all hardware connections between it and the PE component host are assumed to be realised.

9.3.2. Human functional entities

The structure depicted in Figure 66 can be directly applied to human interfaces. Here, separation of technologies coincides with a physical separation between the integrated operation environment and the integrating infrastructure.

The basic requirement of a CIM-OSA resource component (i.e. to operate as a server of functionality structured as independently accessible elements) requires that the tasks and decisions performed by human beings in the system are structured so that they can be triggered by events captured by the business model and that they are executed independently by each human functional entity. Hence, cooperation between human function entities is encapsulated in the business model. Exceptions can be made where a group of people interact directly among themselves and interact with the rest of the system modelled by CIM-OSA via a single instance of a human functional entity. However, in this case, coordination issues related to the tightly coupled group are not included within the business model.

Hence, the structure in Figure 66 can be populated with a human interface which implements a protocol engine and translates functional operations into instructions in a language or format that can be understood by a human being. Likewise, responses and instructions from the operator should be transformed into defined functional operations. An example of a script file that illustrates this concept is shown in Figure 68. Here, whenever a functional operation do(task_B) is received from the Resource Manager, the PE component changes the colour of a widget on the interface, this serves to tell the operator that the task represented by such an event needs be performed. When the operator completes the task, he (or she) informs the interface and a functional operation done(task_B) is passed back to the Resource Manager. The script file could also incorporate more sophisticated means of interfacing (e.g. voice and image recognition and generation, animation, etc.). However, one should notice that this class of interactions with human beings is of a mechanistic nature. This is evidenced by the finite nature of the script. That is, the operator is not allowed to respond in a form that has not been foreseen by the model.

```
FO-1: do(task_B):
    change_widget(list);
FO-2: done(task_B);
    callback_buttom_done(task_B);
```

Figure 68 - Example of a script for a human functional entity

9.3.3. Application functional entities

Integrating existing application functional entities (i.e. legacy software) to the CIM-OSA architecture presents many challenging problems. As previously discussed, contemporary application functional entities seldom comply with the basic features of a CIM-OSA-compliant resource component. Commonly, therefore, they will not operate as a server of functionality within a system, and their functionality may be

partly structured with their component building blocks quite inter-dependent of one another. Additionally, it may be that a large proportion of the code of existing applications is devoted to coordinating the execution of pieces of functionality. A further complicating factor is that CIM-OSA seeks a major separation between the core functionality of applications and their user interface. Indeed, CIM-OSA-compliant application functional entities do not possess a user interface. The functionality provided by a CIM-OSA application functional entity comprises purely of computer processing which is executed without human assistance. This complies with the principle of atomicity which says that a functional operation is a piece of functionality that the resource component should be able to execute on its own, without interacting with other resource components [ESPRIT/AMICE 1993a].

Even when application functionality and human interface functions are grouped into one entity (i.e. system component), other sources of complexity arise from integrating heterogeneous software applications. Among the various sources of complexity[1], the issues of coordination are of particular interest in this research.

Coordination issues comprise of the coordination tasks embedded in software applications for controlling the synchronisation and sequencing of pieces of functionality, in a form that does not take into consideration overall system coordination issues. That is, complexity arises from the convoluted manner in which two basic sets of coordination issues coexist in a software application, namely: tasks that applications accomplish and assumptions they make about the functionality that other system components. Therefore, addressing the conflict between coordination decisions coded within software applications and those embodied in the business model is essential to achieving coherent overall system coordination. If these conflicts cannot be resolved, software applications will need to be re-structured, so as to comply with the mode of operation discussed in Section 9.1.

By considering the requirements imposed by CIM-OSA upon the mode of operation of its functional entities, a way of re-structuring applications could be realised by adopting the structure proposed by Figure 69. Such a structure implies the re-organisation of existing software applications into two sets of functions: (1) those related to tasks performed by the software application of the system (i.e. its actual functional components) and (2) those related to decisions about what the software application should do next (i.e. its local coordination functions). Generally speaking, functionality encompasses tasks performed by functional modules (or group of

---

1. Some other issues are: the heterogeneous nature of the infrastructures, operating systems, programming environments, networks and hardware platforms upon which software applications are developed; and the proprietary nature of the protocols used by software applications to interact with other system components;

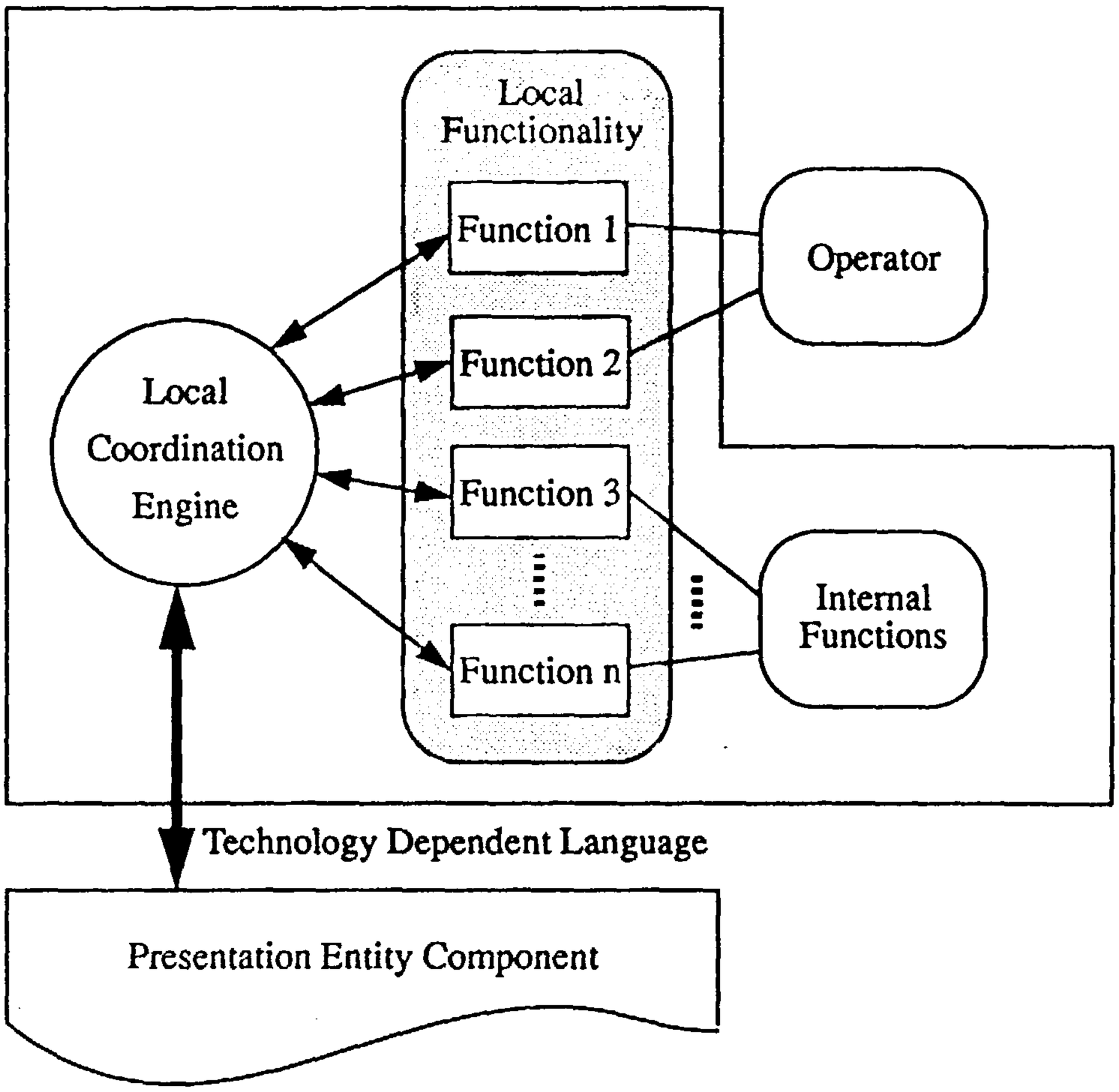


Figure 69 - Proposed structure for an application functional entity

functional modules) contained within software applications and those that are supported and/or executed by the operator of the software application. Within local coordination functions are decisions related to the execution of local functionality and decisions related to external functionality (i.e. the functionality provided by other system components).

The structure depicted in Figure 69 complies with the form in which active resource components are modelled in Chapter 5. Building software applications based on this structure consist of populating the active resource components described by entity behaviour diagrams (see Figure 32) with their relevant functionality described as functional operations. That is, predicate-action Petri-nets populate the local coordination engine of applications, whereby local functions are triggered by the firing of transitions (see Figure 69).

9.4. Current Implementation of Interactions with Active Resource Components in SEW-OSA

Three issues summarise the current status of this research with respect to the realisation of interactions between SEW-OSA and active resource components, namely:

- the idealised structure of an active resource component which was assumed for modelling purposes;
- associated functional operations to be supported in SEW-OSA;
- the strategy adopted (and realised within the context of the "Model-Driven CIM" project) for demonstrating these interactions.

9.4.1. Ideal active resource component

Ideally, a CIM-OSA compliant resource component should interact with the integrating infrastructure via PE components (see Figure 66), this by implementing the specified structure of a functional entity shown in Figure 70. According to this

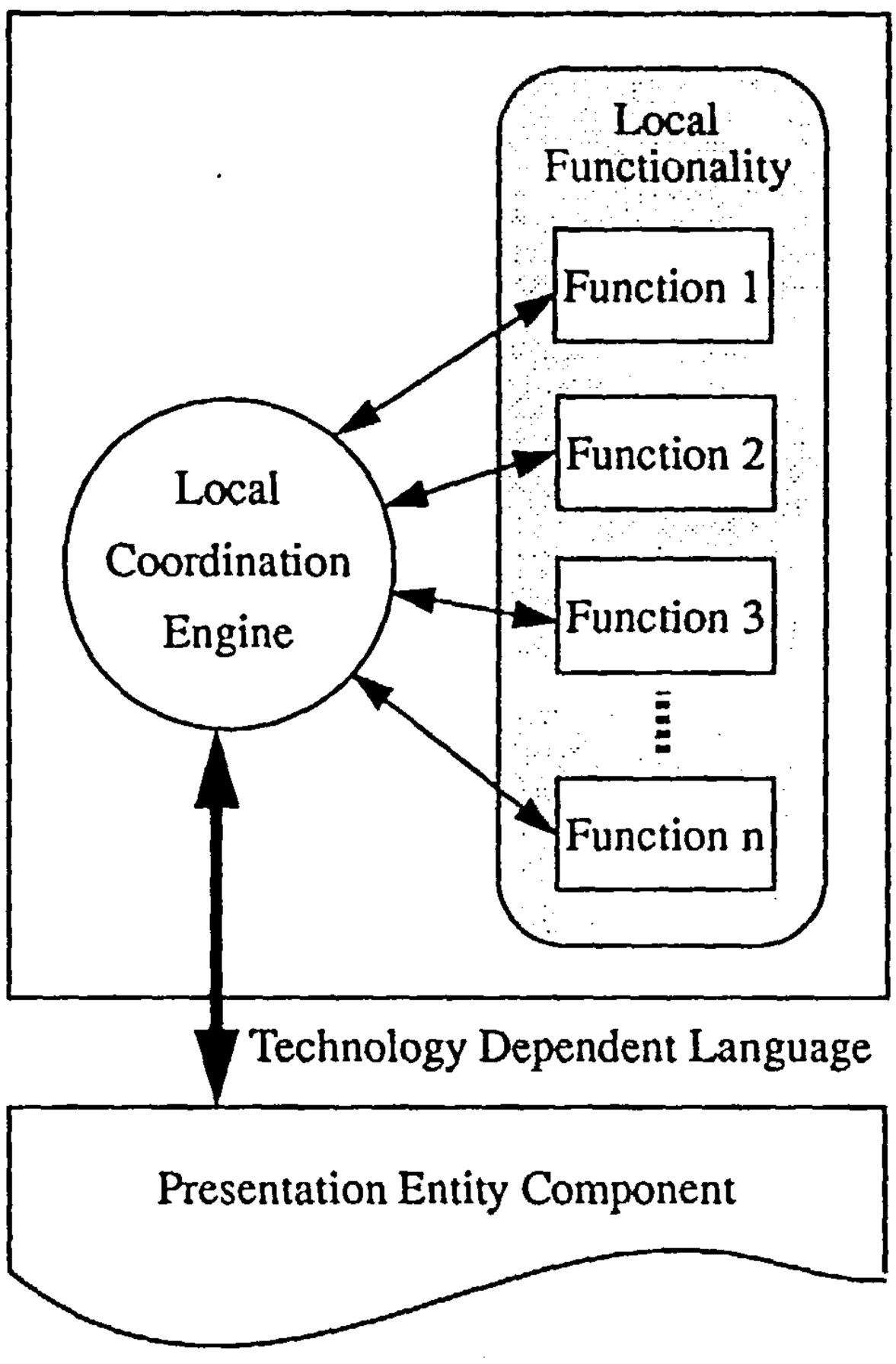


Figure 70 - Ideal functional entity

structure, the local coordination engine receives functional operations (or equivalent commands generated by PE components from a script), and selects appropriate functional building blocks to execute them. Likewise, relevant statuses must be encapsulated into functional operations and reported back to enterprise activities. Ideally, scripts should be created so as to provide null transformations or very simple ones. A null transformation implies that the resource component is able to directly recognise functional operations. Simple transformations mean nearly one-to-one mappings between functional operations and the commands recognised by resource components. Unfortunately, such an ideal situation requires a level of definition of functional operations that CIM-OSA has yet to provide. CIM-OSA partial models define types of functional operations which are not meant to be complete or closed. A considerable amount of work remains to be done in order to define functional operations that can be generally applied to all occurrences of resource component.

9.4.2. Proposed functional operations

Based on an enhancement of definitions provided in the latest frozen specification from the ESPRIT/AMICE consortium [ESPRIT/AMICE 1993a], Figures 71 and 72[1] show the set of functional operations adopted in this research for machines and human functional entities, respectively.

requests	responses
do(task)	**done**(task, status)
verify(part)	verified(part, status)
inspect(part)	inspected(part, status)
insert(part)	inserted(part, status)
store(part)	stored(part, status)
check(process)	checked(process, status)
validate(task)	validated(task, status)
set-up(process)	set(process, status)
monitor(process)	monitored(process, status)

Figure 71 - Proposed human functional operation types

The rationale underlying the definition of functional operations shown in Figures 71 and 72 is as follows:

(1) the use of generic operations for single operation machines [such as make(part) or move(part)] or operators [such as do(task)]. Here, the context in which the operation is requested partly determines the task to be executed;

(2) the use of specific operations for multi-operation resource components (e.g.

---

1. In these figures, bold functional operations consist of classes representing the functional operation instances below them (e.g. class "do" contains instances "verify", "inspect", etc.).

requests	responses
make(part)	**made**(part)
machine(part)	machined(part)
assemble(part)	assembled(part)
print(part)	printed(part)
wash(part)	washed(part)
reflow(part)	reflowed(part)
populate(part)	populated(part)
clean(part)	cleaned(part)
move(part)	**moved**(part)
push(part)	pushed(part)
flip(part)	flipped(part)
rotate(part)	rotated(part)
store(part)	stored(part)
control(device)	**report**(status)
stop(machine)	stopped(status)
start(machine)	started(status)
open(grip)	openned(status)
close(grip)	closed(status)
read(sensor)	value(value)
write(actuator)	written(status)

Figure 72 - Proposed machine functional operation types

print(PCB)), whereby a specific task needs to be identified;

(3) the association of a response to a functional operation (as appropriate);

(4) the indication of details associated with the manner in which a functional operation should be executed via appropriate functional operation parameters (e.g. make(part, part_number), move(part, part_number, from_A, to_B), or inspected(part, part_number, status).

This initial specification was proposed for the following reasons: (1) to define a 'first cut' set of operations that could be readily implemented, so as to allow the application of SEW-OSA; and (2) to test the means of interactions between SEW-OSA and the remaining tools and services produced within the "Model-Driven CIM project". The definition of truly generic functional operations, valid for certain domains in certain types of enterprises, depends upon the identification of reference models.

9.4.3. Set-up for the demonstration of physical integration

The identified functional operations[1] realise a means of interaction between certain components of the business entity (i.e. enterprise activities, the Activity

1. Functional operations exchanged between the business entity and the information entity are discussed in the next chapter.

Controller and the Resource Manager) and with active resource components through their associated PE components (as shown in Figure 73[1]). The structure depicted in Figure 73 was adopted in order to test some of the propositions presented in this chapter and to support the system execution stage. In this structure, execution of functional operations is emulated by a shop-floor animator (see Figure 73).

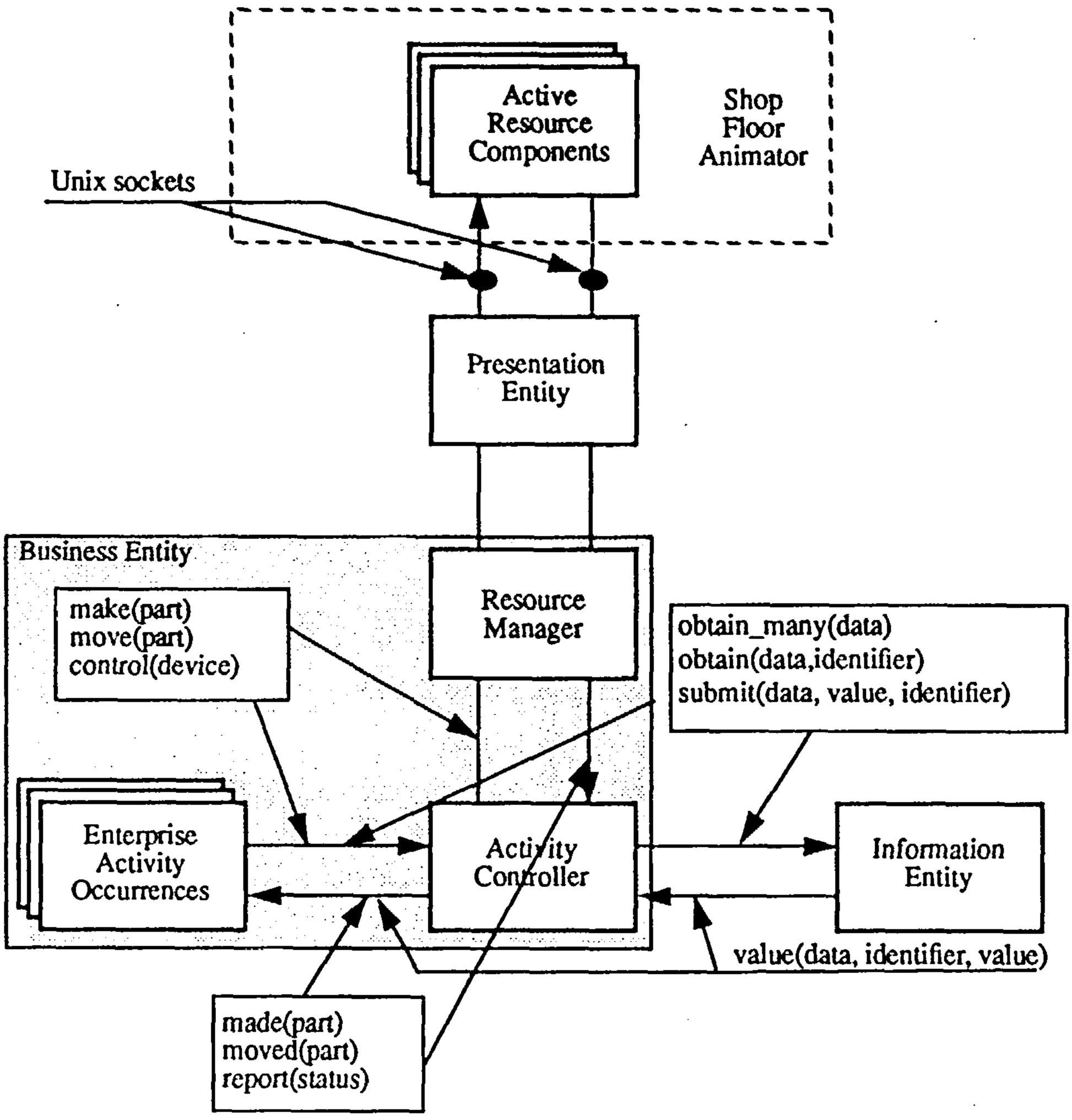


Figure 73 - Interactions between the SEW-OSA entities and the animator

The shop-floor animator is an application program developed in SmallTalk [ParcPlace 1990] by I. S. Murgatroyd (another researcher of the MSI Institute), which

---

1. The integration of the system configuration represented in this figure was a key result of a team effort involving all researchers working in the "Model-Driven CIM" project. Merits on the realisation of individual component are as follows: the "shop-floor animator" was realised by I. S. Murgatroyd, a simplified version of the "presentation entity" was realised by I. A. Coutts and the "information entity" was realised by P. Clements.

defines one SmallTalk object for each resource components of interest on the shop-floor. These objects:

- interface with the presentation entity via internet sockets shared with the PE components (see Figure 73);
- are able to execute some of the functional operations shown in Figures 71 and 72 directly, without the need for scripts (shown in Figure 66) associated with the PE components;
- interact with one another, in order to describe an animation of the physical actions that occur on the shop-floor (e.g. material movement and transformation) as a result of model-enactment.

The main difference between the configuration with emulated components and the one with the animator is that, for the emulated configuration, no movement of objects actually occurs, whereas in the case of the animated solution movements are represented as SmallTalk objects moving on a screen.

Hence, the primary function of the animator is to provide a tool which helps visualise model execution which should be particularly valuable to non-IT specialists. The animator provides a graphical interface which mimics model execution from the view-point of physical resources being used, this by highlighting material movement and information flow (relevant to state transitions in the system).

9.5. Limitations

It should be emphasised that the following limitations are associated with the activities underlying the description presented in this chapter:

a. Implementation

No implementation of device drivers, application enablers or human interfaces was undertaken within the scope of this research. Certain aspects of these issues have been investigated by other MSI researchers within the scope of the "Model-Driven CIM" project and others are to be tackled in forthcoming research work. Additionally, modelling of interactions between machine functional entities is one of the main issues addressed within the scope of the ESPRIT/VOICE project [Didic 1993] which this research did not attempt to duplicate.

b. Usability

Model-based specifications can only be experienced when resource models (which exist as a form of reference model, as illustrated in Figure 7) are available and reasonably well populated. Currently, SEW-OSA is populated with a sub-set of the

resource models required by the shop-floor of D2D. Such models include references to information about configuration and interfacing as required by a PE Component. Although such a level of population is sufficient to demonstrate the concepts proposed in this research, more completely populated models are required if SEW-OSA is to be broadly used for resource specification.

c. Interpretation of the business model

When moving from a rapid-prototyping stage (described in Chapter 7) to a system execution stage, possible modifications in the business model may be required. These modifications are required to adapt the model to particular situations in terms of component interactions that may change when tested with physical components rather than with emulated ones. The shape of the business model and the way in which it is executed should remain the same (i.e. the model continues to be executed in an interpreted form).

An option that may be considered in future versions of SEW-OSA is to replace the interpreted form of the business model by a compiled one (in "C" or "C++") for the sake of improved run-time performance. However, realising such an option is beyond the scope of this research.

d. Event generation

Relationships between an event and a happening in the physical system are defined in two forms, namely: (1) by associating an event occurrence with an action which can be captured by the presentation entity and channelled to the Event Handler (see Figure 52) - e.g. the arrival of a batch on the shop-floor, or the expiration of a time-out, etc.; (2) by defining triggering conditions to information object views held on data-bases, which are transformed into events when certain operations are executed upon the object views (e.g. creation, deletion, update, etc.). Although SEW-OSA provides means of easily incorporating these two forms of event generation, they have not been included in its current implementation.

9.6. Concluding Remarks

This chapter concentrated on analysing (albeit not proving) the impact of the CIM-OSA architecture upon the configuration and execution of a physical system. From this perspective, the main contributions made from this analysis are as follows:

- the definition of a generic structure for implementing the components of a presentation entity (as shown in Figure 66), this being required to facilitate interaction in an homogeneous manner with inherently heterogeneous resource components;

- specifying how the generic structure of a PE component can be used to handle interactions with three types of functional entities (i.e. application programs, machines and human beings);
- defining and implementing a suite of functional operation types, this at a certain level of granularity with respect to the atomic pieces of functionality handled by the business model;

In so doing, this research has examined the issues involved in evolving from a purely SEW-OSA-based system, in which the inherent heterogeneity of active resource components is abstracted through using emulated components, to a scenario in which instances of the three types of physical resource components are an integral part of the system.

Chapter 10 - SEW-OSA within the context of other "Model-Driven CIM" tools and models

Chapter 5 described the model-building capability packaged as the SEW-OSA CASE tool. However, a comprehensive capability for supporting model-building requires additional CASE tool elements, as illustrated in Figure 9. Although these additional elements are outside the scope of this Ph. D. study, they impact directly on it. Indeed, other MSI researchers working on the "Model-Driven CIM" project" (see Appendix 2) produced versions of such tools as indicated in Figure 74. These elements comprised of (see Figure 74): a resource modelling CASE tool based on the Booch object-oriented software design method [Booch 1991], produced by I. S. Murgatroyd [Murgatroyd 1993], CASE tools for modelling systems from a 'bottom-up' perspective, produced by I. S. Murgatroyd [Murgatroyd 1993] and P. Gilders [Gilders 1995] using the support of infrastructural services realised by I. A. Coutts [Coutts 1994] and a CASE tool for information modelling, produced by P. Clements and I. S. Murgatroyd [Clements 1993] which also uses infrastructural services realised by I. A. Coutts [Coutts 1994].

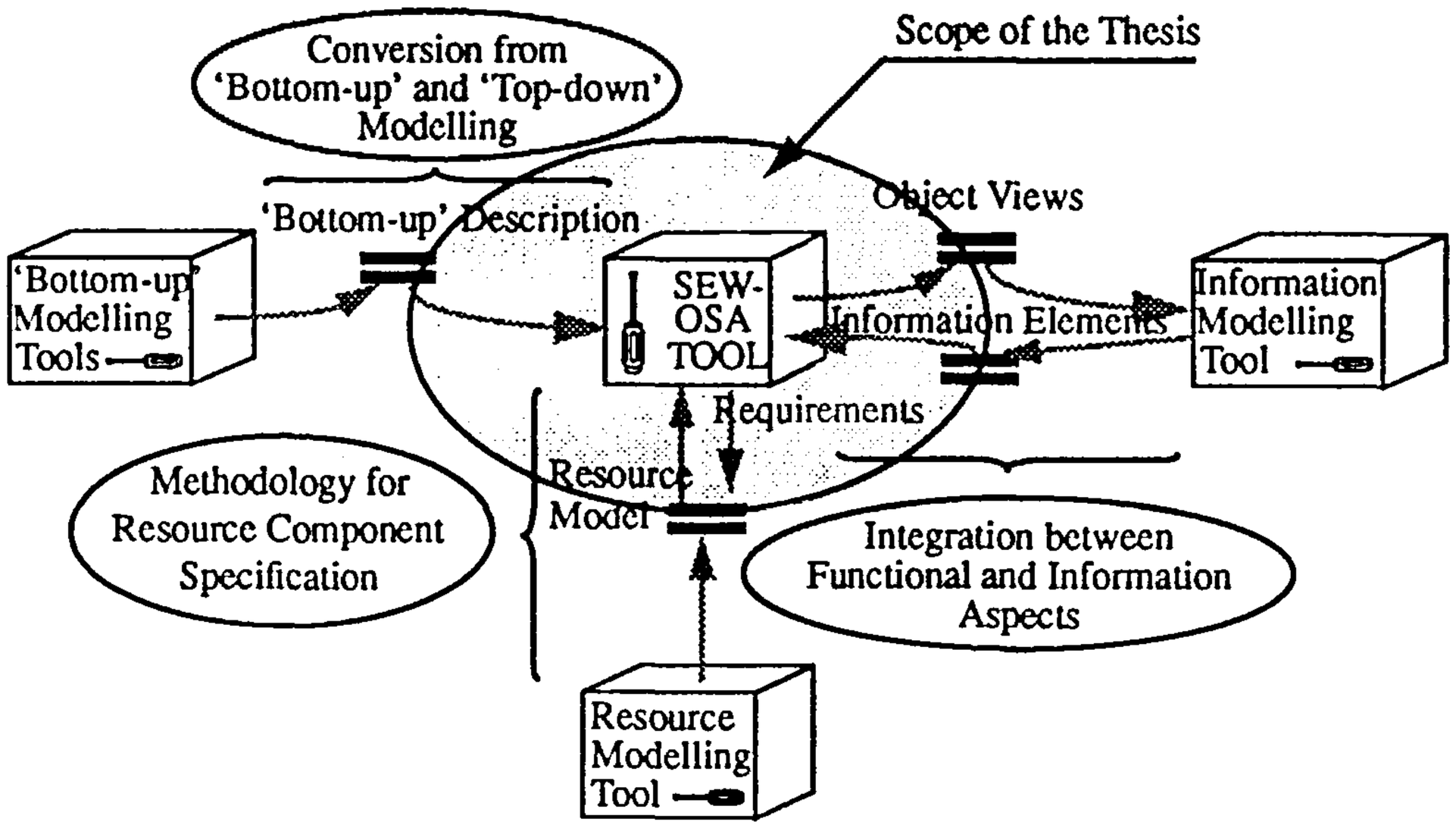


Figure 74 - Relationships among modelling tools

Indeed, many of the ideas presented in this chapter carry a strong contribution of conceptual thinking from these researchers which the author duely acknowledges. On the other hand, this chapter seeks to outline the author's view as to how such elements can be integrated with the model-building capability of SEW-OSA. Indeed, any inconsistencies or limitations in expressing such a view are the responsibility of the author.

At the time when initial writing of the thesis began, the author's view of a unified set of modelling tools had not been tested. However, during the writing up period, the task of integrating these modelling tools has been carried out as part of the ACME funded "Model-Driven CIM" project as a joint effort of MSI researchers (including the author). Hence, the discussion presented here is germane to this thesis as it clarifies the scope of SEW-OSA with respect to the "Model-Driven CIM" project (see Figure 74). It also describes potential uses of the concepts introduced in previous chapters (e.g. concerning the use of resource models, reference models and the need to separate models of functionality and behaviour).

Outstanding activities involved in achieving an integration of the modelling tools are depicted by the ellipses in Figure 74. These design and implementation activities include:

- identification and use of a methodology for resource component specification. This will provide an "interface" between SEW-OSA and a resource modelling tool;
- methods and mechanisms for integrating functional and information modelling perspectives, based on an "interface" between SEW-OSA and an information modelling tool;
- identification and use of an approach for mapping between 'bottom-up' descriptions of a system (which, to encourage re-use, should be an object-oriented-based model) and 'top-down descriptions of systems (i.e. CIM-OSA-based models), thus, realising an "interface" between SEW-OSA and a object-oriented modelling tool.

Possible solutions which meet the needs of the first two "interfaces" are discussed in the following sections. For reasons of conciseness, a proposal related to the third "interface" is presented in an internal report [Aguiar 1995a].

10.1. Methodology for Resource Component Specification

A key issue at the design specification modelling level is the need to specify system components, based on functional requirements identified at the requirements definition modelling level. The approach proposed here recognises certain characteristic properties of integrated manufacturing systems, which include[1]:

- relationships between resource components, which prove to be appropriate for addressing certain requirements, should be documented by the designer based on

1. It is important to notice that the approach taken is geared towards the definition of active resources as supposed to passive resources (see the resource diagram shown in Figure 33).

experience gained from carrying out previous designs. It is likely that models of resource components and their inter-relationships will be domain specific (i.e. they are likely to only be valid for certain types of businesses, certain areas of a business or certain classes of resource).

- when selecting a resource (within the context of the resource diagram shown in Figure 33), it is envisaged that support will be provided in a form which identifies candidate active resource components (which have the capabilities to meet requirements defined by the user).

Therefore, models of resource components should encapsulate the following information:

- attributes which classify them in a "catalogue of resources", i.e. as a resource model which is selected and used when creating resource and object diagrams (similar to those shown in Figures 30 and 33) to facilitate resource selection;
- a description of their internal dynamic behaviour, in response to requests from enterprise activities to execute functional operations on their behalf (i.e. information that is captured by entity behaviour diagrams similar to that of Figure 32, which is shared with the resource model, as depicted in Figure 75);
- a binding between model representations involving an instance of physical resource components and required means of accessing them (i.e. device drivers, application enablers, user interfaces, etc.). This binding should be inherited as part of resource selection operations performed when creating the resource diagram. This binding, to a physical instance of a resource component, should enable automatic replacement of an emulated component by a physical component; this as a key step towards progressing from design specification to implementation description in SEW-OSA. However, such an operation is only possible if the selected resource component and the means by which it is accessed are made available in some form of database. This implies that the resource should have previously been procured on an 'off-the-shelf' basis or developed 'in-house'.

In this context, the process of resource component selection will consist of:

- searching through a library of models of physical resource components, in order to select certain resource models from a suite of options;
- testing the options through simulation and rapid-prototyping of the system;
- analysing the results in terms of the impact of choosing certain options upon the overall performance of the system.

10.1.1. Resource model

To facilitate selection of the resource components of a system, a resource modelling capability needs to be incorporated within SEW-OSA. As part of the "Model-Driven CIM" project, parallel study [Murgatroyd 1993] had proven the potential use of an object-oriented CASE tool in respect of resource modelling. Hence it was decided that such a tool should be related to SEW-OSA, according to the diagram depicted in Figure 75. Indeed, the object-oriented design tool was produced in prototype form by I. S. Murgatroyd, based on the Booch methodology.

In the scheme shown in Figure 75, the information encapsulated by the resource model can be used as an integral part of the modelling method described in Chapter 5. The composition of the cloud labelled "requirements definition of SEW-OSA" (in Figure 75) is shown in greater detail in the upper part of Figure 22.

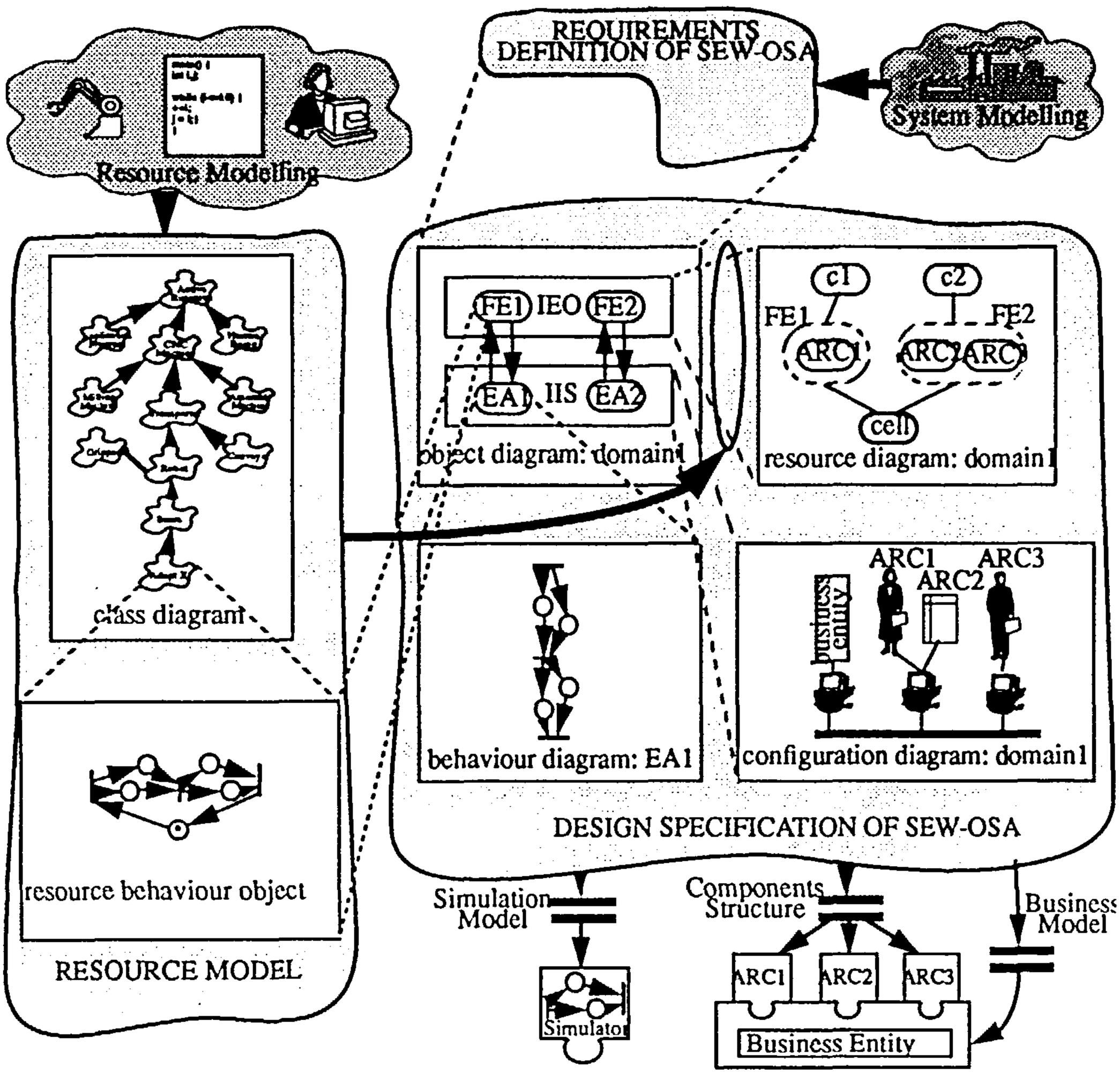


Figure 75 - Interface with a resource model

Essentially, a resource model organises data about currently available resources into a class diagram (according to the Booch methodology [Booch 1991]), as shown by Figure 76 (extracted from [Murgatroyd 1993]).

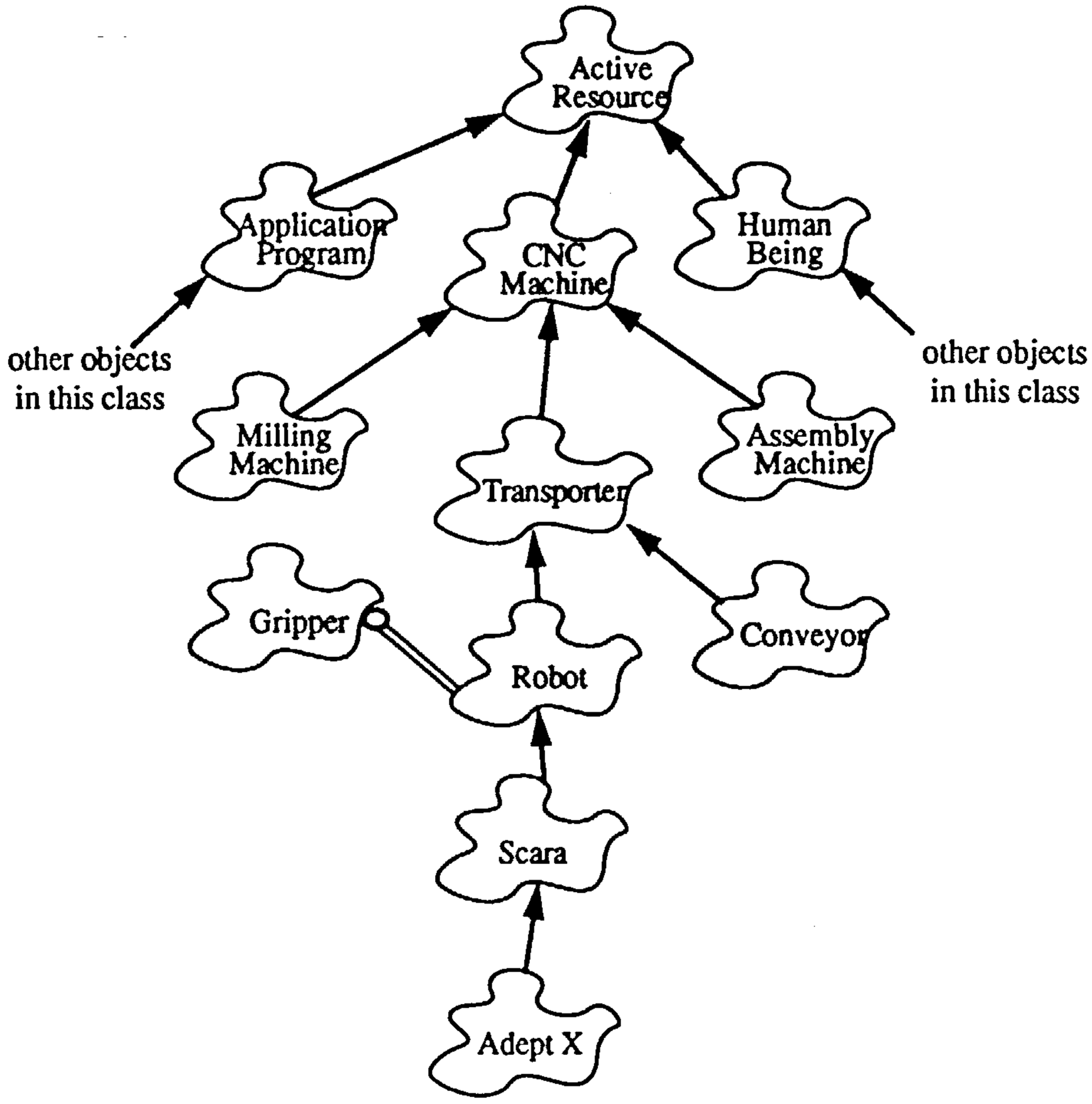


Figure 76 - Example of a resource model class hierarchy for a machine

Any new instance of a resource can be catalogued in this model by defining its place in the class hierarchy, its attributes and its expected internal behaviour (this by means of a Petri-net). As illustrated in Figure 76, the model should facilitate the cataloguing of new resources in the form of active resource components representing (but not limited to) machines, application programs and people (working in a given role). Attributes of resources are expected to embrace relevant data about their features, as well as defining means by which their integration can be achieved (upon an integrating infrastructure).

In this scheme, the internal behaviour of a resource can be represented by information captured in a entity behaviour diagram (similar to that of Figure 32). In the context of the model-building capability, this diagram is assumed to represent the behaviour of a functional entity (i.e. an entity behaviour diagram, as shown in Figure 22). Within the modelling process described in Chapter 5, this diagram is

assumed to be inherited by the SEW-OSA business model from the resource model, as illustrated in Figure 75. Subsequently, information contained in this diagram can be used to generate a prototype of the system.

The resource selection process starts with the definition of the resource capability construct used in the functional diagram (shown in Figure 29). An example of a resource capability template is shown in Figure 77[1] which is associated with the constructs identified as "RC" in the functional diagram. The resource capability construct must capture characteristic properties of a resource, described from the viewpoint of the user. One should notice that this template provides an abstract definition of the overall requirements of a resource without defining details of physical resource components used to realise such a capability.

RESOURCE CAPABILITY TEMPLATE:	
TYPE:	Manufacturing
IDENTIFIER:	RC-2
NAME:	Placement Capability
DESIGN AUTHORITY:	Marcos Aguiar
DESCRIPTION:	Capability to populate boards
CAPABILITY ATTRIBUTES:	
Activity:	Active
Type of Activity:	mechanical and repetitive

Figure 77 - Example of a resource capability template

As one progresses to the design specification stage in Figure 22, resource components are expected to be defined with respect to representations of their functional (i.e. functional entity) and physical (i.e. active resource components) properties. A functional entity construct can be used to represent the class of interactions that a resource should support (this being related to the types of functional operations that the resource can provide). Figure 78 shows a typical functional entity template which was conceived by the author to summarise its attributes of interest.

---

1. Construct templates presented in this chapter have not yet been incorporated into SEW-OSA in a form which can enable integration with a resource model. Therefore, their current attributes should be viewed as the author's recommendation with regard to their definition.

Here, it should be noticed that its attributes define, the type of functional operations supported, a more specific definition of its type (e.g. a machine) and the internal behaviour that it is expected to demonstrate (e.g. a link to an entity behaviour diagram identified as FE-2_Behaviour_diagram).

FUNCTIONAL ENTITY TEMPLATE:	
TYPE:	Manufacturing
IDENTIFIER:	FE-2
NAME:	Placement Entity
DESIGN AUTHORITY:	Marcos Aguiar
DESCRIPTION:	Functionality to populate boards
ATTRIBUTES:	
Functional Operations:	make(PCB) made(PCB)
Behaviour diagram:	FE-2_Behaviour_diagram
Type of Resource:	Machine

Figure 78 - Example of a functional entity template

The model of an active resource component needs to include a description of the information required by the integrating infrastructure to interact with the resource. Here, it was realised that an active resource component can either be represented in an emulated form or in its final physical form. An example of such a description conceived for this purpose is presented in Figure 79. It should be noted that the attributes associated with the "integration features" of such a resource define: a link to a device driver which can be used by CIM-BIOSYS to enable communication with a particular machine (e.g. SMT_dvc); any object-views (e.g. OV) required for the integration to occur (e.g. in this case, "Siemens_integ" identifies information elements that need to be used to accomplish the integration); the name of the CIM-BIOSYS process which manages the integration (e.g. SIEMENS); the host computer to which the machine is attached (e.g. derek), the type of hardware interface required by the machine (e.g. RS-232), and a script which enables it to be integrated into SEW-OSA. Arguably, such a template defines all necessary attributes to realise the integration of a resource into a system.

ACTIVE RESOURCE COMPONENT TEMPLATE:	
TYPE:	Manufacturing
IDENTIFIER:	ARC-3
NAME:	Siemens Placement Machine
DESIGN AUTHORITY:	Marcos Aguiar
DESCRIPTION:	Placement machine, make Siemens, model SM-456, number 4579-49386
MODEL FEATURES	
Behaviour Model	ARC-3_Behaviour_diagram
INTEGRATION FEATURES	
Device Driver:	SMT_dvc
Integration OV:	Siemens_integ
CIM-BIOSYS Process:	SIEMENS
Usual Host	Derek
Hardware Interface:	RS-232
Script:	Siemens_script

Figure 79 - Example of an active resource component template

These three templates which define the resource capability, the functional entity and the active resource component are being incorporated into the model-building capability of SEW-OSA. Indeed, current effort in MSI is in establishing a link between SEW-OSA and a resource modelling capability, in order to provide a means of gradually refining the specification of a resource through the use of information provided by these templates. Currently, SEW-OSA supports system performance tests based on a selection of alternative resources. However, it does not support the process of refining the specification of a resource, so that a physical instance of a resource can be defined from its requirements.

In this research, it is envisaged that this level of support should be encapsulated in the resource model (i.e. within a reference model of resource components). Structuring and populating such reference models is beyond the scope of this research. However, this research has identified necessary constituents of these models, namely:

- a library of resource models with attributes corresponding to the templates shown in Figures 77 to 79;
- means of creating relationships between these models so that the experience acquired in previous designs, with regard to the process of refining a resource specification, can be formally captured in models (i.e. relationships between requirements definition and design specification). These relationships are referred to in Figure 36 as "connectance models"[1].

The resource specification process consists of establishing relationships between attributes associated with a resource capability and associated functional entities and active resource components. In this manner, alternative specifications of resources retrieved from the resource model can be input automatically to the SEW-OSA CASE tool. The relationships represented in the connectance models are likely to possess many-to-many cardinality (i.e. there may be many active resource components capable of providing a certain resource capability and vice-versa). Hence, the modelling process associated with the selection of resource components is envisaged to embrace the following steps:

- definition of resource capabilities (at the requirements definition modelling level) attached to each enterprise activity (via the functional diagram shown in Figure 29);
- retrieving of representations of resources in the form of functional entities and active resource components (at the design specification modelling level), which encapsulate the internal behaviour of the resource, and form a basis for refining the information required to facilitate appropriate selection of resources from the class hierarchy (illustrated in Figure 76). It is anticipated that such a refinement process will be carried out partially by defining both object and resource diagrams (as shown in Figure 75) of the SEW-OSA CASE tool. From the viewpoint of SEW-OSA, such a process consists of obtaining a static model of a candidate solution (e.g. the list of attributes of the type shown in Figures 78 and 77), as well as a dynamic model of its behaviour based on the use of Petri-nets.
- testing of dynamic aspects of the resource through simulation and rapid prototyping. Some of the dynamic issues involved are associated with the metrics

---

1. According to Kwikkers [Kwikkers 1992], connectance models describe connections in a particular field of knowledge. Connectance models consist of generalisations of point solutions encountered in particular designs. Kwikkers uses connectance models to retrieve a particular model from a library of reference models. Work on the definition of connectance models is being carried out at Karlsruhe University [Naeger 1993], with which the MSI institute has collaborative research links.

embodied in the Petri-net model. Additionally, simulation runs are required, for a resource may serve many occurrences of enterprise activities with which it is associated. Hence, there may occur resource utilisation constraints that have to be considered when multiple occurrences of enterprise activities compete for the same resource component. Timing considerations associated with the way functional operations are executed will strongly impact on these dynamic issues;

- review of the specification and re-execution of previous steps, until a satisfactory solution is achieved. There may be more than one resource to address the same set of requirements. In this case, simulation and rapid-prototyping can help in assessing performance considerations which, weighted against the cost of alternative resources, provide the basis for the decision making with regard to resource selection;

- replacing a resource model by the actual physical resource (at the implementation description modelling level) on a component-by-component basis. Such a replacement should be facilitated by the definition of integration aspects already embodied in the modelled resource components (i.e. attributes shown in Figure 79).

It should be noted that the proposed methodology validates a resource selection based on its performance from an integration perspective. That is, it should test whether timing and interaction considerations amongst system components are being performed according to what is expected. However, this methodology does not provide any means of testing whether the functionality provided by a resource effectively addresses the needs identified in the modelling process.

10.1.2. Resource specification considerations

It may be unreasonable to expect that a designer would have available (in a local library) instances of all marketed implementations of resource components that he may require for his design. However, it may not be so unreasonable to envisage that he could have models describing most of these resources to the level of detail that he requires in order to make a decision regarding selection of a particular resource. Essentially this is a primary reason why models can play an important role in the process of selecting a physical resource.

Such an approach also implies a more forward looking view of the necessary provision of information by suppliers about their resource components. In such a scheme, suppliers would be required to provide testable models of the resource components they supply, rather than merely catalogues with data about features and behaviour of resource components which cannot be tested. Modules of resource components associated with such models are expected to be available in the market on

an 'off-the-shelf' basis (e.g. software objects generated via the OMG initiative). Modularity, in this case, implies a twofold requirement, namely: provision of well established interfaces between the modules that comprise a resource component and its external environment; and provision of models that describe the form in which the resource component behaves internally.

10.1.3. The importance of reference models

In previous sections, the way in which SEW-OSA is used to support resource component selection is described. The approach is based on the selection of resource models which enable a link to be established between activities and models created during the conceptual analysis and design and implementation phases of the IMS life cycle (this being a requirement on this research study as highlighted in Section 3.2.4).

Resource models associated with system models (i.e. models of requirements) constitute the reference models required to support activities of these life cycle phases (as shown in Figure 7). At the interface between these two phases, reference models produced under the umbrella of an architecture play an important role in bringing together: (a) the requirements associated with a particular problem domain (i.e. the world of IT usage), (b) the solutions that can be used to address those requirements (i.e. the world of IT supply), and (c) the means by which solutions can be put to work (i.e. the world of integrating infrastructures). Such a role for a reference model is illustrated in Figure 80.

As an integral part of the design and build process, reference models should be used as a reference for decisions upon: (1) how to describe the requirements of a certain domain (through using system models) and (2) what resources to select in order to address those requirements (through resource models). This combined use of system models and resource models is illustrated in Figure 81.

Although this section has concentrated discussion on resource models, it is envisaged that a similar approach can be used to define a library of system models. Such a library is expected to contain partially instantiated models of enterprise activities, business processes, domain processes, etc., as well as models of complete domains of certain types of enterprises (described at various levels of abstraction).

As represented pictorially in Figure 81, libraries of system models could be used to facilitate a description of requirements for a particular enterprise (i.e. a particular domain). Once requirements are captured, a library of resource models could be used to specify a particular selection of resource components configured to address those requirements. Notice in Figure 81, once again, the envisaged role of connectance models in linking a refined form of requirements to a solution in terms of resource specifications. Assuming that this process occurs under the support of a workbench (such as SEW-OSA), a selected system configuration can be tested and put to work

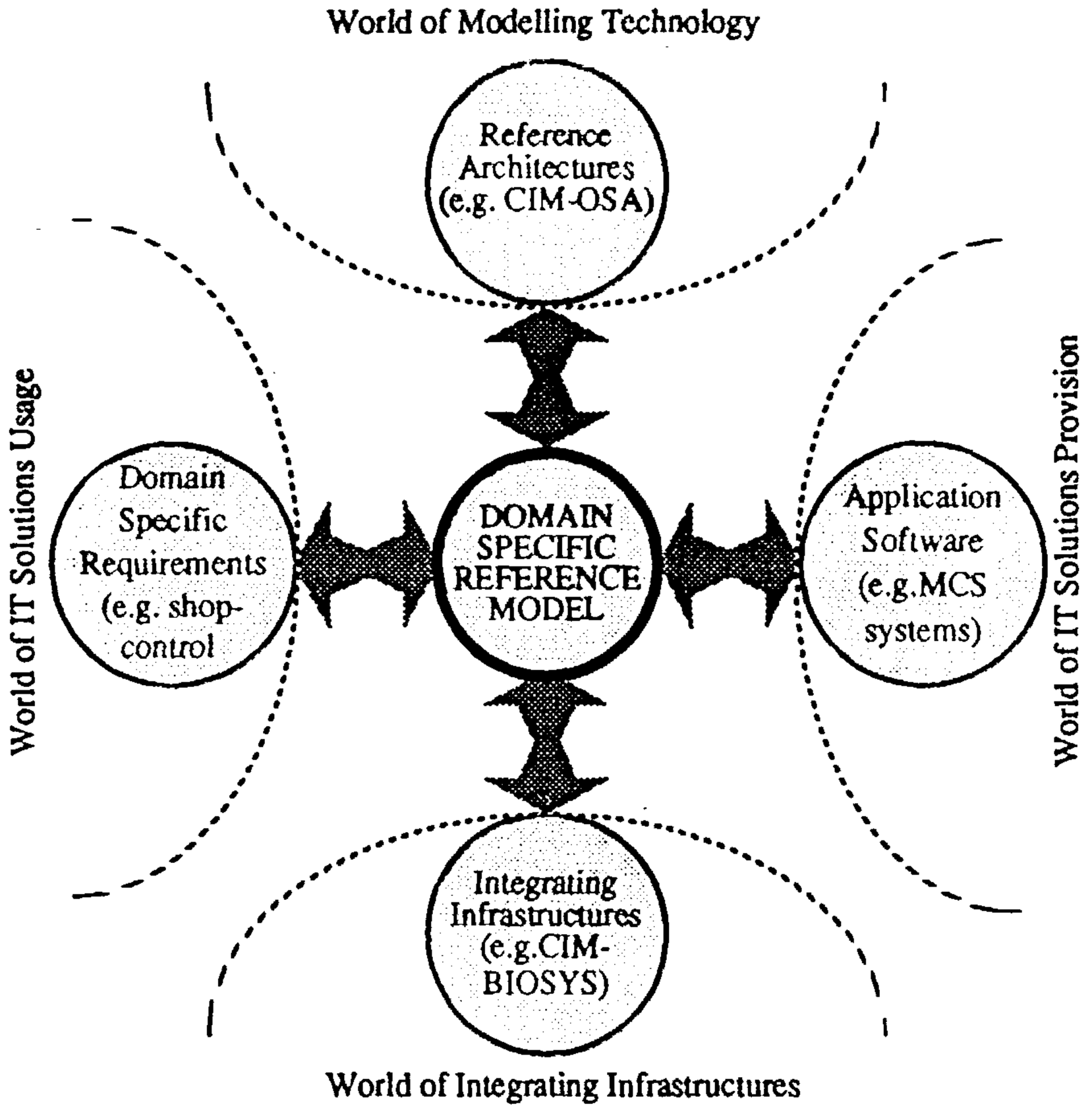


Figure 80 - A key role for reference models

automatically via simulation and rapid-prototyping.

This move from requirements definition to design specification characterises a 'top-down' approach to modelling (i.e. "forward engineering" in Figure 81). However, in some cases, an existing system needs to be changed and re-engineered (i.e. "reverse engineering" in Figure 81). This may require a 'bottom-up' approach to modelling (which is also foreseen in Figure 74).

10.2. Integration Between Function and Information

A crucial limitation of contemporary modelling methods and tools is a lack of integration between two major modelling views of systems, namely: function and information [Mayer 1992] [Coad 1990]. This research does not aim to resolve such a limitation in a generalised manner, but it does aim to provide facilities for realising a link between SEW-OSA functional modelling and an information modelling tool. This section briefly describes such a link and relates this to the model-building capability of SEW-OSA.

Central constructs of SEW-OSA from which relationships between function

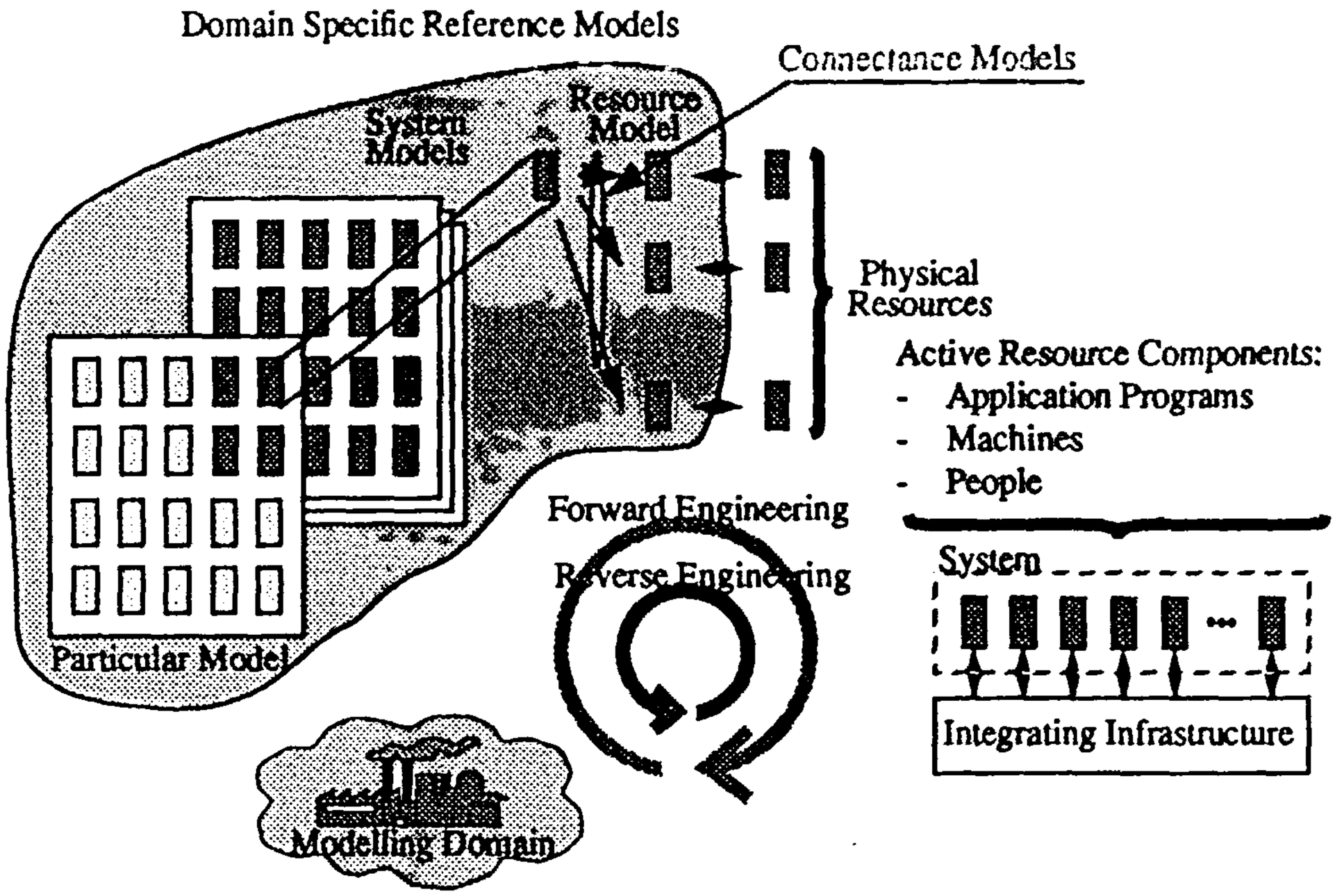


Figure 81 - Reference models applied to the design and build process

and information views can be defined are the *enterprise activity* and the *functional operation*. Based on these relationships, an integration process can be achieved according to the methodology illustrated by Figure 82[1], which summarises the main design and build activities performed at the interface between these views.

From a top-down stand-point, the integration process embedded in SEW-OSA can be realised by the following method:

a. Once the requirements definition model is completed (by use of the method described in Chapter 5) a list of identified object views is generated by the SEW-OSA CASE tool (i.e. the function view). These object views are related to objects manipulated within an information modelling tool (i.e. the information view). An example list of such object views is shown in Figure 83. These object views are mostly identified as part of design definitions encapsulated by the functional diagram (shown in Figure 29), which identifies enterprise activities as consumers and producers of object views.

b. With the help of an information modelling tool, the identified object views are analysed and related to information elements (i.e. atomic pieces of data), as part of the activities supported by the tool at the requirements definition modelling level.

---

1. This figure was developed in a joint effort which involved I. A. Coutts, I. S. Murgatroyd, P. Clements, P. Gilders and the author.

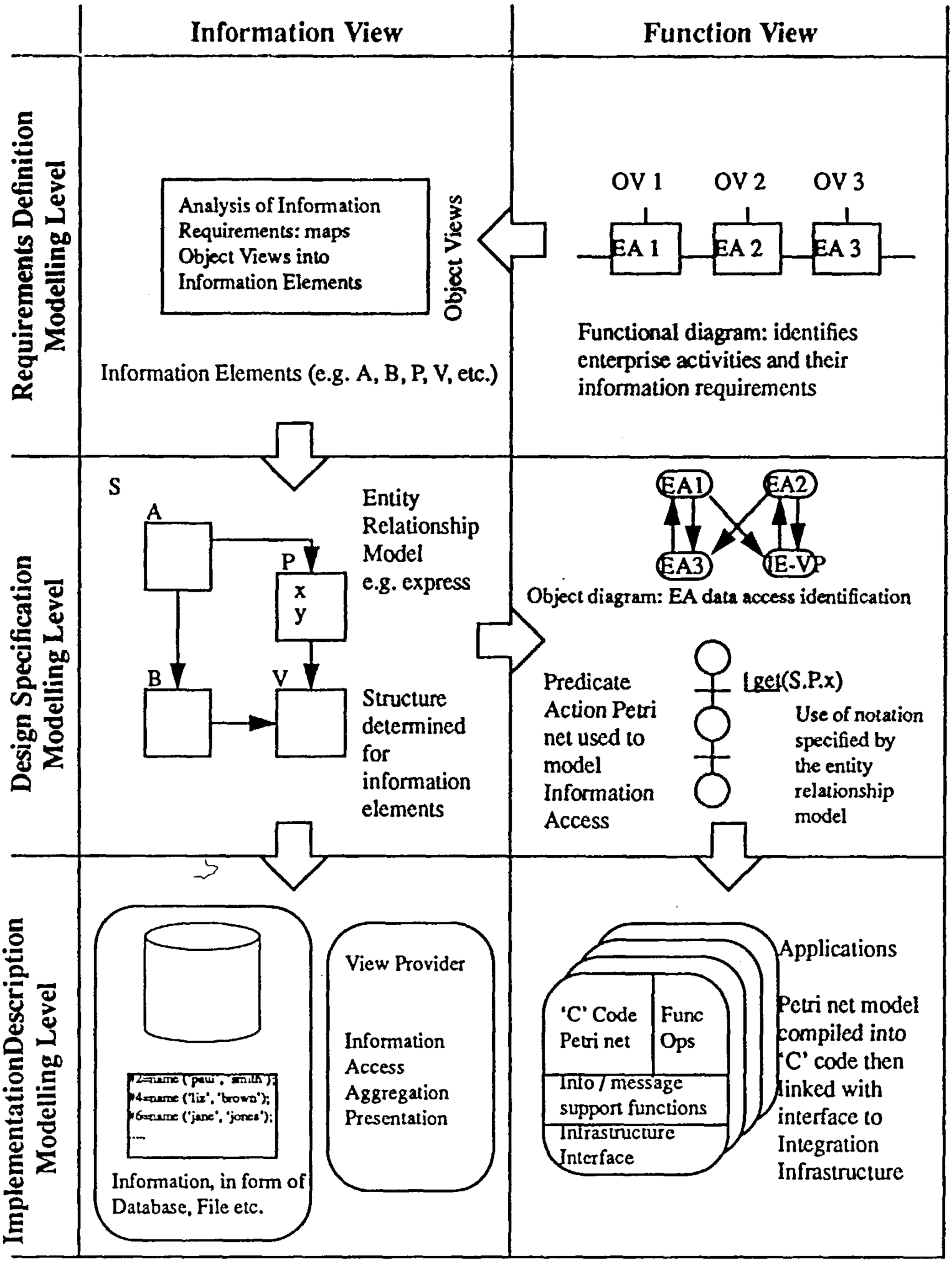


Figure 82 - Relationships between function and information views

Here, it is important to note that an object view (as its name implies) is simply a view into a set of information elements. Therefore, an object view can be mapped into one or a combination of information elements.

c. The information elements can then be mapped into entities or attributes, identified

CIM-OSA Design: smt

List of Object Views:

OV-1/Batch BB/P
OV-2/PCB/P
OV-3/released BB/P
OV-4/ready to print/P
OV-5/printed/P
OV-6/populated/P
OV-7/checked/P
OV-8/ready to insp/P
OV-9/inspected/P
OV-10/finished/P
OV-11/ready to reflow/P

Figure 83 - Example of a list of object views

and structured through the use of an entity-relationship-attribute (ERA) diagram. This diagram is used to formalise the conceptual design of a data-base structure which will hold the information element. This is the central activity supported by the information modelling tool, at the design specification modelling level.

As a result of this modelling step, a list of parameters (which can be used to construct transactions to provide access to these information elements) is supplied as input to SEW-OSA (at the design specification modelling level) by the information modelling tool (see Figure 82).

d. By referring to such a list, functional operations can be defined which can be used by enterprise activities to access information elements. These information elements are related to the object views that are consumed and produced by enterprise activities, as defined at the requirements definition modelling level.

Functional operations of the type "data transaction" are modelled in the object diagram by means of message exchange. Figure 30 provides an example of transactions between enterprise activities and the information entity. The definition of parameters of these functional operations, as well as the processing performed upon information elements retrieved through their use, are modelled in activity behaviour diagrams.

Data processing internal to an enterprise activity object can either be through direct manipulation of the values held by the information elements or through passing such values as parameters to functional operations. Functional operations, in turn,

can execute some defined task on behalf of the enterprise activity (e.g. a functional operation representing a collection of "C" functions which require input parameters). The direct processing can be by means of checking conditions, defined by the predicate part of the Petri-net, or by using the value in operations, defined in the action part of the Petri-net. Examples of both cases are shown in Figure 31.

The functional operations implemented in SEW-OSA for data transactions are shown in Figure 84, where the parameter "identifier" relates to a thread of business model execution.

```
obtain_many(data)
obtain(data,identifier)
submit(data, value, identifier)
value(data, identifier, value)
```

Figure 84 - Data transaction functional operations implemented

e. The structure defined in an entity-relationship diagram (as referred to in item "c") is then used to generate code in the Express syntax [Boyle 1991] which, in turn, is used to populate the data-bases where the related information elements will be maintained [Clements 1993].

   Here, one should note that these data can be distributed among a number of heterogeneous data-bases. The information services of CIM-BIOSYS enable transparent access these data, allowing simple reference to them by their names [Coutts 1994].

f. At run time, the transactions defined at the design specification modelling level are executed by the model-enactment capability of SEW-OSA which passes them to the CIM-BIOSYS integrating infrastructure, where they are transformed into appropriate data-base transactions (e.g. SQL statements). This is accomplished through the CIM-BIOSYS view provision facility, which provides the information element to the enterprise activities in the format required; this being independent of details about the location and format of the data [Coutts 1994].

As illustrated in Figure 74, integration between SEW-OSA and the information modelling tools has been implemented in the form of files which transfer a list of object views from SEW-OSA to the information modelling tool and models of the information structure from the information modelling tool to SEW-OSA.

10.3. Limitations

A limitation common to the three links discussed in this chapter consists of the lack of a tight integration[1] between SEW-OSA and these tools. Currently, they constitute completely separate CASE tools which can share data through transferring files with the data in it. A closer integration between these tools would facilitate consistency checking across the complete model (i.e. embracing function, resource and information views). In this respect, consideration must be given to efforts to standardise interfaces and reference models related to the inter-operation of tools (e.g. "Portable Common Tools Environment (PCTE), CASE Data Interchange Format (CDIF) and 'de-facto' standards, namely: HP Softbench, Microsoft Data Exchange (DDE) and Object Linking and Embedding (OLE) [Brown 1993]).

Other limitations associated with the proposals for each link concerns the following factors:

a. Methodology for Resource Specification

Here, a limitation is related to the fact that as yet only a limited amount of information about different types of resources and their requirements has been formalised in the CASE tools. In other words, not enough reference models are currently available, so that full advantage can be taken of the method proposed.

b. Integration Between Function and Information

Here, limitations of time and resources have constrained the implementation of only part of the CIM-OSA specifications, leading to:

- no support being given to formal analysis of information requirements at the requirements definition modelling level (see Figure 82);
- the transactions implemented at the design specification modelling level were of a simple nature (see Figure 84), namely: read and write operations.

10.4. Concluding Remarks and Contributions

The main contributions to knowledge made by conceiving and developing the methods and tools described in this chapter, include:

a. Methodology for resource specification

This methodology provides an essential link between the "conceptual analysis"

---

1. The term "tight integration" is used to indicate a level of interaction between the tools which would allow exchange of data and, hence, the propagation of model changes between their different modelling perspectives.

and "design and implementation" phases of the IMS life cycle. Indeed, the approach proposed and partially implemented in this chapter helps clarify how reference models (in the form of resource models and system models) can be used to aid associated integration processes.

b. Integration Between Function and Information

Through using the method described in this thesis, a useful level of integration between functional and information modelling can be achieved. Although "tighter integration" could not be provided, a working solution was realised which provides an initial level of support to systems design and modelling activities.

Chapter 11 - Analysis of Results

Previous chapters described the conception and development of SEW-OSA and how it was applied. This chapter describes and discusses results obtained with the purpose of evaluating various properties of SEW-OSA. It also seeks to provide a unified view of the deliverables obtained from this research. These results are classified, respectively, as "case study results", "implementation results" (in terms of material deliverables) and architectural results (as initially classified in Figure 8)[1].

11.1. Case Study Results

This includes results obtained from case study work when engineering an integrated shop-floor system for D2D. These results also constitute an evaluation of SEW-OSA with regard to the following issues:

(1) **Application.** This concerns the types of analysis supported by SEW-OSA during modelling and simulation stages. It includes an exploratory analysis of improvements in terms of integration and coordination.

(2) **Workbench Performance.** This centres on an analysis of the performance of the workbench, when supporting model-building and model-enactment. Prime focus is on the overhead imposed by the workbench during model-enactment.

(3) **Engineering Process.** This concerns the benefits that can be obtained when using SEW-OSA to support the re-engineering of certain domains.

The case study work provided an opportunity to benchmark the industrial application of SEW-OSA during various phases of the engineering cycle, i.e. from modelling through to analysis, simulation and rapid prototyping. As discussed in Chapter 8, two sets of models were created corresponding to two different levels of abstraction, namely: (1) models that represent the complete D2D shop-floor; and (2) models of a single D2D SMT assembly line (i.e. "line segment domain"). The engineering process associated with creating the first set of models was limited to using the model-building capability of SEW-OSA. This limitation stemmed from an implementation constraint imposed by the current version of SEW-OSA, which limited the maximum size of models that can be manipulated at simulation and rapid-prototyping stages. As later identified, these limitations centred on implementation issues related to the analysis and simulation tool (i.e. ARP) and the CIM-BIOSYS infrastructure. This will be exemplified later in this chapter.

---

1. The "research findings" shown in this figure are discussed in the next chapter.

Conversely, the second set of models was evaluated through analysis, simulation and rapid-prototyping phases, thereby providing a way of evaluating SEW-OSA as a whole.

11.1.1. The Model-building stage

The SEW-OSA cycle of model-building and code generation was tested following its use when producing three sets of models. The first two sets are mentioned above whereas the third was a simplified representation of the production process of the MSI Research Institute. These modelling exercises provided data on relationships between: (1) the size of a model (in terms of number of constructs) and the amount of code generated for the purpose of simulation and rapid-prototyping and (2) the size of a model and the lead-time involved in the model-building to code generation cycle. Figure 85 illustrates these relationships for each of the three models. The meaning attached to the information represented in Figure 85 (i.e. number of constructs in the model, code generated during simulation, code generated during rapid-prototyping and the model-building to code generation lead-time) is described in greater detail in Appendix 10.

As one might have expected, the plots in Figure 85 show that as the model size is increased, there is a linear correspondence between the amount of code generated and the lead-time involved in creating the model. Arguably, the plots can then be extrapolated and used in other applications to estimate the effort involved in model formalisation during the model-building process, this based on an estimate of the complexity of the models concerned.

As discussed later in this chapter, the size and complexity of the code generated from a model also directly impacts on the overhead faced when it is enacted. This can provide an indirect estimate of system performance before implementation decisions are taken. Such indications could be very valuable in support of investment decisions.

11.1.2. Simulation

Models that emerged from the modelling stage were used during subsequent analysis and simulation stages. The case study was limited to use models of one of the SMT assembly lines (i.e. a "line segment domain", as explained in the previous chapter). Analysis and simulation studies of the SMT assembly line concentrated on system performance issues related to a series of variables associated with the configuration and operation of the line. Figure 86 provides a pictorial representation of these variables represented in the bubbles (which are associated with the various operational stages of the SMT assembly line).

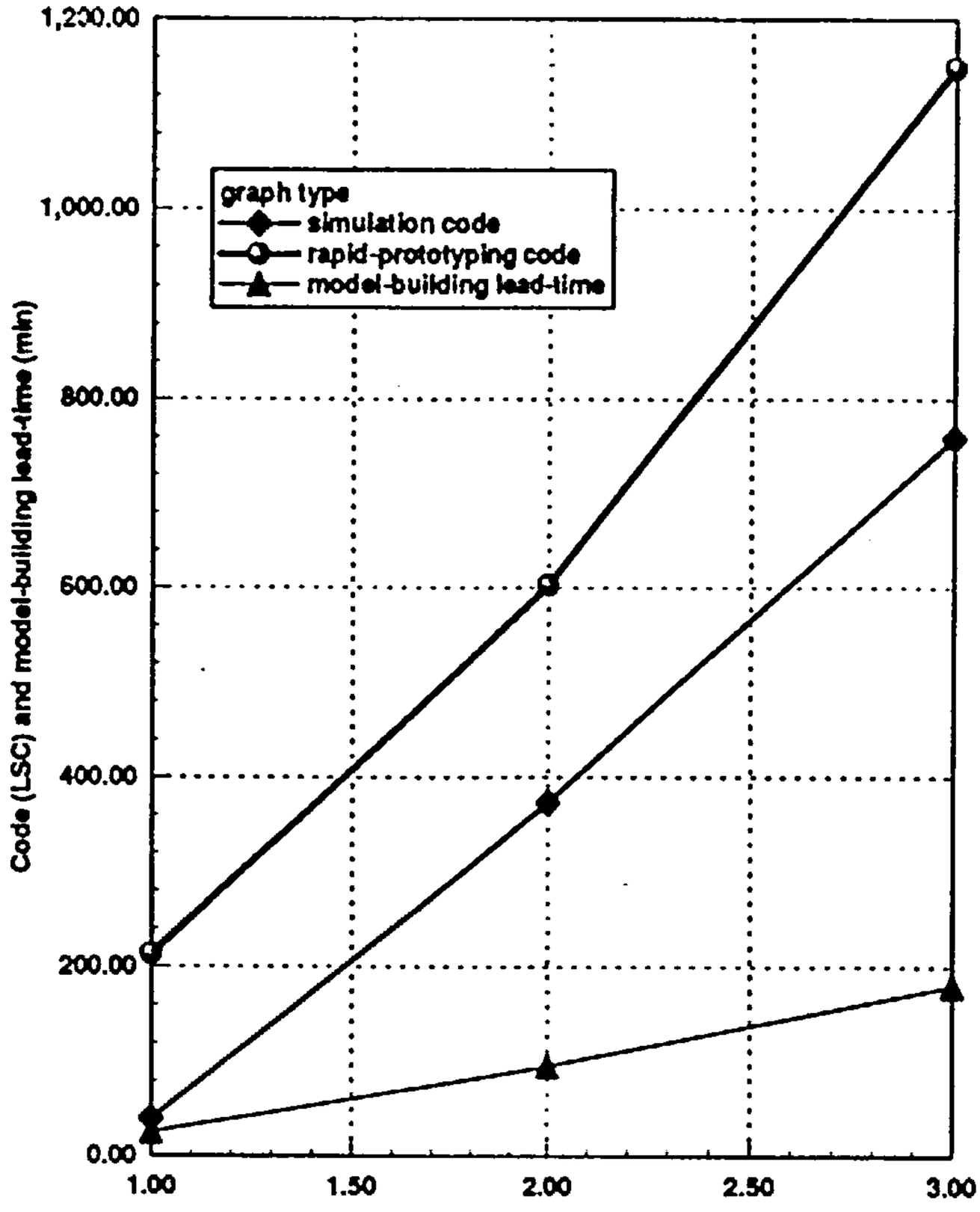


Figure 85 - Model-building process and model sizes (model)

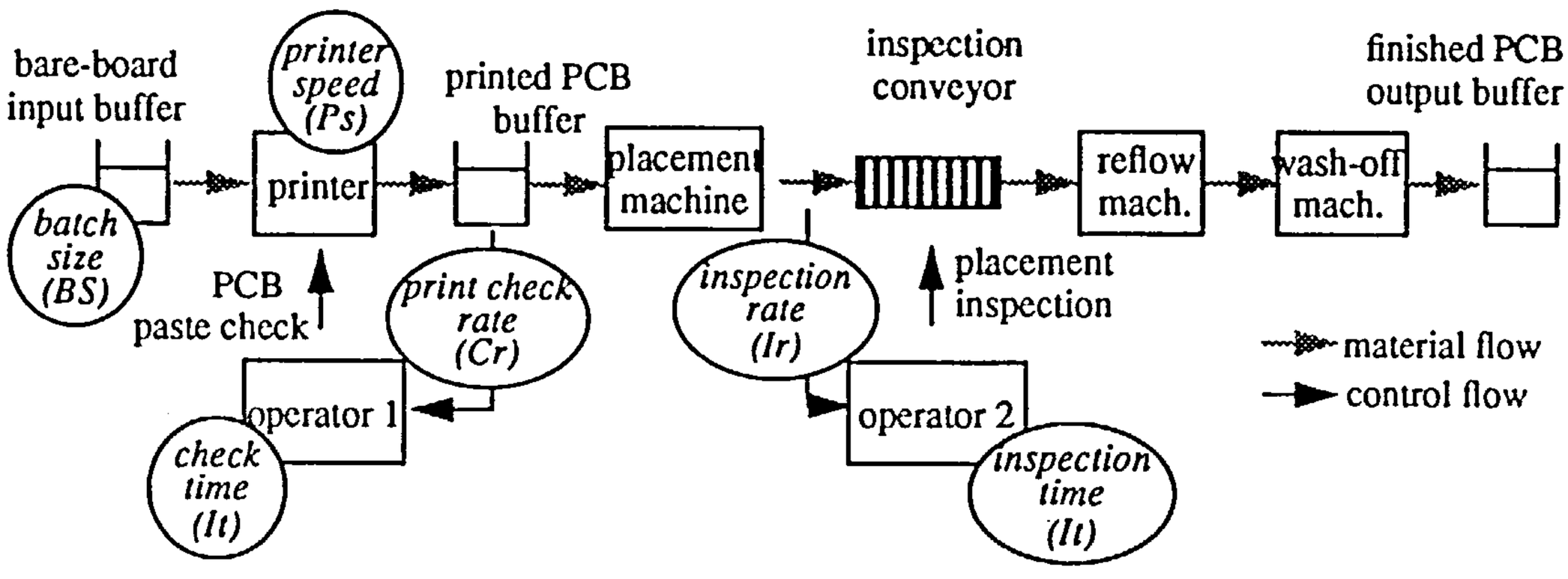


Figure 86 - Key variables associated with the SMT assembly Line

a. Simulation data

The SMT assembly line configuration depicted in Figure 86 was modelled using the parameters and assumptions listed in Table 5, thereby assigning values to each process and activity modelled. The parameters and assumptions were based on time and motion data supplied by D2D. Here, lower and upper limits were placed on time intervals associated with each process operation and a probability distribution was associated with the interval. The way in which parameters are assigned (to the functional operations identified in the models) was discussed in the previous chapters.

In some simulation studies, certain of the parameters were treated as independent variables (i.e. variables shown in Figure 86). However, where this was not the case, default values were chosen according to Table 5. It may be deduced from this table that: (1) the values correspond to a relatively simple PCB; and (2) "set-up" time does not include time to carry out "kitting" operations[1].

Table 5 - Parameters associated with the SMT assembly line

Operation	minimum time (s)	maximum time (s)	probability distribution
set-up	30	60	exponential
move to printer	1	2	exponential
print	18	22	normal
check of paste	60	120	normal
placement	64	72	normal
move into conveyor	3	4	exponential
move out of conveyor	3	4	exponential
inspection	160	380	normal
finishing	50	60	normal

Other assumptions made and conditions assumed for the simulation study include:

- a batch size of fifty PCB's. A small batch size (in respect to D2D's practice) was adopted to limit the time consumed during simulation.
- a checking rate at the printing process (i.e. the parameter **Cr** in Figure 86) of twenty percent (i.e. one board in five is checked), this being the default rate used at D2D.
- an inspection rate (i.e.the parameter **Ir** in Figure 86) of ten percent, this also being the default rate used at D2D.
- a percentage of defective boards of zero, as encountered at the "check paste" and

---

1. These can be the most time consuming operation, but are only of interest when job sequences are examined, such as when aiming to minimise change-over times.

"inspection" points.

- the existence of 6 slots on the conveyor which takes boards through the reflow and wash-off machines; 3 slots on the inspection conveyor, 3 slots inside the placement machine and buffers (i.e. input, output and print buffers, as depicted in Figure 86). A study of the impact of different slot values upon manufacturing (i.e. assembly) lead time is presented in Appendix 10;
- the existence of a buffer between the printer and the placement machine which is capable of storing a complete batch (i.e. 50 PCB's).

b. Analysis carried out on the Petri-net tool (i.e. ARP tool)

Petri-net models generated by the SEW-OSA CASE tool were submitted to an analysis stage which included:

- verification of Petri-net properties. As a result of this analysis, the model was found to be free of deadlocks and livelocks. Appendix 6 presents a report generated by the ARP on such a verification. If undesirable problems had occurred in the model, ARP would have identified them, the region in the model where they occurred and an indication of the changes required in the model.
- a step-by-step execution of the model was carried out to check for inconsistencies in the operation of the system (a fragment of an ARP report on this analysis is also presented in Appendix 6). No inconsistencies were found and the model behaved in a manner expected at the modelling stage. If any inconsistencies had been found, ARP would have pointed the source of inconsistency.

Following this analysis, a thorough evaluation of performance was conducted with respect to a number of performance metrics which served to qualify the model (and the system being modelled). This evaluation consisted of monitoring certain parameters of the line as the net evolved from an initial event to a final event. These events were established, respectively, as being "a request to assemble a batch of boards" and "the issue of a status indicating completion of a batch". The parameters monitored during net evolution were:

- manufacturing (or assembly) lead-time (i.e. the total time to produce a complete batch of boards). This measure is germane to any definition of throughput (i.e. number of boards produced in a unit time);
- levels of work-in-progress at several stages along the SMT assembly line; and
- levels of utilisation of various resources included in the SMT assembly line.

These performance measures were derived from the following information

generated via ARP reports (refer to Appendix 6 for an example of an ARP report on performance):

- average time that the net took to evolve between initial and final events, this to calculate the manufacturing lead-time;
- the average marking of places, this in order to obtain an estimate of levels of work-in-progress and utilisation.

Averages were obtained from a number of simulation runs. Essentially, ARP continuously executes simulation runs, whilst accumulating values from each run, calculating an average value and comparing values with previous values. If the difference between values is less than a specified level of precision, ARP stops the simulation automatically and issues a report. A simulation can also be stopped by the user striking a key. For the particular SMT assembly line model investigated, ARP executed 52 simulation runs per minute (on a 486, 66MHz machine).

Prime focus of the simulation studies was on assessing:

a. manufacturing lead-time; level of utilisation of operator 1; level of utilisation of the printer; and profile of work-in-progress - each as a function of batch size and various inspection and printer checking rates.

b. manufacturing lead-time; and work-in-progress - both as a function of the speed of the printer.

These performance measures so derived should provide sufficient data to support decisions about how the line should be configured and operated. Issues of importance which underlie these decisions are:

- identification of critical resource constraints (i.e. bottleneck operations [Goldratt 1984]);
- control over operations for which execution time can vary considerably (e.g.printer checking and inspection operations);
- definition of a strategy for balancing the line, so that a uniform distribution of work-in-progress along the line can be achieved;
- evaluation of the levels of utilisation of resources so that decisions on possible capacity increases can be made;
- verification of the impact that certain configuration decisions can have upon manufacturing lead-time, thereby influencing the throughput of the shop-floor.

The performance data obtained and a brief discussion of the decisions that can be made based on these results is presented below.

c. Manual operations carried out during the "preparation" process

From the operations listed in Table 5, "check" (of the printing process) and "inspection" (on the inspection conveyor) are the only ones not carried out for every board (see behaviour models of Figure 61). During production periods they are also the only operations performed by human beings. To evaluate the impact that manual operations can have upon the performance of the SMT assembly line, studies were carried out to assess the impact of the time that operators take to perform their functions.

Figure 43 depicts changes in manufacturing lead-time as a function of various checking rates associated with the printing process. The values of checking rate plotted in this figure range from an extremely low value (i.e. 5 boards in every 100 boards printed) to one hundred percent. Each plot corresponds to a different time that operator1 takes to conduct a check of paste size and distribution for a single PCB. The extreme values assigned range from 5 s to 5 minutes. As emphasized earlier in this section, this study was conducted using the default values presented in Table 5 and the conditions defined on page 226.

The family of plots indicates that for values of checking time above 100s the checking rate becomes an important consideration, with respect to limiting lead-time. For instance, with a checking time of 100 s, the lead-time grows rapidly if the checking rate is increased beyond the default value used at D2D (i.e. 20%). This is a good example of where simulation results confirmed empirical values (of good practice) which were assigned based on D2D's practical experience of the process. This study was the first simulation exercise conducted in this area at D2D.

The effect that these conditions have upon the level of utilisation of operator1 and the printer can be observed from plots of Figures 87 and 88. Here, as one would expect, when checking time and checking rate are increased, the level of utilisation of operator1 increases dramatically. Indeed, this increase in the level of utilisation of operator1 causes an increase in the manufacturing lead-time, following a corresponding increase in the volume of boards waiting for a print check (see discussion about work-in-progress later in this section).

Figure 87 also indicates the point at which additional operators may be required to supervise the printing process, should a high checking rate and checking time be necessary to enable operation of the line.

As indicated by Figure 88 utilisation of the printer remains at a level between 40 to 45 percent for values of checking time below 100s. This level of utilisation decreases considerably when the values of checking time and checking rate are

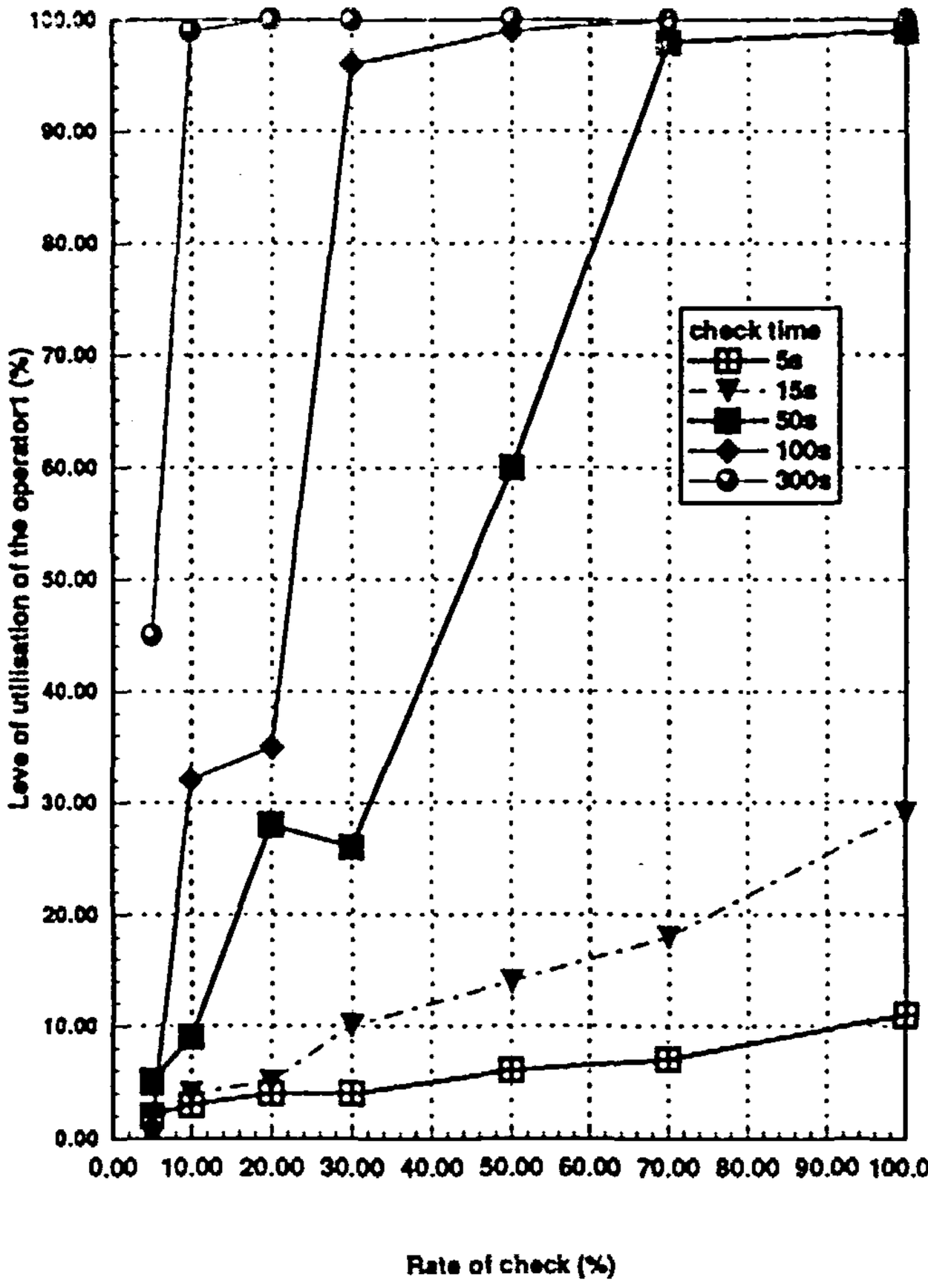


Figure 87 - Level of utilisation of operator1

increased. In a situation in which all boards are checked with a unit checking time of 300 s, the level of utilisation of the printer can be as low as 5%.

The data on resource consumption, for a certain configuration for the assembly line, can also be an input to an analysis of cost associated with alternative modes of operation (e.g. through the use of "Activity-Based Costing" [Shaharoun 1994]).

d. Manual operations involved in the "populating" process

A similar performance study was conducted for the operations performed by operator2, who inspected populated boards. In this study, inspection rate and inspection time were varied whilst other parameters of the SMT assembly line were kept constant.

The plots of Figure 89 show how manufacturing lead-times vary for various inspection rates. It can be observed from this figure that, as with the printing process, the manufacturing lead time is significantly affected by a simultaneous increase in inspection times beyond 100 s, and inspection rates beyond 20%. However, these values (of inspection time and inspection rate) have a weaker effect upon the lead-time

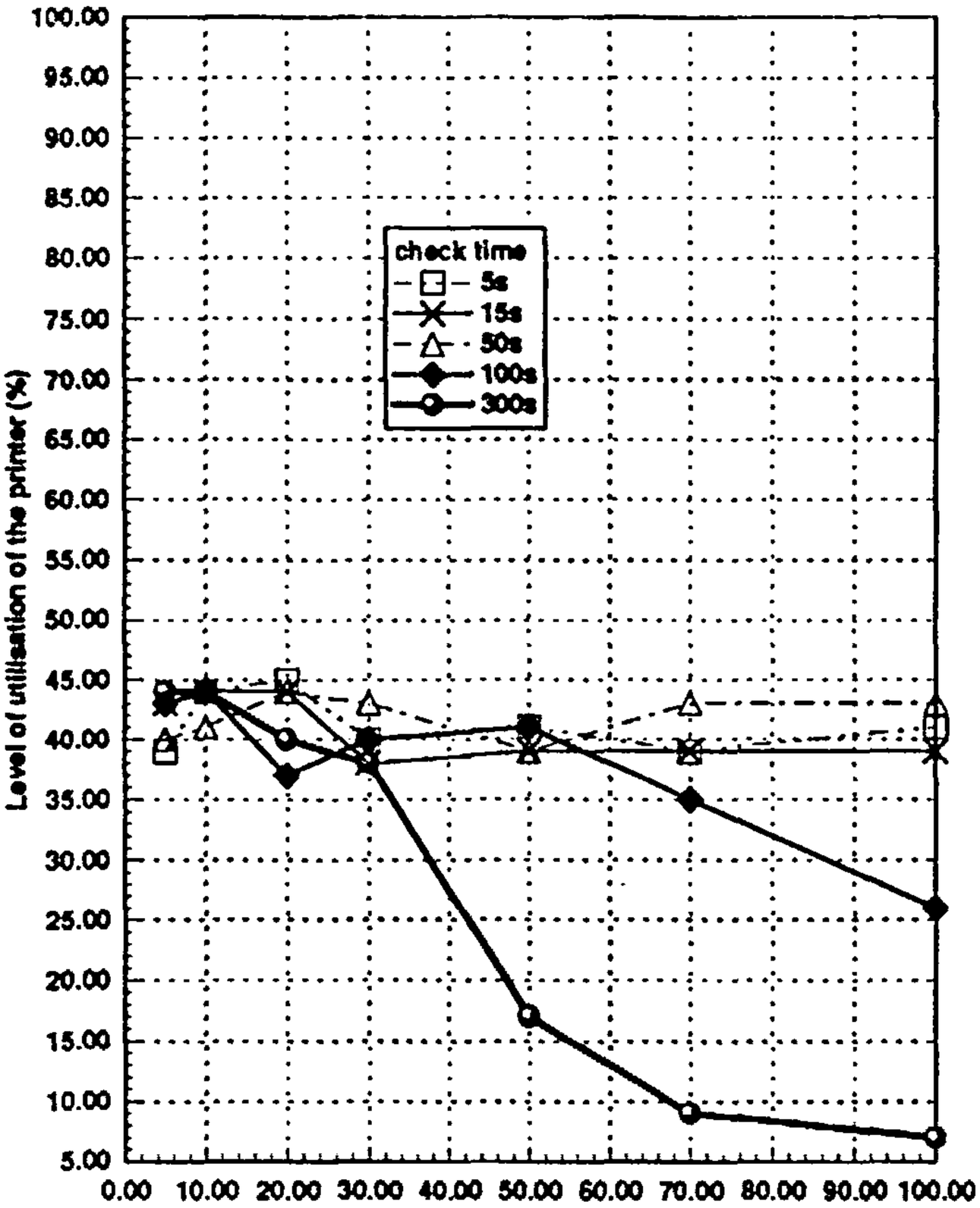


Figure 88 - Level of utilisation of the printer as a function of checking rate and time

than corresponding factors for the printing process. This fact is clearly visible when the maximum values of lead-time are compared with each other. Even though a higher maximum inspection time was used in the latter case (i.e. 500 s as opposed to 300 s), a lower worst-case lead time was obtained (i.e. 10,240.91 s as opposed to 15,224.92 s). This difference can be attributed to the fact that the print process stops when a checking is being performed. Conversely, the placement process only stops when the inspection takes so long that the available PCB's slots on the inspection conveyor are filled up.

During the analysis of the printing process, it was found that the influence that checking time and checking rate had upon the placement process was negligible. However, the same cannot be said in regard to the influence that inspection time and inspection rate had upon the printing process. Figure 90 illustrates such an influence by depicting the level of utilisation of the printer, as variations occur in the inspection time and inspection rate. It can be observed from this figure, that the range of printer utilisation (which in the default conditions range from 40 to 45%) decreases as the

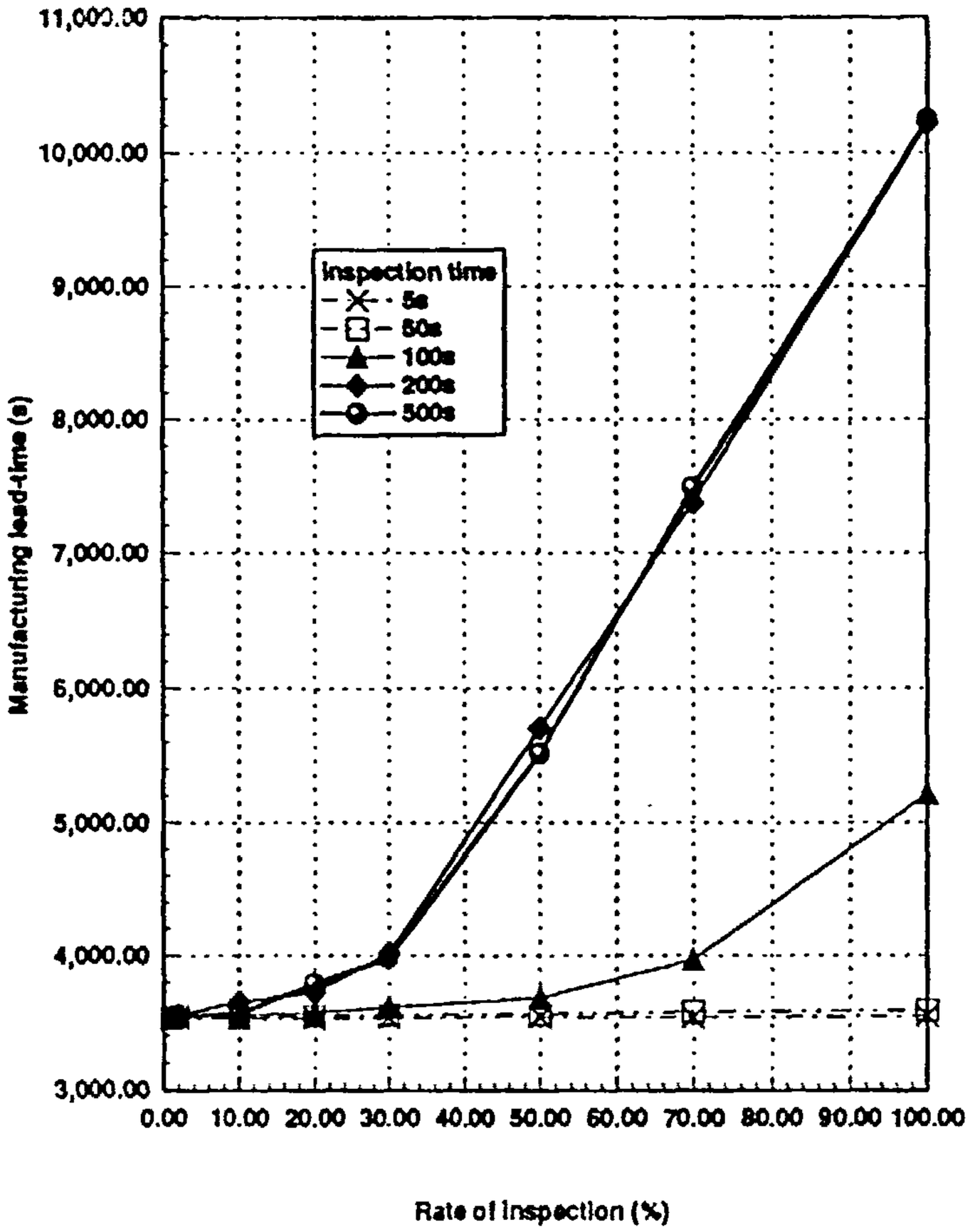


Figure 89 - Manufacturing lead-time as a function of inspection rate and time

inspection rate increases (for inspection times beyond 100 s). This also influenced the work-in-progress levels, as explained later in this section, and indicated that the placement process constitutes the bottleneck operation in the SMT assembly line.

e. A profile of work-in-progress

Related to the identification of bottleneck operations is the study of the profile of work-in-progress along the SMT assembly line. The level of work-in-progress was investigated at three points along the SMT assembly line (these points being related to control points of the business processes depicted in Figure 61), namely:

- at the **preparation** stage. This includes any boards in the printer or ones being checked by operator1.
- on the **buffer** between the printer and the placement machine.
- at the **population** stage. This includes any boards in the placement machine or on

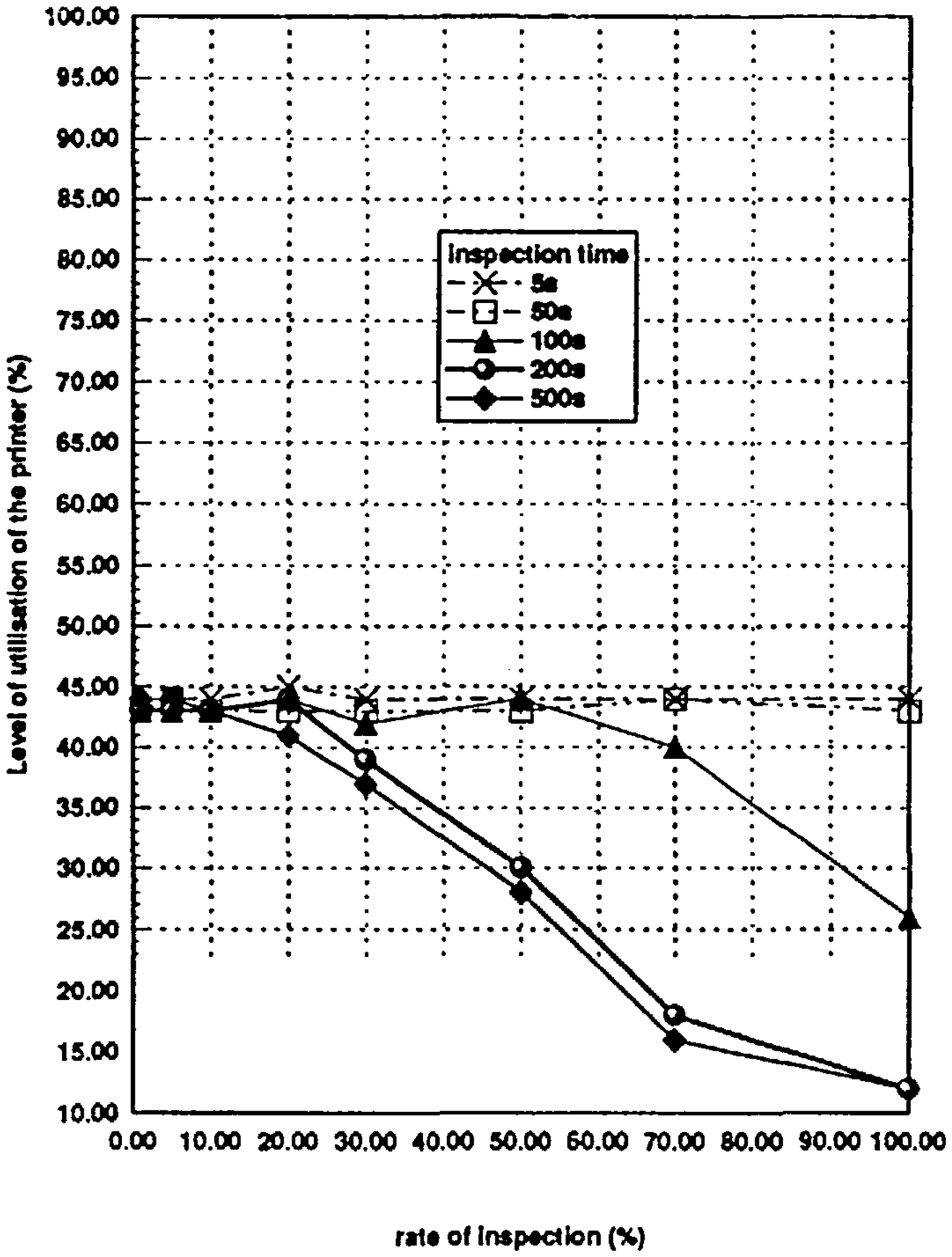


Figure 90 - Level of utilisation of the printer as a function of inspection rate and time

the inspection conveyor.

- at the finishing stage. This includes any boards in the reflow solder machine or in the wash-off machine.

Figure 91 depicts a profile of work-in-progress along the SMT assembly line for the default operating conditions discussed earlier in this section. Values attached to the vertical axis represent the average number of boards at each stage of the line. The simulations demonstrated that these values always total about 25 boards, with the remaining 25 boards being located at input and output buffers.

Figure 91[1] clearly indicates that the placement machine is the bottleneck operation, as the greatest proportion of the work-in-progress is concentrated before it. The second slowest operation is the printer, followed by the reflow solder and wash-off machines.

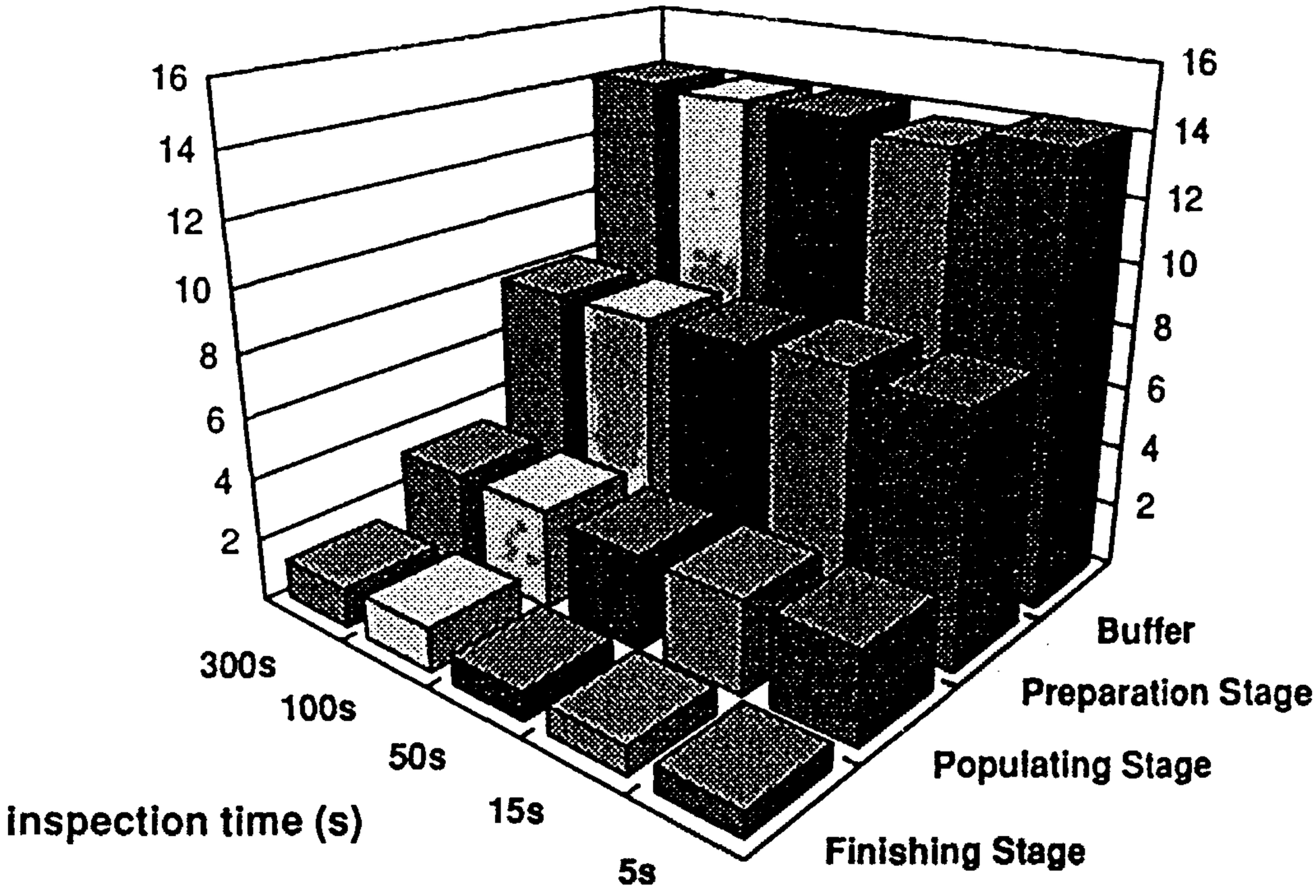


Figure 91 - Work-in-progress profile vs inspection time

One can also observe in this figure that the inspection time does not affect the profile of work-in-progress in the line for the default inspection rate (i.e. 10%). However, when the inspection rate is made to vary (for a fixed inspection time of 200s), the profile of work-in-progress does change as indicated by Figure 92. As the inspection rate is increased, the work-in-progress at the preparation stage decreases, whereas the average number of boards at the remaining stages decrease to compensate for it. As the populating stage is slowed, in relative terms, the preparation stage is completed much earlier than the downstream operations.

Further results were obtained from a study of changes on parameters associated with the preparation stage. In this case, it was observed that the checking time does not affect the level of work-in-progress in the line (for the default checking rate of 20%). However, for higher checking rates (i.e. 70% and greater), the checking time had a much greater influence, as depicted by Figure 93. As one could expect, as the checking time continues to increase, the level of work-in-progress increases leading to an emptying of remaining stages.

The differences in behaviour between the preparation and populating stages, in regard to the effect upon work-in-progress as a function of checking and inspection

---

1. It should be noted that the order of the stages along the line was changed in this figure to improve the visibility of the results. This was due to a greater quantity of boards being stored in the buffer compared to the preparation stage.

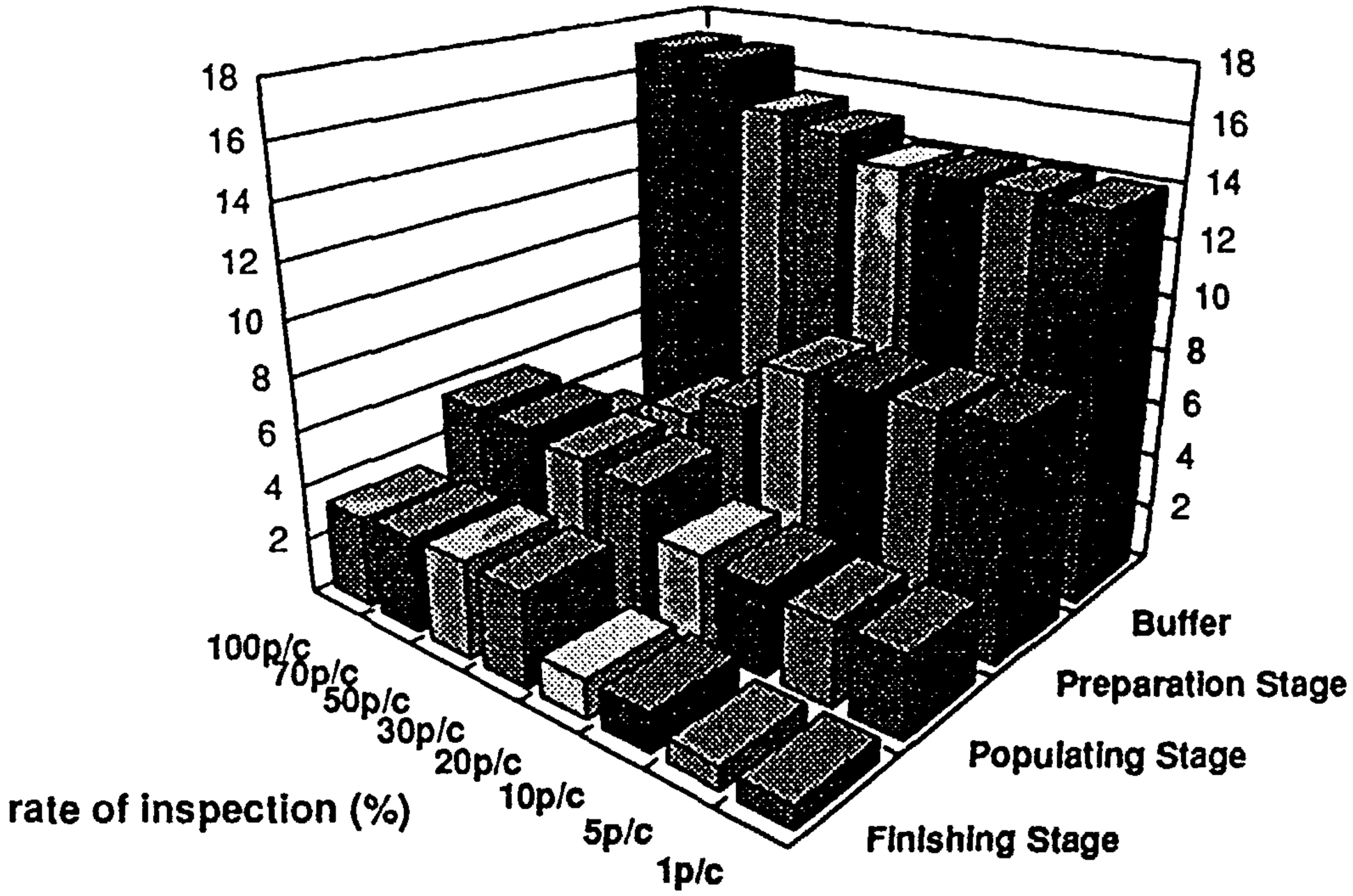


Figure 92 - Work-in-progress profile vs inspection rate

times, respectively, can be attributed to their different speed and the different capacity of their associated buffers. A profile variation similar to the one depicted in Figure 92 was also observed for variations in checking rate for a checking time of 100s.

The results obtained in this study of work-in-progress can be used to establish suitable operating conditions which could improve line balancing. Bearing in mind the philosophy of bottleneck management proposed by Goldratt [Goldratt 1984], which seeks to synchronise shop-floor operations based on bottleneck operations, a further study was conducted. Here, it was observed that control of work-in-progress profile can be obtained by controlling the rate at which boards are printed. As shown in Figure 94, an approximate print rate of 50 s (between boards) provides a uniform distribution of boards along the assembly line.

Control of the printer in order to synchronise its operation with the placement machine leads to a well balanced line. This can be achieved in a system prototype by including a time parameter as an attribute in the transfer function of the enterprise activity "print" (this enterprise activity is part of the model depicted in Figure 61).

f. Discussion of simulation results

The simulation results and their analysis simply illustrate the type of evaluations that can be performed by using the functionality provided by the simulation capability of SEW-OSA. These evaluations can serve as a basis to support decisions on the selection of alternative system configurations. Generally speaking, this can allow

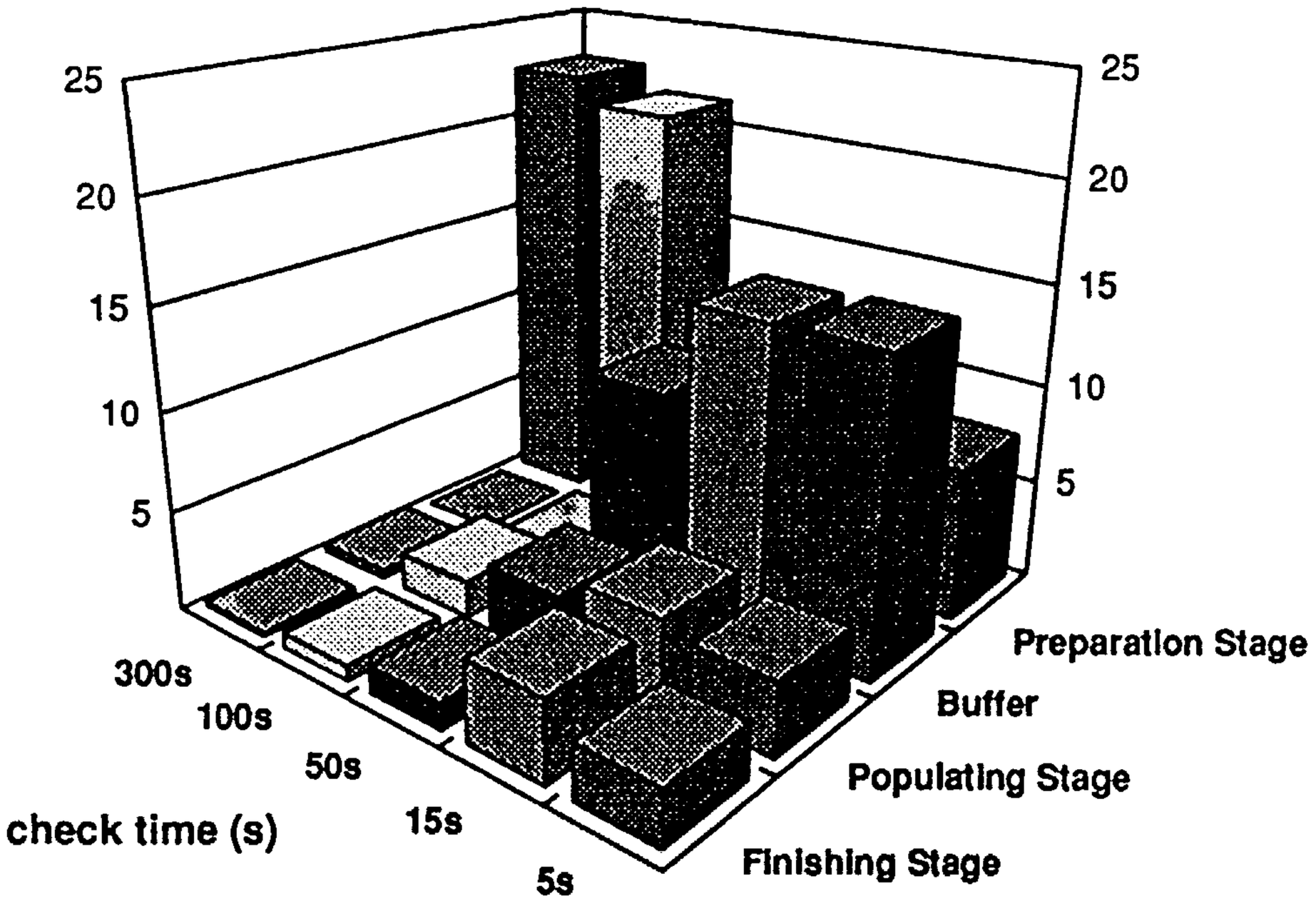


Figure 93 - Work-in-progress profile vs checking time

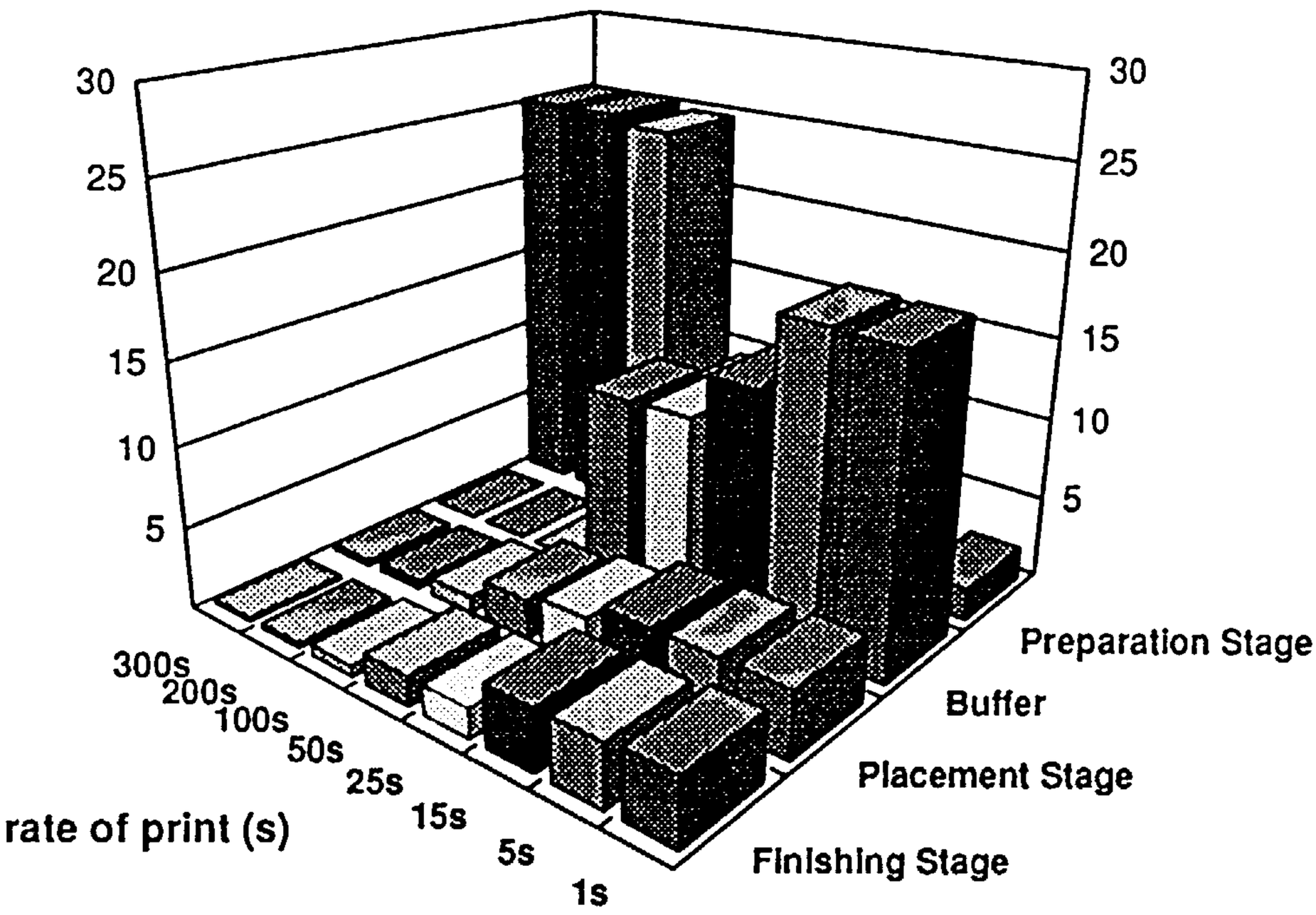


Figure 94 - Work-in-progress profile vs print rate

the following:

- an analysis of system operation through a verification of the Petri-net properties and step-by-step simulation;
- the verification of appropriateness of configuration parameters, which typically are empirically defined and taken for granted during system operation (e.g. which typically are checking and inspection rates);
- the effect of process parameters (e.g. printing rate) upon system performance (e.g. manufacturing lead time) and operation (e.g. line balancing);
- an evaluation of the effect that configuration parameters can have upon system performance (i.e. level of utilisation of the system components; system throughput; and distribution of work flows through the system, e.g. work-in-progress). Costing data can also be added to these measures (e.g. through ABC [Shaharoun 1994]) to provide an indication of system performance based on metrics such as those recommended by Goldratt [Goldratt 1984], e.g. throughput, work-in-progress and cost.

It should be emphasized that these measures are not limited to shop-floor systems. Throughput, work distribution and level of utilisation are metrics which can be applied to any system. Very importantly, the ability to control factors which determine the results obtained, provides a basis for justifying the introduction of model-driven operation. In the case of the SMT assembly line, this can be manifest by introducing model-driven cell controllers. In an office system, it would likely require the use of work-flow systems.

Similar results and analyses can be realised from models of primary elements of D2D shop-floor. These could be used to define shop-floor configurations which have a capacity to produce projected orders over a period of time. These could lead to an identification of:

- the need for additional lines in order to cope with the throughput required from the shop-floor;
- the need for better lines, in order to smooth the distribution of work-in-progress, improve throughput or achieve better quality levels.

The above examples of manufacturing considerations can impact upon the shop-floor as a whole. Nonetheless, certain of these considerations depend upon factors (and associated variables) which are localised within a single assembly line. For example, simulation results for a single assembly line can enable investigation of the

impact of "micro-decisions" (e.g. those taken within the context of one line) upon the performance of the shop-floor as a whole. This also illustrates the idea of inter-relating analysis carried out upon models created at different levels of abstraction.

11.1.3. The performance of SEW-OSA at the rapid-prototyping stage

The simulation results discussed in the previous section only take into consideration manufacturing process parameters (e.g. process cycle times). They do not account for the overhead introduced by the model-enactment capability whilst controlling the interactions among system components and hence system operation. This overhead is considered to include the effect (in terms of time delay) of:

- internal processing within each business entity component;
- communication protocol among these components (implemented to achieve model-enactment);
- use of the CIM-BIOSYS integrating infrastructure (by the communication protocol); and
- use of the underlying computer infrastructure, by both CIM-BIOSYS and the communication protocol (i.e. X-Windows programming environment, Unix operating system, TCP/IP and the Ethernet stack of protocols).

In any study of overhead, two issues are of particular importance:

- the value of overhead and how it relates to changes in the operating conditions of the system (e.g. configuration of the computer infrastructure);
- an identification of key contributors to this value of overhead.

To investigate these issues, the following study activities were conducted:

- a set of tests were carried out to measure overhead (in terms of time) for a number of different configurations;
- a theoretical study of the time values was conducted to identify major contributors to overhead.

a. Overhead determination

The overhead, measured as a function of the total manufacturing lead-time can be expressed as follows:

$$OH\ (\%) = (LT_{rp} - LT_s) \times 100/LT_s \quad (1)$$

where,

- OH is the overhead as a percent of an ideal value of manufacturing lead-time (i.e. without overhead);
- LT_{rp} is the lead time measured through the log files of the business entity;
- LT_s is the lead time measured through simulation runs.

In SEW-OSA, the difference (LT_{rp} - LT_s) constitutes the time-delay overhead in seconds (OH_s), added by the model-enactment capability to the actual manufacturing lead-time. LT_{rp} is the time between the two event occurrences, registered by the Event Handler, which mark the beginning and end of a given set of model occurrences. In the case of the SMT assembly line, this is the time between EV-1 and EV-2 (defined in the domain diagram shown in Figure 25) which set the time limits for the start and the end of production of a batch of boards.

In order to determine the level of overhead imposed by the model-enactment capability, a number of prototypes were generated and enacted. A series of values of LT_{rp} were obtained by subtracting the time-stamped events EV-2 from EV-1. The overhead (OH) was calculated according to equation (1), where LT_s was measured through simulation runs.

The elements which influence performance, over which a designer has more direct control, are related to the system configuration (i.e. the definitions encapsulated in the configuration diagram shown in Figure 34). Some studies (by the author) of this influence which have been investigated are:

- the number and type of hosts used in a given configuration;
- the number of software processes allocated for execution on each host;
- the number of threads of business model execution competing for resources (this related to the batch size).

The study concentrated on two conditions: (1) the behaviour of overhead as a function of the batch-size and (2) the level of overhead for alternative system configurations. For both conditions, the values of LT_s were extracted from Figure 95, which describes the behaviour of the manufacturing lead-time as a function of the batch size. In order to limit variations in the measures of LT_{rp}, rates of inspection and checking of 0% were adopted in this study (for they are the nearest to the default operating conditions of the assembly line).

The overhead was measured with respect to the system configuration summarised in Table 6. These configurations relate to alternative ways of distributing both the components of the business entity and the active resource components of a

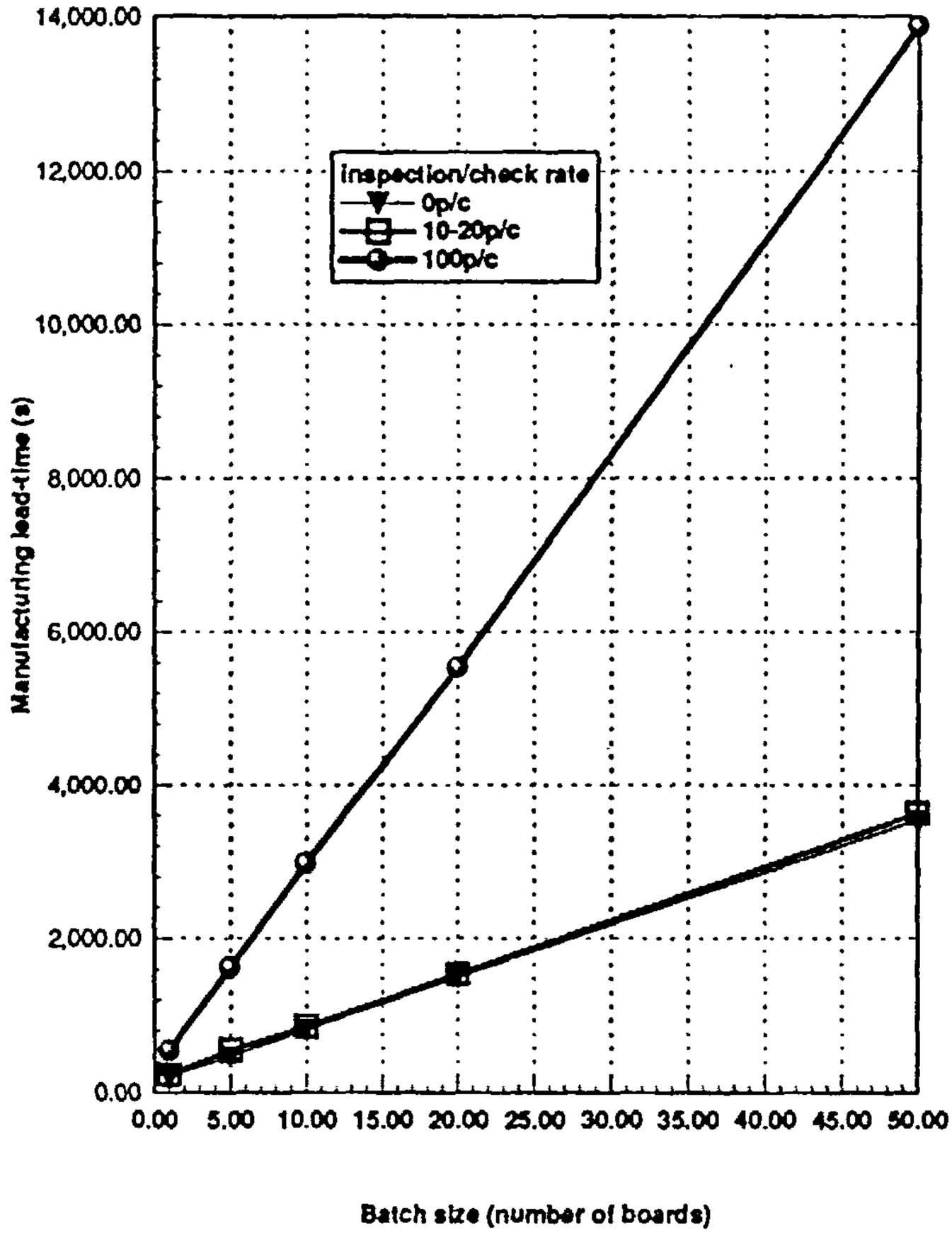


Figure 95 - Manufacturing lead-time as a function of the batch size

system solution (across the most powerful Sun workstations available at the MSI Research Institute). Some important underlying conditions associated with tests performed on the model-enactment capability were as follows:

(1) The tests were performed by seeking to maximise the speed of operation of the model-enactment capability, whilst providing a reliable service. This includes minimising the time-delay between firings of two consecutive transitions of the Prolog engine. A minimum time delay existed due to a problem of deadlock in data-base transactions between the Prolog engine and the functions that provided access to the CIM-BIOSYS integrating infrastructure;

(2) CIM-BIOSYS provides two methods of transferring messages, namely: expedited and secure. The latter provides confirmation of message transferred between CIM-BIOSYS and its applications, whilst the former does not. Thus, whilst the former provides the best message transferring speed, it can deteriorate the level of reliability of the infrastructure for situations of very high traffic. Tests were

conducted in both situations which have led to a recommendation to adopt the latter method, due to an unacceptable level of message loss in the former method, in respect of the traffic required by the business entity.

(3) An inherent overhead associated with the debugging capabilities of the business entity is included as is an overhead concerned with registering events in the log files.

(4) Tests were performed with the associated hosts fully dedicated to serving the processes depicted in Table 6[1]. That is, no additional Unix process was being executed when the model-enactment capability was tested.

Table 6 - Alternative system configurations

n.	hosts	configuration[a]
1	1	BE +ARCs -> Sparc10
2	2	BE -> Sparc10, ARCs -> Sparc10
3	3	BE -> Sparc10, 1/2ARCs -> Sparc10, 1/2ARCs -> Sparc1+
4	2	EV + AC + EAs -> Sparc10, PC + RM + ARCs -> Sparc10
5	2	EV + PC -> Sparc10, AC + RM + ARCs + EAs -> Sparc10
6	5	EV + AC + EAs -> Sparc10, PC + RM -> Sparc10, 1/3ARCs -> Sparc2, 1/3ARCs -> Sparc1+, 1/3ARCs -> SparcClassic,
7	4	EV + PC -> Sparc10, AC + RM + EAs -> Sparc10, 1/2ARCs -> Sparc1+, 1/2ARCs -> Sparc2
8	4	EV + PC + RM + AC -> Sparc10, EAs -> Sparc10, 1/2ARCs -> Sparc1+, 1/2ARCs -> Sparc2
9	2	EV + PC + RM + AC -> Sparc10, ARCs + EAs -> Sparc10
10	5	EV + PC -> Sparc10, AC + RM -> Sparc10, EAs -> Sparc1+, 1/2ARCs -> Sparc2, 1/2ARCs -> SparcClassic

a. Legend:

BE: all business entity components (i.e. EV + PC + RM + AC + EAs);

ARCs: all active resource components;

EV: Event Handler;

PC: Process Controller;

AC: Activity Controller;

RM: Resource Manager;

EAs: enterprise activity occurrences.

The values of overhead obtained by applying these conditions are shown in Figures 96 and 97. Figure 96 depicts how the overhead behaves with an increase in the number of threads of the business model, each competing for active resource

1. The current version of SEW-OSA allows distribution of the business entity to the level of granularity of a component. It is envisaged that future versions will support the distribution of several instances of each component to a number of computer hosts.

components. These plots were obtained for configuration 1, as defined in Table 6. A maximum batch size of six boards was used in the tests, as this was found to be the maximum number of threads of the business model that the business entity and its supporting software could execute without losing messages. For a larger number of threads in the business model messages can be lost.

This limitation can be caused by limits in the size of the internal buffers of the CIM-BIOSYS infrastructure which are used to store the messages processed on each host[1]. Basically, the business entity generates messages at a rate (i.e. above five messages per second) with which the CIM-BIOSYS infrastructure is unable to cope (i.e. it is unable to process all the messages before its internal buffers are filled up).

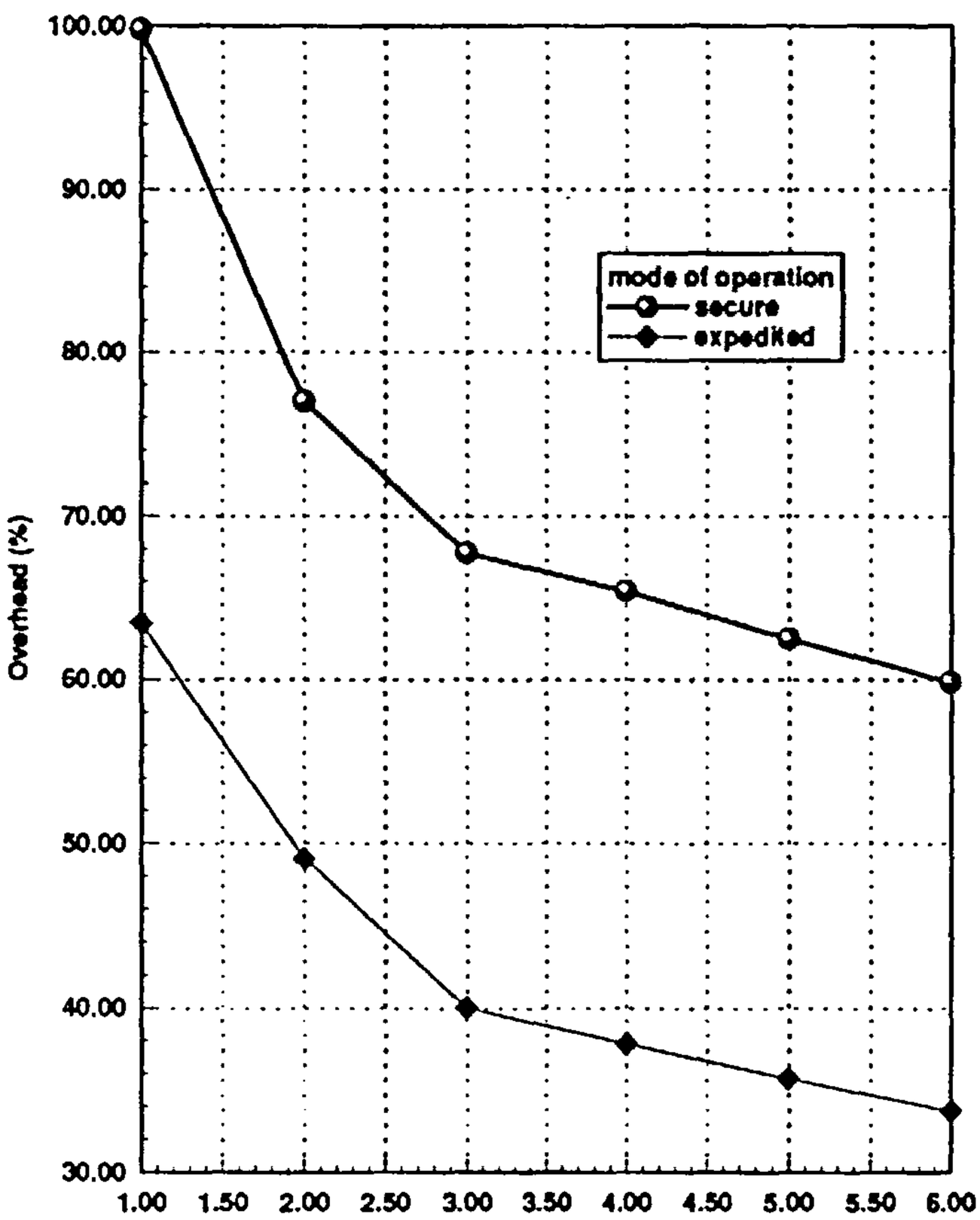


Figure 96 - Overhead as a function of batch size

Notwithstanding this limitation on batch size, the plots shown in Figure 96

---

1. Although it should be stated that it is a relatively easy matter to change the size of the CIM-BIOSYS internal buffers, this was not done here as the author was keen to maintain experimental conditions in as a stable a state as possible.

provide an interesting indication of how the overhead behaves following increases in the number of threads of business model execution. The behavioural pattern obtained with these plots came somewhat as a surprise, for it challenged an intuitive judgement made beforehand. That is, one might expect the overhead to increase with the increase in batch size. However, apparently, the overhead decreases as the batch size increases.

The phenomena demonstrated in Figure 96 may be a direct result of the way in which the CIM-BIOSYS infrastructure handles its buffers. Messages passed via the CIM-BIOSYS infrastructures are firstly stored in a buffer at the time of their arrival. These messages are passed to appropriate applications via a polling process which is activated periodically. Thus, message passing is performed as a batch thereby, saving time.

It is also clear from Figure 96 that the level of overhead imposed by the business entity as a whole is quite considerable (i.e. above 50% for the secure mode and above 30% for the expedited mode). Strategies for reducing this overhead need to be identified and adopted in order to practically implement SEW-OSA-based model-driven control of systems.

In seeking further data to define such a strategy, the results depicted by Figure 97 were obtained. Here, the overhead associated with a batch size of one board was measured for each of the configurations shown in Table 6.

Once again, probably contrary to intuitive judgement, the overhead generally increases, as the number of host computers is increased, so that the processes are more distributed across the computer infrastructure. Indeed, the lower overhead corresponds to a configuration in which the complete system is executed on a single host (i.e. configuration 1 in Table 6), whereas the worse case is for the configuration in which the processes are distributed among five hosts (i.e. configuration 10 in Table 6). Once again, possibly this result can be explained by the fact that message waiting time (in the CIM-BIOSYS buffer) is the determinant factor in the overhead for each of these configurations. Distribution might be expected to reduce CPU load albeit that, for this evaluation work, the CPU load was not very high (i.e. less than 20%). In such circumstances, apparently, distribution is only adding to the overhead related to waiting time.

Other interesting observations made from the evaluations were as follows:

- even though configurations 4 and 5 use two hosts (similarly to configuration 2), their performance is poorer than that of configuration 3 (see Figure 97) which uses three hosts. This may well be due to a separation of business entity components into two hosts. These components are tightly coupled, so that they exchange a large number of messages (adding communication network overheads). This problem is also apparent from the results obtained with configuration 10, whereby the business entity is distributed over three hosts, one of them being a relatively slow computer

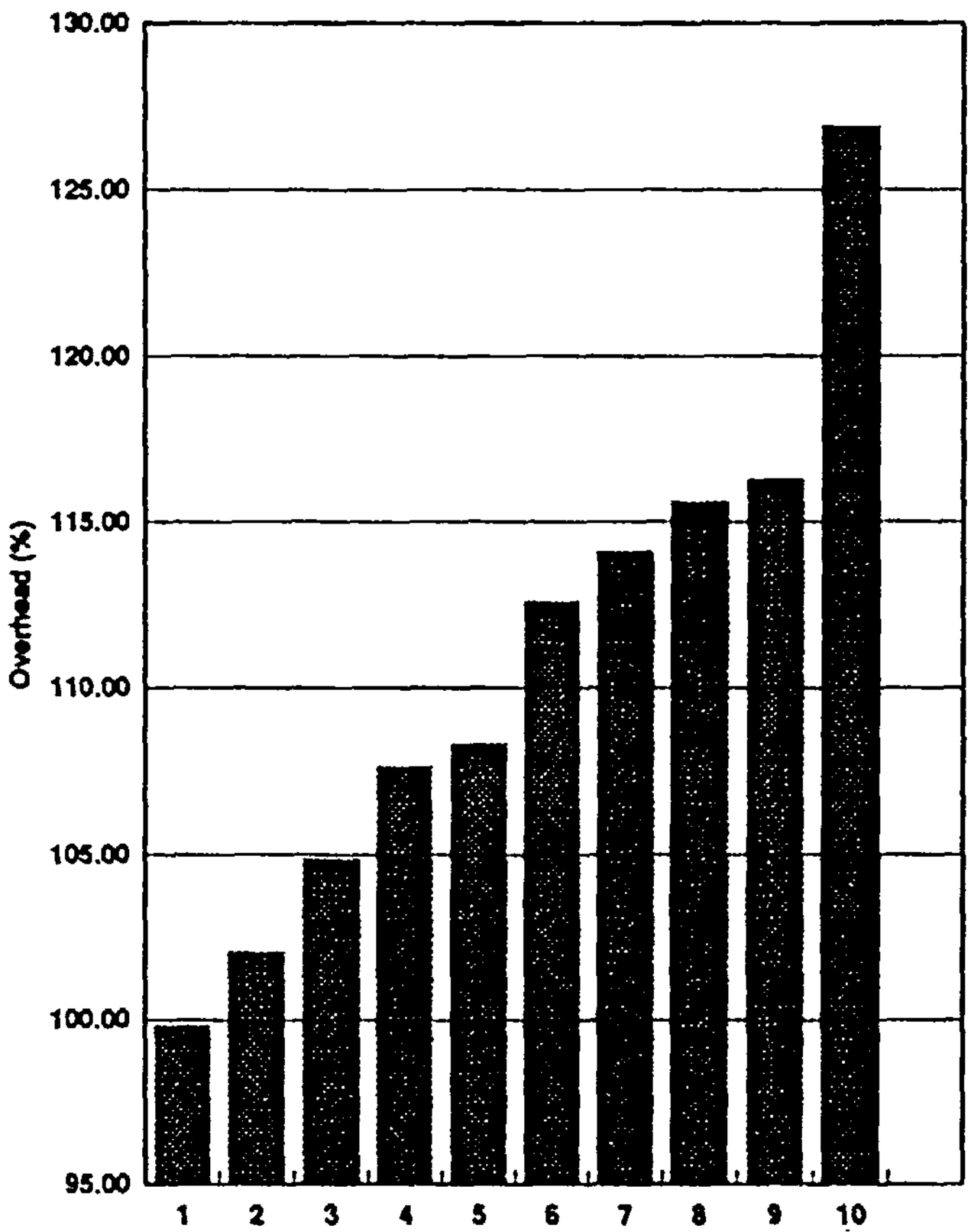


Figure 97 - Overhead for alternative system configurations

(i.e. a Sparc1+).

- if distribution of the business entity is a requirement, a clustering of certain processes into the same host is imperative, as some components are more tightly coupled than others (e.g. the Activity Controller and enterprise activity occurrences are very tightly coupled).

In order to recommend design improvements for the business entity (or any of its supporting infrastructural services) which could lead to a reduction in the levels of overhead, an analysis of contributions of a number of factors to the total overhead must be performed. This analysis is developed as follows.

b. Overhead contribution

A mathematical analysis of factors which contribute to LT_{rp} is summarised in the Appendix 11. This analysis has led to the definition of a general set of equations which combines the most important factors contributing to overhead. Results obtained

from applying this set of equations can be used to estimate the overhead produced when enacting a particular SEW-OSA model. These estimates can provide important measures of performance to be used in conjunction with practical measurements reported in previous section, in order to support design decisions.

It should be noted that the overhead contribution of certain factors will depend upon:

- features of the model being executed;
- run-time happenings associated with model execution (e.g. number of procedural rules and functions used by a process, number of events generated and functional operations used by an enterprise activity, etc.) which may vary with alternative threads in the flow of model execution; and
- variations in the processing loads of CPU's and the computer configuration over which the business entity processes and the active resource components are distributed.

Therefore, any calculation of overhead based on these relations should be viewed as an estimate, and refer to particular conditions of operation. However, estimates can provide an indication of relative contributions to overhead.

The equations derived from Appendix 11 were applied to the model of the SMT assembly line, this resulting in the following formula:

$$OH_s = 5.t_{SP} + 101.t_S + 96.t_{CBS} + 16.t_{PR} + 8.t_{aloc} + 8.t_{link} + 48.t_{tr}$$

This equation basically combines major time factors[1] which contribute to overhead.The conditions for which this formula provides a valid model are discussed in the

---

1. These time values are defined as follows: t_{SP} is the time required in the Unix operating system to spawn a new process representing a domain process (or business process) occurrence; t_{aloc} is the time that the Resource Manager can take to bind an active resource component to an enterprise activity occurrence; t_{link} accounts for the time taken by the CIM-BIOSYS infrastructure to start an application and establish a link (i.e. a connection) between that application and any other applications wishing to communicate with it (i.e. the application protocol of the CIM-BIOSYS infrastructure is connection oriented); t_{tr} is the elapsed time between two consecutive firings of transitions within an enterprise activity occurrence or active resource component; t_{PR} is the average time that a domain process or a business process occurrence takes to execute one of its procedural rules; t_S is the transmission time of a message between two Unix processes communicating via local sockets; and t_{CBS} is the transmission time of a message between two software applications communicating via the CIM-BIOSYS infrastructure.

Appendix 11, as is a detailed explanation of each of its terms.

The equation was populated with time values obtained from log files, having established the said operating conditions with configuration 1 (listed in Table 6). The resultant time contributions were estimated as follows:

- $t_{SP} = 0.320$ s;
- $t_{aloc} = 0$ s (this assumption is acceptable for the established operating conditions, i.e. batch size of 1 PCB);
- $t_{link} = 1.026$ s (for the secure communication mode[1]);
- t_{tr} 0.977 s;
- $t_{PR} = 0.010$ s;
- $t_S = 0.006$ s;
- $t_{CBS} = 1.018$ s (for the secure communication mode).

$$OH_s = 155.198 \text{ s}$$

which implies an overhead (in percentage) of:

$$OH(\%) = 81.56\%$$

When this value of percentage overhead is compared to the value of overhead in Figure 96, for the same configuration[2], an error of 18.3% was obtained. Considering the necessary simplifications embedded in the mathematical approach, this error may be considered to be acceptably small.

In any case, the relative contribution of the factors contained in the equation is more important than the total value of overhead. Table 7 illustrates the percentual contribution of these factors as estimated mathematically.

Table 7 indicates that the greatest proportion of contribution (i.e. 63%) stems from a communication overhead introduced as the business entity needs to draw very frequently upon the application services of the CIM-BIOSYS infrastructure (i.e. services which enable application-to-application communication). The second largest overhead contribution is imposed by the Prolog engine, which executes the behavioural models of enterprise activity occurrences and active resource components. The third largest is the process of starting a CIM-BIOSYS application in the event of an enterprise activity occurrence. This overhead occurs largely from the need to establish

---

1. As previously indicated, the "secure communication" mode was adopted due to reliability be an essential requirement of the business entity.
2. The value of overhead is 99.80% for the secure communication mode.

links with a CIM-BIOSYS application before any communication can occur.

Table 7 - Contribution to overhead

factor	value (s)	contribution (%)	source of overhead
$5.t_{SP}$	1.6	1.0	Process Controller
$8.t_{aloc}$	0	0.0	Resource Manager
$8.t_{link}$	8.208	5.3	CIM-BIOSYS infrastructure and Activity Controller
$48.t_{tr}$	46.896	30.2	Prolog engines
$16.t_{PR}$	0.16	0.1	Process Controller
$101.t_{S}$	0.606	0.4	Process Controller
$96.t_{CBS}$	97.728	63.0	CIM-BIOSYS infrastructure
OH_s	155.198	100.0	Total business entity

The overhead imposed by the structure of the Process Controller (i.e. 1.5%) stems from the complexity of the strategy adopted for executing process behaviour (i.e. communication between Unix processes representing business processes and domain processes, as illustrated by Figure 52).

c. Recommendations about overhead reduction

The results presented in this section provide a basis upon which design and implementation improvements can be made to various elements contained in or used by the business entity. Areas which might be addressed in pursuing such improvements are:

i. **The Prolog engine** used to produce active resource components and enterprise activity occurrences:

   This engine can execute interpreted code compliant with the surface syntax built in Prolog. Although interpreted code enables immediate execution, it also imposes significant time constraints and processing loads upon the host CPU. An alternative that is being considered for implementation by I. A. Coutts (the MSI researcher responsible for the realisation of this module) is to eliminate the Prolog engine and directly enact models in compiled code. That is, the syntax which currently represents active resource components and enterprise activity occurrences would be replaced by "C" programs.

   Such initiatives can be expected to considerably reduce the overhead introduced by this component.

ii. **The CIM-BIOSYS infrastructure:**

   Arguably, the greatest reduction in overhead could be realised by re-designing and re-implementing the services provided by the infrastructure, so that it specifically

meets requirements to enact SEW-OSA models. In meeting those requirements, two issues are of particular interest: (1) to reduce the time to transfer messages between applications and (2) the need to reduce the overhead associated with establishing links with applications[1], before communications can take place between them (such as the establishment of a connectionless service for application-to-application communication).

It is important to emphasise that the recommended improvements should not be viewed as a criticism of the capability provided by the CIM-BIOSYS infrastructure. Furthermore, the CIM-BIOSYS infrastructure was not initially designed to serve time critical interactions such as those required by the business entity. The author also expects that similar limitations would have emerged other integrating infrastructure, if alternatives to CIM-BIOSYS had been available. Indeed, in spite of its different underlying concepts and resultant overhead limitations, the CIM-BIOSYS infrastructure provided a level of location transparency which enabled the business entity to be developed in the first place.

Here, it is important to note that: (1) neither CIM-BIOSYS nor SEW-OSA were designed to meet requirements or optimise the performance of the other; (2) either of them could be tuned or upgraded to improve the performance of the whole; and (3) both may need radical re-design or use of different enabling technologies for the SEW-OSA approach to work generally in practical situations.

Some minor changes to the CIM-BIOSYS infrastructure are being carried out by J. Gascoigne [Gascoigne 1992] (the MSI researcher responsible for its realisation) to improve its performance when enacting SEW-OSA models.

Some improvements can be made by tackling key problem areas identified by this study which introduced significant overhead. However, the author recognises that the issue of providing adequate IT services to enable model-enactment at an acceptable level of performance is a complex one. In the case of SEW-OSA, this complexity is re-enforced by the level of functional requirements that the business entity meets, such as:

- to transform a static description of types of processes and activities into actions and transactions within the business entity in a manner which reflects the behaviour of instances of processes and activities being executed;
- to achieve such a transformation by binding functions and resources at the latest

---

1. Here, it should be noted that CIM-BIOSYS was not conceived to facilitate high speed interaction, but to enable non-time critical interaction between manufacturing applications in a highly flexible (configurable and extendible) manner. The way in which CIM-BIOSYS establishes links between applications reflects its different emphasis and intended purpose.

possible stage of system run-time, with a view to achieve complete separation between behaviour and functionality. This should facilitate use of standard elements of functionality supplied in the market on an 'off-the-shelf' basis.

One might argue that model-enactment can be achieved at better performance levels through designing solutions which do not emphasise these issues. This research does not refute this claim. However, as discussed throughout this thesis, these among other key concepts constitute means by which the complexity of an integrated manufacturing system can be handled by a CIM-OSA-based approach, whilst creating solutions which can be flexibly modified.

11.1.4. Performance at run-time

One of the main reasons for building a system prototype from SEW-OSA models is to enable analysis and evaluation, examples of which are given in previous sections. The external behaviour of a prototype should resemble as closely as possible the behaviour of the final system. If this requirement is realised, then performance measures obtained from a prototype can be used as a performance indicator for the final system. These considerations allied to the limited resources to carry out a physical implementation were the main reasons for not carrying out an overhead study of a physical SMT assembly line.

Nevertheless, a qualitative performance evaluation of the run-time system was carried out for a system configuration in which active resource components are emulated by a shop-floor animator, as discussed in Section 9.4.

The main qualitative result obtained, involving all members of the "Model-Driven CIM" project team is to use the model-enactment capabilities of SEW-OSA to demonstrate a fully operational system with the business entity integrated to the animator. This helped prove the feasibility of many concepts, including concepts described in this thesis.

Further quantitative analysis of system performance associated with this demonstration solution have not yet been attempted, as the animator (which represents all active resource components) is executed upon only one host. Therefore, measures of overhead would have led to very similar results to those obtained with configurations which clustered all active resource components into one host (as listed in Table 6).

11.1.5. Performance issues related to the engineering process

These issues seek to qualify aspects of the process of engineering a certain enterprise domain with the support of SEW-OSA. Absolute measures of "quality" associated with such a process are difficult to establish. However, by comparing engineering processes conducted in different ways it is possible to obtain an indication

of the relative merit of a given approach. Ideally, comparisons should be based upon the use, application and evaluation of the approaches under well defined experimental conditions, i.e. by engineering solutions under equivalent conditions and comparing the resulting performance of the solutions (also) under equivalent conditions. Examples of approaches which could be compared are: (1) SEW-OSA against current practice and (2) SEW-OSA against other emerging environments for enterprise integration (as discussed in Chapter 2). Such a comparison should then be based on:

- **process metrics**[1]. This should related to relevant features of the actual engineering cycle (e.g. lead-time, complexity, reusability of results, etc.);
- **system metrics**: This is related to the quality of the product (i.e. the integrated system), in terms of fitness to purpose, delivered by the engineering process (i.e. performance, level of flexibility, usability, etc.)

However, it is important to note that inevitably such comparisons will have inherent limitations, stemming from:

(1) the different levels of familiarity (and vested interest) that those conducting experiments will have with the approaches used (e.g. SEW-OSA will always be more familiar to the author than any other approach);

(2) the fact that these tests can never be conducted under identical conditions. That is once the engineering exercise has been completed for one approach, those conducting the experiment will know more about the application, which will add a positive bias to later approaches.

In any case, experiments of the order of magnitude of building a complete integrated system cannot easily be controlled. Although enabling and performing experiments is germane to assessing benefits of a certain approach and should justify or otherwise underlie model-driven approaches, a proper execution of such experiments is well beyond the time and resources apportioned to this Ph.D. research study. Hence, evaluation here is limited to considering how some of the results discussed in this thesis could be related to process metrics and system metrics.

a. Process metrics:

With respect to the concepts and methodologies developed in this research, process metrics can be related to architectural metrics and workbench performance metrics.

Architectural metrics measure the quality of the architectures utilised to support

---

1. A metric is viewed here a fixed parameters against which performance can be measured.

the engineering process. These metrics should be related to desirable features of an architecture as postulated by the IFIP/IFAC Task Force (and discussed in Chapter 1). Examples of such metrics are those associated with dimensions proposed by Bohms [Bohms 1990] and extended by the author [Aguiar 1993e], in order to compare the architectures discussed in Chapter 2, namely: degree of completeness, character, life cycle support covered, degree of prescription, degree of openness, level of heterogeneity, modelling aspects covered, type of framework description, scope of functions embraced, degree of applicability, approach to design, model executability, availability of supporting tools, level of documentation and stand-points embraced. An initial analysis of architectures based upon metrics constitutes one of the pillars of this research. An early study of currently available architectures developed by the author is presented in [Aguiar 1993e].

Workbench performance metrics measure the quality of the CASSE environment developed to support the utilisation of the architecture. They basically qualify the process of using the theoretical formalism of the architecture to address an engineering problem in the real world. Some of the metrics that could be used to compare these environments are: potential for change (i.e. flexibility to adapt to different architectures), consistency within the formalism adopted, usability, comprehensiveness of the implementation, level of support of the implemented facilities (e.g. graphical user interfaces, animation, simulation, report generation, code generation, rapid-prototyping, system configuration, operation, etc.), availability of supporting reference models, reliability and observed design to build lead-time (when applied to a certain problem domain).

b. System metrics.

System metrics measure the quality of the resultant system solution engineered by having applied the architectural formalism through using the workbench. Here, important metrics include:

(1) the ability of the approach to cope with system complexity. Here, complexity can be defined in terms of number and characteristics of system elements (i.e. components and processes), number and character of relationships between elements (i.e. predictable, deterministic, probabilistic or chaotic, homogeneous or heterogeneous), number of system states (i.e. small or large, bound or unbounded and predictable or unpredictable).

(2) the performance of the final system measured in terms of particular features of interest, namely: throughput, lead-time, work-in-progress, level of utilisation of resources and cost, all considering the overhead added by the workbench.

(3) the flexibility engineered in the system in order to enable it to adapt to changes.

When these metrics are examined in the light of the implementation, application and evaluation activities associated with the realisation of SEW-OSA, it can be demonstrated (but not proven) that SEW-OSA enhances the engineering process in relation to current practice.

With regard to the metrics stemming from process metrics, it should be noted that:

- **architectural metrics**. The formalism introduced in the configuration of D2D shop-floor by reference architecture-based models provides a means of organising all the issues pertaining to such a configuration. However, the quality of these models in relation to models which could have been produced through the use of other architectures is arguable.
- **workbench metrics**: the most outstanding feature of SEW-OSA is its ability of support modelling, analysis, simulation, rapid-prototyping, configuration and operation of a system in an integrated manner. The metrics that were obtained to qualify SEW-OSA are those discussed in this chapter.

With regard to the system metrics, it should be noted that:

- In spite of its current limitations in terms of maximum size of a model, SEW-OSA provides a means of approaching complexity in a structured manner. The models discussed in Chapter 8, illustrate the manner in which SEW-OSA can encapsulate the complexity and details embedded in a certain construct, this in order to handle sizeable models. However, in order to actually measure the level of complexity with which SEW-OSA could cope, considerably more complex domains need to be modelled at a very fine level of detail.
- As illustrated by earlier results presented in this chapter, SEW-OSA can provide the support required to perform a range of performance evaluations related to a model. However, the impact that SEW-OSA can have upon the performance of an engineered systems (as compared to other approaches or to current practice) has yet to be proven.
- Finally, although an in depth analysis of flexibility has not been conducted, the ability to change a model and immediately propagate the change to the implemented system is an outstanding feature (i.e. the core competence) of SEW-OSA. This support is particularly strong as a result of a separation of behaviour and functionality and the ability to bind resources to function as late as possible in the life cycle of a system.

11.2. Deliverables

This section briefly summarises what has actually been implemented as a result of activities associated with conceiving and developing SEW-OSA. Where appropriate performance metrics are introduced (e.g. number of lines of code).

11.2.1. Proof of concept experiment

The deliverables produced by this research can be pictorially related to the proof of concept experiment shown in Figure 98. This diagrammatic representation also summarises relationships between the activities of this research and those of other MSI researchers working on the Model-Driven CIM project. The elements inside the shaded bubble were produced as a result of research effort by other MSI researchers, whereas the remaining elements were produced within the scope of this research.

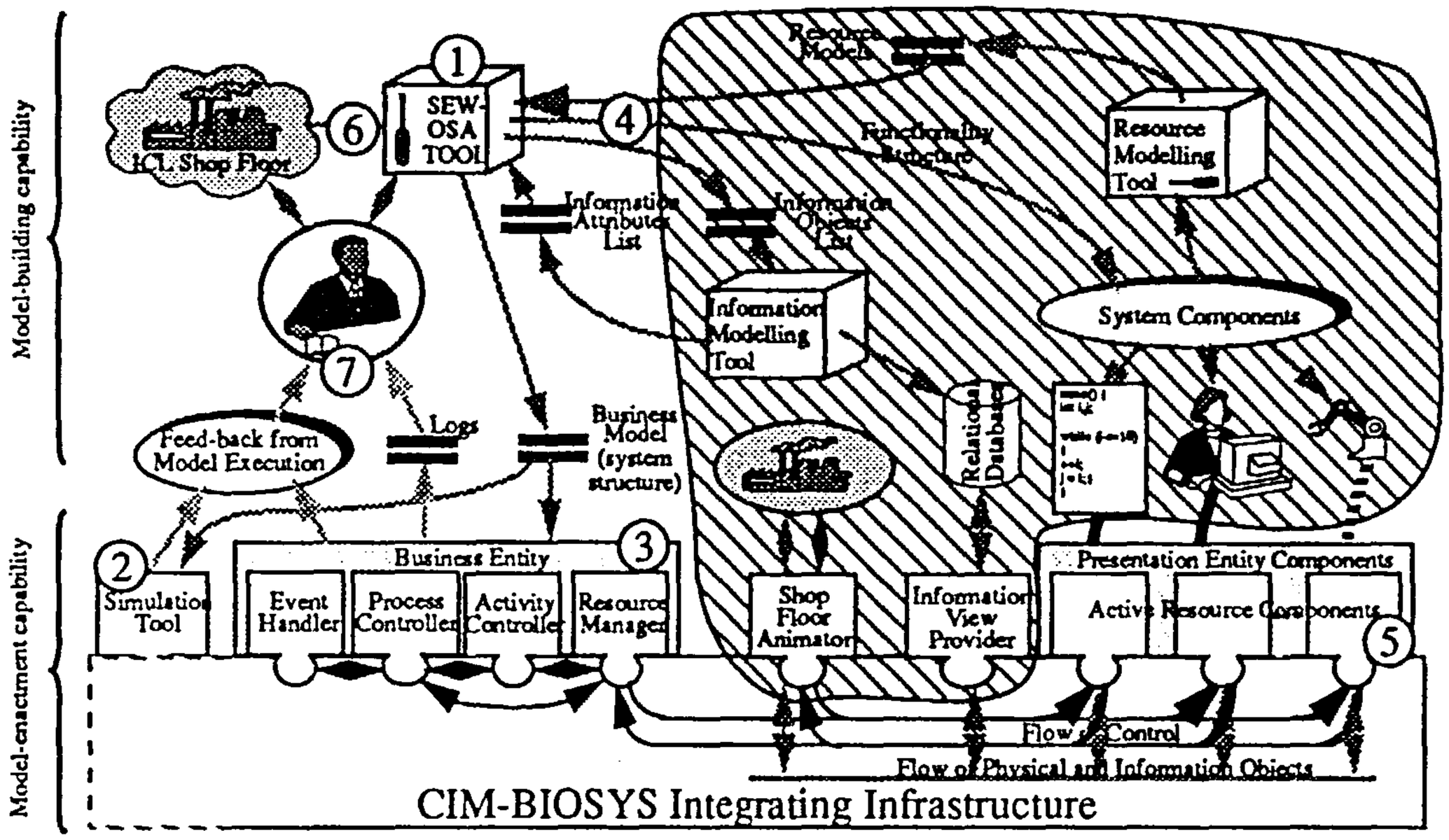


Figure 98 - SEW-OSA: System Engineering Workbench for CIM-OSA

Each element produced by this research is identified in Figure 98 by a tag number (referred to in the text below), namely:

(1) the SEW-OSA CASE tool, the model-building capability which is discussed in Chapter 5.

(2) a link to a Petri-net analyser and simulator, the main features of which are presented in Chapter 6.

(3) a realisation of a business entity for SEW-OSA, as discussed in Chapter 7.

(4) links to other methods, tools and infrastructural elements created within the

"Model-Driven CIM" project (as discussed in Chapter 10), for information and resource modelling which address complementary design issues to those tackled by SEW-OSA.

(5) an outline methodology for replacing emulated resource components by physical ones, this through the specification of a presentation entity for SEW-OSA and an adequate structure for such components (these specifications are presented in Chapters 9 and 10).

(6) a case study of modelling, analysis, simulation and rapid-prototyping, this to investigate the issues related to the coordination of resource components on an electronics manufacturing shop-floor (the activities performed in this case study are discussed in Chapter 8).

(7) an evaluation of the above deliverables, based on performance tests.

Figure 98 is a populated version of the conceptual diagram introduced in Chapter 3 (i.e. Figure 9), based on which the areas of investigation associated with the realisation of SEW-OSA are delineated.

11.2.2. Summary of deliverables

A number of material deliverables have been produced in this study, in the form of modelling tools (i.e. the SEW-OSA CASE tool), infrastructural support elements (i.e. the business entity) and models (i.e. the shop-floor models).

Table 8 summarises the main features of each deliverable, in terms of its scope of life cycle coverage, effort required to build it, and size of its resulting implementation[1].

Table 8 - Main material deliverables[a]

Deliverable	Life-Cycle Coverage	Development Status	Effort (man-weeks [%])	Size (LSC or NC)
workbench integration	CA->OM	α-release	41 [27]	-
SEW-OSA CASE tool	CA/DI	α-release	57 [38]	30221
business entity	DI-OM	α-release	34 [23]	37332
D2D's models	CA-DI	proof of concept	19 [12]	393

a. Legend:
- LSC or NC: lines of source code (for the CASE tool and business entity) or number of constructs (for D2D's models);
- CA, DI and OM: life cycle phases of "conceptual analysis", "design and implementation" and "operation and maintenance", respectively (see Figure 7).

1. Here, it is important to note that the CASE tool size was limited by the maximum number of relationships between constructs and their graphical representations supported by ToolBuilder.

Following is a description of each of these elements.

11.2.3. The SEW-OSA workbench

The SEW-OSA workbench integrates the following elements:

- the SEW-OSA CASE tool, which includes:
  - a CASE tool for modelling the function view, at the requirements definition modelling level;
  - a CASE tool for modelling function and resource views, at the design specification modelling level, and generation of interpreted code for the business entity of SEW-OSA. This tool also incorporates a CASE tool for modelling behaviour with predicate-action Petri-nets based on which code can be generated for a predicate-action execution environment;
  - code generation for a generalised-stochastic time Petri-net analyser and simulator from SEW-OSA models;
- a business entity for SEW-OSA.

The implementation results associated with tools and infrastructural services that comprise SEW-OSA have been discussed in earlier sections. With the exception of the strategy definition phase, the SEW-OSA workbench supports all life-cycle phases of an integrated manufacturing system by formalising key aspects of model-building and model-enactment at each phase; this leading to the generation of executable models which drive (and hence coordinate) system operation upon the CIM-BIOSYS infrastructure.

The SEW-OSA workbench is described as a unified whole to emphasise the effort taken with regard to its conception and integration. Therefore, the figures presented in the "Effort" field in Table 8 refer only to such efforts (i.e. they do not add up the man-power required to develop each individual element of SEW-OSA).

In the author's knowledge, SEW-OSA is the first instance of a comprehensive integrated enterprise engineering environment which provides the basis for a model-driven approach to enterprise integration. SEW-OSA provides a framework into which functionality is expected to be continually added in order to embrace analysis and design aspects of an increasing proportion of the issues involved in the life cycle of an integrated manufacturing system, as shown in Figure 7. Its main tools, infrastructure elements and models, produced as part of this research study are briefly summarised in the following subsections.

11.2.4. CASE tool for requirements definition (i.e. RD tool)

This is a CASE tool for modelling the function view at the requirements definition modelling level (this refers to the upper part of Figure 22). This tool formalises the CIM-OSA recommendations into an organised method that allows users to create models in a graphical and interactive manner. This is in fact the most comprehensive CASE tool available based on the CIM-OSA architecture (this statement has been confirmed by the people involved in the development and validation of CIM-OSA).

Part of the solutions adopted in implementing the method encapsulated in this tool are completely novel. Compared to other tools developed in parallel with this research, this tool is one of the most comprehensive, as well as the most faithful, in terms of conforming to CIM-OSA specifications. The flexibility that resulted from using a meta CASE tool, to develop this tool, also helped in maintaining a potential for change and conformance with the European standards which are likely to emerge based on CIM-OSA specifications.

11.2.5. CASE tool for design specification (i.e. DS tool)

This is a CASE tool for modelling the function and resource views at the design specification modelling level (this refers to the lower part of Figure 22), leading to the automated generation of interpreted code for the business entity.

This tool inherits models produced by the RD Tool. It also operates as a link between that tool and the resource model discussed in Chapter 10, allowing: resource selection and specification; definition of parameters for simulation; and definition of the CIM-BIOSYS configuration.

The majority of the solutions adopted in this tool are completely novel and represent a significant extension of the CIM-OSA methods. The modelling scope covered by this tool embraces the main gaps in CIM-OSA to which SEW-OSA has provided formal support, particularly in terms of associating models of requirements with models of resources.

The DS tool also embraces a CASE tool for modelling behaviour with Petri-nets. This CASE tool enables the generation of code for enterprise activity occurrences and active resource components. It also enables the creation of predicate-action Petri-net models, association of information which transform such a model into a GSTPN and generation of code used by a Petri-net analyser and simulator. This tool can also be viewed as a graphical front-end for the Petri-net analyser and simulator used in this research.

The primary novelty added by the DS tool is in its utilisation to describe the internal behaviour of communicating objects within the context of a CIM system, and its ability to generate code that can actually execute over an integrating infrastructure.

11.2.6. Link to a GSTPN analyser and simulator

The GSTPN simulator is a product developed at the "Laboratory of Control and Micro-Computers/University of Santa Catarina - Brazil", this being produced in part by the author's research work as part of his M. Sc. dissertation [Aguiar 1989]. What this research has added to this tool was an ability to translate a model, built in the CIM-OSA language, into a processable Petri-net that can be interpreted by a simulator. In this respect, SEW-OSA works as a front-end used by the system modeller to overcome the work of tediously modelling directly with Petri-nets. In the author's knowledge, this mapping between CIM-OSA and Petri-nets is the most comprehensive study performed to date on the behavioural constructs of CIM-OSA. Additionally, extensive use of this mapping through case study work has also contributed to validating the approach.

11.2.7. The business entity of SEW-OSA

The business entity of SEW-OSA is a layer of services running upon the CIM-BIOSYS infrastructure which enables the enactment of business models. It is a simplified but original solution (in relation to the specifications of CIM-OSA), providing an implementation of the services required (but not yet fully specified) for the business entity of CIM-OSA. In addition to being able to execute a business model, the business entity operates as a model debugger and performance analyser, enabling the user to check whether run time interactions are working according to that was expected at the modelling stage. This is the first instance of a business entity (known to the author) which is implemented based on the CIM-OSA specifications and runs upon an industrially tested integrating infrastructure.

11.2.8. Model of the SMT assembly line

This model, described in Chapter 8 and evaluated in Section 11.1, consists of a SEW-OSA description of the functional, behavioural and resource aspects of D2D shop-floor with particular emphasis on one of its SMT assembly lines. As it stands, the model can be used to illustrate the use of the SEW-OSA constructs. It can also be used as an input into a reference model, this following a number of additional case studies of other electronic manufacturing companies, so that a generalisation of findings can be carried out.

To date, four published case studies (known to the author) on the application of parts of the CIM-OSA methodology have been developed by a number institutions across Europe. Main distinguishing features of this study (in relation to the others) are that it is the first developed for the electronics manufacturing sector (covering function and resource views) and is the most complete in regard to its modelling facilities, which include: modelling, analysis, simulation, rapid-prototyping, configuration and operation of an integrated system.

11.2.9. Comments on the complexity of the workbench

This section provides an indication of the complexity of the major elements of SEW-OSA, this with respect to "Size" and "Effort" metrics as listed Table 8. It should be noticed from these columns that:

- even though the figure included for the number of lines of source code (i.e. LSC) of the CASE tool also includes support for model-building and generation of various types of code, it is approximately 70 percent larger than the average size of CASE tools previously built at MSI with the Ipsys/ToolBuilder.
- with respect to the implementation effort of about 151 man-weeks, the greatest proportion was dedicated to the realisation of the CASE tool, followed by effort required to integrate all the elements depicted in Figure 98, followed by the realisation of the business entity and finally by conducting the case study activities (which led to the creation of the D2D models). Notwithstanding the greater effort involved in producing the CASE tool, the business entity is the largest software component produced (in terms of lines of codes). However, it should also be noticed that the measure of LSC has been applied for code produced using different modelling languages, as well as code produced via use of tools such as BuilderXcessory [ICS 1991a] [ICS 1991b].
- D2D models do not include intermediate models in other modelling languages.

Further data about the major elements of the SEW-OSA CASE tool and its business entity are presented in Appendix 10. Additionally, thorough documentation of these deliverables as well as the research activities associated with their realisation is presented in a series of articles and internal reports generated by the author (see Appendix 12)[1].

It is important to note that the 'size and 'effort' measures are presented here to provide a *indication of the complexity of the deliverables produced by this research,* and a nominal basis on which to compare the elements of SEW-OSA with each another. It is not the intention to use these metrics as a measure of programming productivity or quality of SEW-OSA elements.

---

1. This documentation consists of three journal publications (additional journal publications are still being written) [Aguiar 1993a] [Aguiar 1993e] [Aguiar 1994b], eleven conference publications [Aguiar 1992e] [Aguiar 1993b] [Aguiar 1993c] [Aguiar 1993d] [Aguiar 1994a] [Aguiar 1994a] [Aguiar 1994c] [Aguiar 1994d] [Aguiar 1994e] [Aguiar 1994f] [Aguiar 1994g] and twelve internal reports [Aguiar 1991a] [Aguiar 1992a] [Aguiar 1992b] [Aguiar 1992c] [Aguiar 1992d] [Aguiar 1992f] [Aguiar 1994h] [Aguiar 1994i] [Aguiar 1994j] [Aguiar 1994l] [Aguiar 1994m] [Aguiar 1995a].

11.3. Architectural Results

The results presented hitherto in this chapter constitute physical instances of results related to achievements represented in the form of architectural results. Architectural results were obtained through analysis and design activities associated with the realisation of SEW-OSA. The main findings which emerged from these activities, are summarised in the following list of contributions to knowledge:

- realisation of a model-building capability, the application of which can lead to the production of a complete business model. Outstanding features of this capability are to: (1) to capture essential definitions of CIM-OSA modelling constructs, (2) organise them in a usable method and (3) enhance them with additional constructs defined by this research.
- the proposition of a methodology for resource specification which enables SEW-OSA to address the gap between *conceptual analysis* and *design and implementation* phases of the life cycle of integrated manufacturing systems.
- the provision of a working solution, in which an initial level of integration is realised between function and information.
- the first instance of an application of CIM-OSA which analyses and simulates relevant design issues of a manufacturing system, thereby demonstrating the integrated use of modelling and simulation as a basis for making design decisions.
- the provision of adequate support for executing simulations, by incorporating a syntax mapping between CIM-OSA and Petri-nets (in the code generation facilities of a CASE tool).
- the first instance of an implementation of a (CIM-OSA-based) business entity capable of enacting business models upon the CIM-BIOSYS integrating infrastructure, thereby enabling the generation of a system prototype based on models common to those used for simulation.
- a consideration of the issues involved in integrating resource components to the SEW-OSA architecture, by (1) defining an adequate structure for the presentation entity of SEW-OSA and (2) investigating how the functionality of active resource components should be organised in order to comply with such a structure.
- the first case study on the application of CIM-OSA in an electronic manufacturing company, and also one of the most complete case studies developed (aimed at validating an engineering environment, whilst providing some insights into how D2D shop-floor improvements can be engineered).

11.4. Concluding Remarks

The portion of D2D shop-floor (i.e. the SMT assembly line) that has been modelled, analysed, simulated, prototyped and configured through the application of the *implementation results* discussed in this chapter is the centrepiece of *case study results*. These results represent an evaluation of the deliverables from this research, as well as an indication of how SEW-OSA can be used with benefit to address engineering processes. This is evidenced in the types of evaluations that SEW-OSA supports, based on an application of its facilities for:

- analysis and simulation, which provide supporting data for a number of design decisions upon possible system configurations;
- rapid-prototyping and its associated features of debugging and performance evaluation, which enable the rapid testing of how the system will actually perform when its components are fully integrated.

Chapter 12 - Conclusions and Issues for Further Investigation

12.1. Summary of the Research Approach[1]

The starting point for this research was a requirement to support enterprise engineering with the aid of modelling technology, in order to enable business integration, this with a view to using IT engineering environments and infrastructures to support integration during downstream life cycle engineering processes. This definition of the problem space was achieved by outlining basic assumptions about the need to attach the formalism of architectures to the life cycle of integrated manufacturing systems.

With respect to this, state-of-the-art architectures and their application were reviewed, leading to: (1) an identification of the need for CASSE environments to support the life cycle of integrated manufacturing systems, and (2) the amalgamation and incorporation of architectures into such environments, as a strategy for fleshing out their formalism.

Possible solutions which address these needs were proposed which embodied the following:

- a structure for the life cycle of integrated manufacturing systems;
- an organisation for the CASSE environment proposed;
- a methodology for realising and evaluating the environment; and
- the materialisation of these issues into a usable workbench.

The organisation of the CASSE environment was based on the adoption of a selection of architectures centred on CIM-OSA. These provided a framework for organising the modelling formalism of the workbench.

Within the structure provided by CIM-OSA, other architectures were introduced to flesh out formalism not supported by CIM-OSA, as well as to provide a means of applying such a formalism to problems found in contemporary manufacturing integration projects. Tools and services associated with these architectures were introduced, as was the rationale for their selection.

Based on the collection of architectures selected, the model-building and model-enactment capabilities of SEW-OSA were realised and applied to a case study in a particular manufacturing industry. Relationships between SEW-OSA and a

---

1. The research described in this thesis was carried out according to the methodology depicted in Figure 8.

collection of sister tools being developed within the "Model-Driven CIM" project evolved and have been defined in this thesis, leading to an identification of a set of requirements for further research.

An evaluation of SEW-OSA was conducted and results obtained, with respect to: (1) performance measures which qualified properties of the workbench, (2) metrics associated with the engineering process supported by the workbench and (3) an analysis of the material deliverables produced.

12.2. Conclusions

12.2.1. Research findings

The research findings which emerged from these activities characterise an examination of primary research issues classified into three classes of objectives, namely: "addressing the research need", "achieving the expected benefits" (for industry) and "enabling the engineering facilities" (in the workbench).

a. Addressing the research need:

Two issues raised (in Chapter 1) were:

- How can the formalism of models and architectures be amalgamated into a usable whole?
- How can such an amalgamated formalism be organised, so that it can be applied to support the life cycle of integrated manufacturing systems?

In seeking answers to these questions, this research has proposed: (1) a structure to organise life cycle processes; (2) the adoption of a selection of architectures to address formalism of part of such processes; and (3) the encapsulation of these architectures into capabilities for model-building and model-enactment.

The research has also tested these proposals by: (1) realising a workbench which provides the capabilities required and (2) evaluating the workbench through case study work. A key architectural result obtained was a combination of CIM-OSA, object-oriented concepts, Petri-nets and the services of the CIM-BIOSYS integrating infrastructure. Collectively, when implemented into an appropriate engineering environment (i.e. SEW-OSA), they can bring closer to industrial reality the use of model-driven formalisms and, hence, help bridge the gap between the modelling (theoretical) and the physical worlds (see Figure 99).

In proving these proposals, this research has addressed some of the major requirements of a reference architecture (listed in Chapter 1) identified by a number of authors (namely: [Williams 1993] [Mize 1992] [Petrie 1992a] [Jorgenson 1992]).

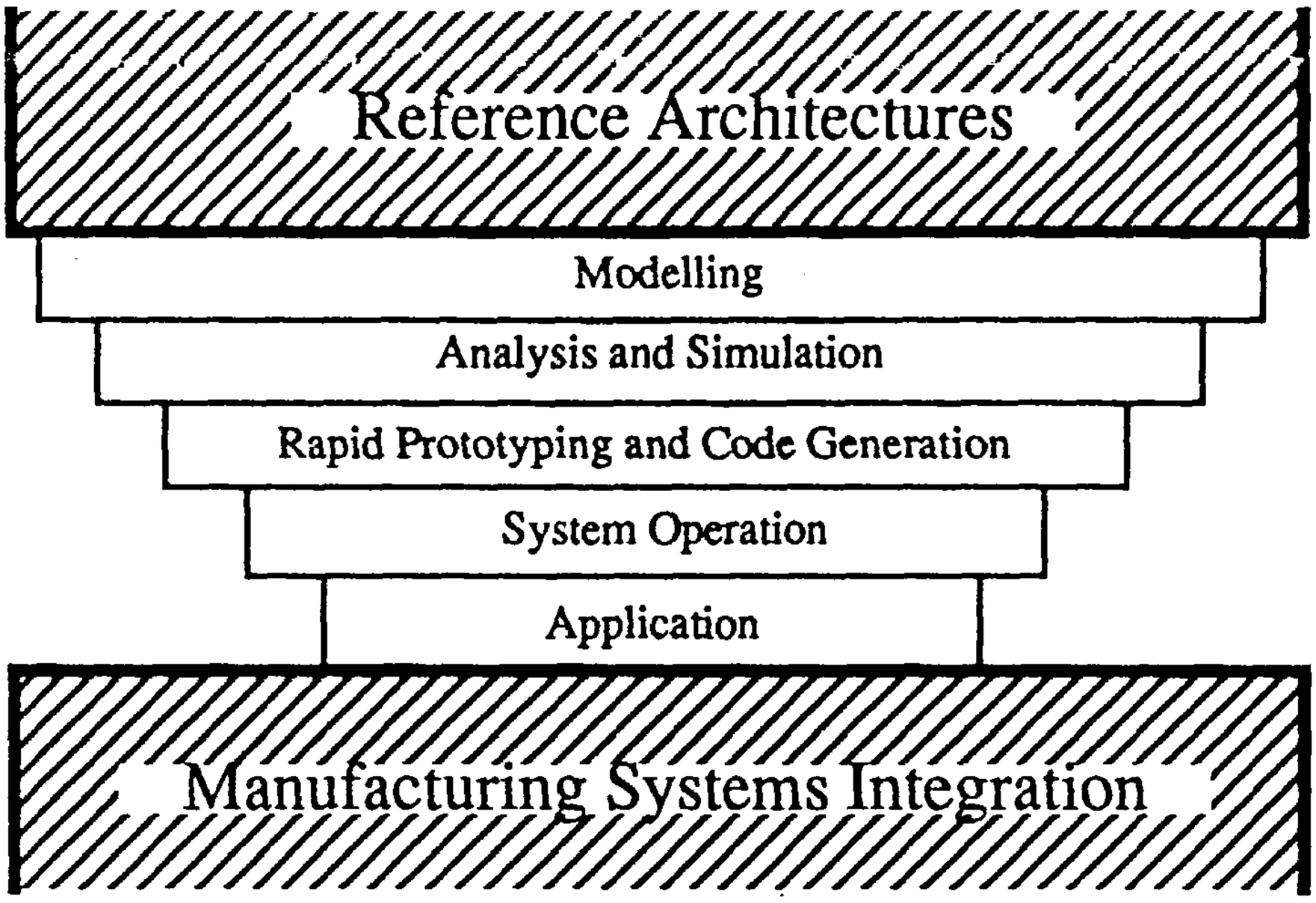


Figure 99 - The 'gap' between the modelling world and the physical world

b. Achieving the expected benefits:

Associated with 'bridging this gap', this research identified a set of anticipated benefits which should be achieved with the introduction of a workbench to support the life cycle of integrated manufacturing systems. Such benefits are: improved consistency; reduced system failure through improved traceability of design decisions; shorter system design-to-build lead-times; and lower costs of integration projects and improved flexibility.

Although these benefits can only be *proven* through use of the workbench in a number of engineering projects in industry, the potential of SEW-OSA to realise those benefits has been *demonstrated* by the results obtained.

c. Enabling engineering facilities:

Such facilities were envisaged to consist of the integrated support for modelling, analysis, simulation, rapid-prototyping, configuration and operation of an integrated manufacturing system. Meeting this requirement stands out as an outstanding feature of the SEW-OSA workbench.

Indeed, a primary contribution of this research is to demonstrate that it is possible to retain an adequate level of formalism, via computational structures and models, which extend through the IMS life cycle from a conceptual description of the system through to actions that the system performs when operating. The underlying methodology which supported this contribution is based on enacting models of

system behaviour which encode important coordination aspects of manufacturing systems. The strategy for demonstrating the incorporation of formalism to the IMS life cycle was to enable the aggregation into a workbench of knowledge of 'what' the system is expected to achieve (i.e. 'problems' to be addressed) and 'how' the system can achieve it (i.e possible 'solutions'). Within the workbench, such a knowledge is represented through an amalgamation of business process modelling and object-oriented modelling approaches which, when adequately manipulated, can lead to business integration.

However, the approach adopted to filling such a 'gap' (as most likely for any approach) can only be expected to partially succeed, due to inherent limitations in our understanding of systems which are as complex as manufacturing enterprises. In a sense, the approach taken in this research implies that attempts to bridge the 'gap' can only be made on an incremental basis and the success of any bridge-head is likely to depend strongly on how well we understand the complexity involved in the classes of systems that need to be supported. A continuous motivation for the author's research is to unravel such complexity.

12.2.2. Concluding remarks

The body of activities developed in this thesis were based on an approach which reflects the strategy adopted to tackling the complexity of enterprise integration. Certain of the assumptions which characterise this approach are:

- **a life cycle approach.** This reflects a view that complexity can be tackled by approaching integration at different levels of abstraction. This is achieved in a manner as to enable the solution to emerge from design decisions made by gradually progressing to lower levels of abstraction.
- **the integrated use of modelling and simulation to support design decisions.**
- **the use of the same models** that *characterise the system* for the purpose of simulation **to derive the system structure** and prototype its components.
- *the* **characterisation process** *is a centre-piece in mapping descriptions of "what" the system is expected to do* and *"how" it does it.* In the case of SEW-OSA, such a process is based on a combination of business process and object-oriented models.
- **the use of CASSE environments to support this process,** thereby capturing the formalism in which the process is expected to be conducted. This implies an assumption that the enterprise (or the integration issues of interest to it) can actually be characterised by models.
- **the ultimate deliverable of the enterprise engineering process is an integrated manufacturing system** structured in such a way that relevant relationships among

its components are captured in models, with the functionality of the components (i.e. the tasks that each of them performs in isolation) separated from it. This implies an assumption that systems can be built from 'off-the-shelf', reusable and inter-changeable components (produced based on agreed 'standards').

- **the physical detail** of communication interfaces and the inherent heterogeneity of computer platforms and infrastructures **can be encapsulated and handled by an integrating infrastructure,** thereby hiding implementation specific details from more abstract enterprise characterisation processes (i.e. focus can be on business integration).
- **business integration requirements can be achieved by enabling coordination** of system activities, this being realised at an appropriate level of granularity as defined by a trade-off between the system wide coordination required (to safeguard the achievement of enterprise goals), and the level at which sub-systems and their components can be directly controlled.
- **reference models can provide suitable indications of the level of coordination appropriate to a certain type of enterprise.** This implies that enterprises can be categorised into types about which generalisations can be made (an issue not fully addressed in this thesis, however).

Alternative perspectives exist within the academic community as to how the complexity of enterprise integration process should be tackled. Two important viewpoints relate to: **the IT perspective** whereby integration is viewed from a systems engineering standpoint; and **the organisational perspective,** whereby integration is viewed as the process of aligning the dominant culture of an organisation with its goals. Although, this research has focused on IT issues, clearly optimal solutions will require consideration of both issues.

Although this thesis has in part fulfilled the requirements established in Chapter 1, the realisation, application and evaluation of SEW-OSA indicates that the complexity of enterprise integration problems may be orders of magnitude greater than can be handled by the formalism of architectures or CASSE environments (which seek to encapsulate such formalisms)[1]. Issues of formalism, coordination and change management are important aspects of realising control over complex systems. In reality, the inherent complexity of the very systems we seek to control can resist our control.

---

1. This is illustrated by the many limitations presented by the (already very) complex tools utilised in this research(e.g. maximum complexity of a CASE tool built with ToolBuilder and maximum model size supported by CIM-BIOSYS and ARP).

Emerging ideas on complexity (and chaos theory) are demonstrating our inability to control certain natural and man-made systems, integrated manufacturing enterprises being typical examples of largely man-made systems.

Perhaps, the approach defined and developed in this thesis for *enacting business process models in support of the integrated manufacturing system life cycle* is yet another attempt to establish control over inherently uncontrollable systems. However, if it leads to improvement over current practice (and hence a competitive advantage for enlightened manufacturers), it may well prove to be an important step forward.

12.3. Issues for Future Investigation

Two sets of issues are considered here: (1) enhancements to SEW-OSA, in order to overcome certain limitations identified in previous chapters; and (2) extensions of SEW-OSA, by incorporating in it a second selection of architectures.

12.3.1. Enhancements of SEW-OSA

a. Implementation issues

This future effort would address limitations of SEW-OSA which can be overcome through additional implementation effort. Such limitations are primarily due to: (1) simplifications with respect to the CIM-OSA specifications in order to make their realisation viable within the resources and time-scales available; (2) additional programming effort required to make the workbench more robust and improve its usability.

b. Case study issues

This concerns various factors that should be considered if: (1) an industrial implementation of the approach is to be carried out by D2D, based on the recommendations proposed by this research; and (2) further simulation and rapid-prototyping tests are required.

c. Architectural issues

Architectural issues are related to: (1) certain limitations listed in the previous chapters; and (2) the study of how links to other architectures can be realised.

12.3.2. Extension to SEW-OSA

Extending SEW-OSA could embrace two different sets of activities. The first set would focus on extending its life-cycle coverage. The second set would add certain capabilities to SEW-OSA, in order to expand its natural domains.

a. Extending SEW-OSA

Investigating underlying formalism which can traverse different life cycle phases of integrated manufacturing systems represents an area for long term research, to which the author wishes to actively contribute. One of the prime challenges connected to this task is expected to arise at the interface between *strategy definition* and *conceptual analysis*, as well as within the *strategy definition* phase itself, where difficulties of encoding intuitive processes exist.

b. Expanding SEW-OSA

Research areas identified during this research with respect to expanding SEW-OSA include:

- **Incorporation of cost models into SEW-OSA:** This activity would combine methods such as "Activity Based Costing" [Shaharoun 1994] [Turney 1991] with the SEW-OSA business models to provide powerful tools for metric-based decision support (applied at different levels of abstraction within an enterprise).
- **Integration of organisational and IT issues:** Here, investigation is required of the research issues associated with enterprise design from an organisational perspective and integration of such issues with those connected with IT perspectives.
- **Reference models for SEW-OSA:** Activity is required to populate SEW-OSA with reference models of requirements (i.e. system models) and solutions (i.e. resource models), thereby enabling the creation of a more complete design environment for systems design which can take full advantage of the reusability of resources and models. Here, a crucial factor is to structure reference models in a form which can render them widely applicable.
- **Application of SEW-OSA in other domains:** The application of SEW-OSA needs to be proven in other domains. For example, it could be used to facilitate the functional re-organisation of a number of inter-related software packages and applied to particular business processes (e.g. the PI process) as a means of achieving software interoperability.
- **Combination of SEW-OSA with alternative federated architectures:** As earlier discussed, the federated architectures approach to enterprise integration is distinct from that adopted by CIM-OSA (and, consequently, SEW-OSA). A research topic strongly advocated within the work of the US/Europe Enterprise Integration Initiative concerns integration between unified and federated architectures.

List of References

[Adiga 1993] ADIGA, S. *Object-oriented software for manufacturing systems*. Chapman & Hall, USA, 1993.

[Aguiar 1989] AGUIAR, M. W. C. *Comparative performance analysis of FIP and PROFI-BUS, candidates to a field-bus standard, using generalised stochastic time petri-nets*. Universidade Federal de Santa Catarina, Brazil, 1989.

[Aguiar 1991a] AGUIAR, M. W. C. *On the Life Cycle of Integrated Manufacturing Systems - Research Plan Outline. Document no. 1, version 1.1*, Loughborough University, England, 1991.

[Aguiar 1992a] AGUIAR, M. W. C. *First Exercise on the Application of the CIM-OSA Modelling Methodology - Working Group 3 - ICL/LUT-SIG Joint Work On CIM-OSA. Document no. 2, version 1.0*, Loughborough University, England, 1992.

[Aguiar 1992b] AGUIAR, M. W. C. *Data Collection on the SMD Assembly Line - Working Group 3 - ICL/LUT-SIG Joint Work On CIM-OSA. Document no. 3, version 1.0*, Loughborough University, England, 1992.

[Aguiar 1992c] AGUIAR, M. W. C. *Data Gathering on the SMD Assembly Line - Third Interview- Working Group 3 - ICL/LUT-SIG Joint Work On CIM-OSA. Document no. 4, version 1.0*, Loughborough University, England, 1992.

[Aguiar 1992d] AGUIAR, M. W. C. *Study on Reference Architectures for CIM Systems Design and Building Based on Executable Models. Master of Philosophy Report*, Loughborough University, England, 1992.

[Aguiar 1992e] AGUIAR, M. W. C. Case study on the application of the CIM-OSA methodology for manufacturing process modelling. *Proceedings of the European Simulation Symposium*. Germany, 1992.

[Aguiar 1992f] AGUIAR, M. W. C. *Petri-net Case Tool Documentation. Document no. 5, version 1.0*, Loughborough University, England, 1992.

[Aguiar 1993a] AGUIAR, M. W. C.; WESTON, R. H. CIM-OSA and stochastic time petri nets for behavioural modelling and model handling in CIM systems design and building. *Proceedings of the Institution of Mechanical Engineers Part b - Journal of Engineering Manufacture*, 1993, vol. 207, no, 3, pp. 147-158, England.

[Aguiar 1993b] AGUIAR, M. W. C.; WESTON, R. H. CIM-OSA and time petri nets for CIM systems modelling and simulation. *Proceedings of the International Conference on Factory Automation and Information Management - FAIM/93*. Ireland, 1993.

[Aguiar 1993c] AGUIAR, M. W. C.; WESTON, R. H. Reference architectures for enterprise integration. *Proceedings of the International Conference on Computer Aided Manufacturing, Robotics and Factories of the Future CARS-FOF/93*. USA, 1993.

[Aguiar 1993d] AGUIAR, M. W. C.; WESTON, R. H. Model based approach supporting the life cycle of integrated manufacturing enterprises enterprise integration. *Proceedings of the International Conference on Computer Integrated Manufacturing - ICCIM/93*. Singapore, 1993.

[Aguiar 1993e] AGUIAR, M. W. C. And Weston, R. H. A model driven approach to enterprise integration. *International Journal of CIM*. England, 1993.

[Aguiar 1994a] AGUIAR, M. W. C.; WESTON, R. H.; COUTTS, I. C. Manufacturing systems design and implementation based on formal modelling methods and tools. *Proceedings of the 27th International Symposium on Automotive Technology and Automation (ISATA)*. Germany, 1994.

[Aguiar 1994b] AGUIAR, M. W. C.; COUTTS, I.; WESTON, R. H. Rapid prototyping of

open software systems. *Proceedings of the European Simulation Multi-confcrence*. Spain, 1994.

[Aguiar 1994c] AGUIAR, M. W. C.; COUTTS, I.; WESTON, R. H. Workbench for rapid prototyping of open software systems. Submitted to the *International Journal of Manufacturing Systems Design*. England, 1994.

[Aguiar 1994d] AGUIAR, M. W. C.; WESTON, R. H. SEW-OSA - Systems Engineering Workbench centred on CIM-OSA. *Proceedings of the 10th International Conference on Computer Aided Manufacturing, Robotics and Factories of the Future CARS-FOF/94*. Canada, 1994.

[Aguiar 1994e] AGUIAR, M. W. C.; WESTON, R. H. The business entity of SEW-OSA - Systems Engineering Workbench centred on CIM-OSA. *Proceedings of the 10th National Conference on Manufacturing Research*. England, 1994.

[Aguiar 1994f] AGUIAR, M. W. C.; WESTON, R. H. Petri-Nets in SEW-OSA - Systems Engineering Workbench centred on CIM-OSA. *Proceedings of the 1994 International Conference on Systems, Management and Cybernetics*. USA, 1994.

[Aguiar 1994g] AGUIAR, M. W. C.; COUTTS, I.; WESTON, R. H. Petri-Nets in SEW-OSA - Systems Engineering Workbench centred on CIM-OSA. *Proceedings of the First IMSE*. France. 1994.

[Aguiar 1994h] AGUIAR, M. W. C. *Notes from the case study application of SEW-OSA to D2D shop-floor. Internal Report*, MSI Research Institute. England, 1994.

[Aguiar 1994i] AGUIAR, M. W. C. *SEW-OSA (Systems Engineering Workbench centred on CIM-OSA) Integrated CASE Tools - Printout of the tools developed with IPSYS/ToolBuilder. Internal Report*, MSI Research Institute, Loughborough University, England, 1994.

[Aguiar 1994j] AGUIAR, M. W. C. *SEW-OSA (Systems Engineering Workbench centred on CIM-OSA) Business Entity - Printout of the software code. Internal Report*, MSI Research Institute, Loughborough University, England, 1994.

[Aguiar 1994l] AGUIAR, M. W. C. *Collection of five reports to CAPES on the progress of the PhD research work - issued between 1991 and 1994*. England, 1994.

[Aguiar 1994m] AGUIAR, M. W. C. *Preliminary review of methods and techniques to support business strategy development - working paper*, Loughborough University, England, 1994.

[Aguiar 1995a] AGUIAR, M. W. C. *A proposal for linking SEW-OSA with a bottom-up modelling tool. Internal Report*, MSI Research Institute, Loughborough University, England, 1995.

[Akif 1991] AKIF, H. C.; Doumeings, G. Computer aided GRAI method (C. A. GRAI) .*Proceedings of the Conference in Advances in Production Management Systems - IFIP/91*. Netherlands: Elsevier Science Publishers B. V., 1991, pp. 283-292.

[AMR 1991] ADVANCED MANUFACTURING RESEARCH. *Manufacturing application source book - Application enabler. Internal Report*, AMR, Singapore, 1991.

[Alderson 1991] ALDERSON, A. Meta-CASE technology. Lecture Notes in Computer Science Software Development Environments and CASE Technology. *Proceedings of European Symposyum*. pp. 81-91. Springer-Verlag. Germany. 1991.

[Back 1986] Back, M. J. *The design of the UNIX operating system*.USA: Prentice Hall, 1986. ISBN 0-13-201757-1 025.

[Bailin 1989] BAILIN, S. C. An object-oriented requirements specification method. *Communications of the ACM*, 1989, vol. 32, no. 5, pp. 608-623.

[Beeckman 1989] BEECKMAN, D. CIM-OSA: computer integrated manufacturing - open

system architecture. *International Journal of CIM*, 1989, vol.2, no.2, pp.94-105.

[Bernus 1994] BERNUS, P.; NEMES, L. A framework to define a generic enterprise reference architecture and methodology. *Proceedings of ICARV'94*. Singapore, 1994.

[Biemans 1990] BIEMANS, F. *Manufacturing planning and control - a reference model*. Netherlands: Elsevier Science Publishers B. V.,1990.

[Bishop 1989] BISHOP, T.; SCHOFIELD, N. *Unlocking the potential for CIM - a management guide*. P. A. Consulting Services. UK, 1989.

[Blinn 1991] BLINN, T.; MAYER, R. J. CULLINANE, T. A service based approach to information integration in concurrent engineering environments. *Proceedings of Autofact 91*. USA, 1991, pp. 7.27-7.38.

[Bohms 1990] BOHMS, H. M. RIA: Reference Model for Industrial Automation. *Proceedings of CIMCON'90*. USA: NIST special publication 785, 1990.

[Boldyreff 1994] BOLDYREFF, C. *Software engineering design: a paradigm case of CSCW - Design issues in CSCW*, Springer-Verlag, pp. 140, 1994.

[Bonney 1992] BONNEY, M. C.; BARSONN, R.J.; HEAD, M.A. et al. *UNISON - A tool for enterprise integration. Internal report*. Department of Manufacturing Engineering & Operations Management. University of Nottingham, UK, 1992.

[Booch 1991] BOOCH, G. *Object Oriented Design with Applications*. USA: Benjamin/ Cummins, 1991.

[Boswell 1992] BOSWELL, L. Experiences with building a multimedia mail system prototype on the OSF DCE platform. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Boucher 1990] BOUCHER, T. O.; JAFARI, M. A. The design of a petri net controller from an IDEF0 specification. *Proceedings of the Conference on Factory Automation and Information Management - FAIM/90*. Ireland: CRC Press, 1990.

[Boucher 1991] BOUCHER, T. O.; JAFARI, M. A. The design of a petri net controller from an IDEF0 specification. *The Automated Factory Handbook - Technology and Managemen*t. USA: TAB Professional and Reference books, 1991, pp. 804-815.

[Boykin 1990] BOYKIN, R. E. 1990, CAM-I CIM reference model - historical reflection. *Proceedings of the 1990 CIM Conference - CIMCON 90* . USA: NIST,1990, pp. 35-41.

[Boyle 1991] BOYLE, A. *Express usage guide - external representation of product definition data - ISO TC184/SC4/WG5*. Internal Document, 1991.

[Brathwaite 1990] BRATHWAITE, K. S. *Applications development using CASE tools*. USA: Academic Press, Inc.1990.

[Bravoco 1985a] BRAVOCO, R. R.; YADAV, S. B. A methodology to model the functional structure of an organisation. *Computers in Industry*, 1985, vol. 6, pp. 345-361.

[Bravoco 1985b] BRAVOCO, R. R.; YADAV, S. B. A methodology to model the information structure of an organisation. *The Journal of Systems and Software*, 1985 ,vol. 5, pp. 59-71.

[Bravoco 1985c] BRAVOCO, R. R.; YADAV, S. B. A methodology to model the dynamic structure of an organisation. *Information Systems*, 1985, vol. 10, no. 3, pp. 299-317.

[Brenner 1987] BRENNER, J. B. Open distributed processing. *ICL Technical Journal*, 1987, vol.5, no.4, pp.613-637.

[Brown 1993] BROWN, G. H. *Application Development - OPENframework Systems*

Architecture. England: Prentice Hall, 1993.

[Bruno 1994] BRUNO, G. et al. Making CIMOSA operational. Submitted to the *Proceedings of the First IMSE*. France, 1994.

[Brunt 1992] BRUNT, R.; HUTT, A. ed. *OPENframework - The systems architecture: an introduction - International Computers Limited*. USA: Prentice Hall, 1992.

[BSL 1991] BUSINESS SERVICES LTD. *New Product Introduction into Manufacture at ICL Ashton and ICL Kidsgrove Manufacturing Facilities - Final Draft Report*. University of Salford. July 1991.

[Buckanan 1994] BUCKANAN, D. *Business process analysis and business process re-engineering: Loughborough University of Technology - EPSRC innovative manufacturing initiative. Position Statement*, England, 1994.

[Carrie 1993] CARRIE, A. et. al. Linking strategy to production management structures and systems. *Proceedings of the Conference on Managing Integrated Manufacturing - Organization, Strategy & Technology*. England: KAMG - Keele University, 1993, pp. 51-68.

[Carswell 1987] CARSWELL J. L. JR.; NAVATHE, S. B. SA-ER: a methodology that links structured analysis and entity-relationship modelling for database design. *Proceedings of the Conference on Entity-Relationship Approach*. Netherlands: Elsevier Science Bulisher, 1987, pp. 381-396.

[CEN 1994a] CEN/TC 310/WG1. *CIMOSA Document R0493/0*. Denmark, 1993.

[CEN 1994b] CEN/TC 310/WG1. *Contributions from QCIM and CIM-OSA - Comparison CIM-OSA - IEM Modelling Constructs*. Document no. 41. Denmark, 1994.

[Checkland 1981] CHECKLAND, P. *Systems thinking, systems practice*. UK: John Wiley & sons, 1981. ISBN 0-471-27911-0 [003 CHE].

[Chen 1976] CHEN, P. P.-S. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1976, vol. 1, no. 1, pp. 9-36.

[Childe 1993] CHILDE, S.; BENNETT, J.; MAULL, R. Manufacturing re-engineering around business processes. *The International Conference on Managing Integrated Manufacturing - Organization, Strategy & Technology*. England: KAMG - Keele University,1993.

[Clark 1989] CLARK, G. M. ; WITHERS, D. H. Architecture for an Integrated Simulation/CIM Systems. *Proceedings of the 1989 Winter Simulation conference*. USA: IEEE, 1989, pp.942-948.

[Clements 1993] CLEMENTS, P.; COUTTS, I.; WESTON, R. H. A life-cycle support environment comprising open systems manufacturing modelling methods and the CIM-BIOSYS infrastructure tools. *MAPLE'93 - Symposium on Manufacturing Automation Programming Language Environments*. Ottawa, 1993.

[Coad 1990] COAD, P.; YOURDON, E. *Object-oriented analysis*. USA: Prentice-Hall, 1990.

[Colquhounn 1991] COLQUHOUNN, G. J.; BAINES, R. W. A generic IDEFo model of process planning. *Int. J. Prod. Res.*, 1991, vol.29, no.11, pp.2239-2257.

[Colquhoun 1994] COLQUHOUN, G. J.; BAINES, R. W. A "process description" approach to manufacturing system modelling. *Proceedings of the Tenth National Conference on Manufacturing Research*. England: Taylor & Francis, 1994, p360-364.

[Coutts 1992] COUTTS, I. A.; WESTON, R. H.; MURGATROYD, I. S.; GASCOIGNE, J. D., "Open Applications within Soft Integrated Manufacturing Systems", *Proceedings of the International Conference on Manufacturing Automation*, Hong Kong 1992.

[Coutts 1994] COUTTS, I. A. *MSI application generation libraries vol. 0.2 - Internal Document*. England, 1994.

[Cox 1986] COX, B. J. *Object-oriented programming: an evolutionary approach*. USA: Addison Wesley, 1986.

[Davenport 1994] DAVENPORT, T. H. Saving IT's soul: human-centred information management. *Harvard Business Review*, March-April 1994, pp.119-131.

[David 1994] DAVID, R. AND ALLA, H. Petri-nets for modeling of dynamic systems - a survey. *Automatica*. 1994, vol. 30, no. 2, pp. 175-202.

[Davis 1991] DAVIS, L. *Manufacturing systems integration project: Organisational aspects of information systems design*. Loughborough University, 1991.

[DEC 1987] DEC (Digital Equipment Incorporation); PHILIPS (Netherlands Philips Bedrijven BV). *Reference model of production systems version 1.0. CFT Report 13/87*. Netherlands: Philips, 1987.

[DEC 1991] DEC (Digital Equipment Incorporation). *Open Systems Handbook: a guide to building open systems* DEC. USA, 1991.

[DEC 1992] DEC (Digital Equipment Incorporation). Network Application Support (NAS): an open system for application integration, support documentation, 1992.

[Deignam 1993] DEIGNAN, F.; HOLLINGSWORTH, D. *Networking Services - OPENframework Systems Architecture*. England: Prentice Hall, 1993.

[Devapriya 1991] DEVAPRIYA, D. S.; DESCOTES-GENON, B.; LADET, P. A Petri Net based blackboard type architecture for FMS control (ESCOPE). *Proceedings of the Conference on Advances in Production Management Systems - IFIP 91*. Netherlands: Elsevier Science Publishers B.V., 1991, pp.531-541.

[Didic 1992] DIDIC, M. Rapid prototyping for MAP/MMS based CIM-OSA enviroments. *Proceedings of the 3rd Int. Workshop on Rapid System Prototyping*. USA: IEEE, 1992, pp.221-233.

[Didic 1993] DIDIC, M.; NEUSCHELER, F.; BOGDANOWICZ, L. et al. McCIM: Execution of CIMOSA models. *CIM Europe 93*.

[DIN 1988] DIN. *The standardisation of interfaces for CIM- current state of development and future requirements - DIN Technical Report no. 15*. German Standards Institute (DIN). Germany, 1988.

[Doumeingts 1992] DOUMEINGTS, G.; CHEN, D.; MARCOTTE, F. Concepts, models and methods for the design of production management systems. *Computers in Industry*, 1992, vol.19, pp.89-111.

[Edwards 1993] EDWARDS, J. *A reference architecture for flexibly integrating machine vision within manufacturing systems. PhD Thesis*. Loughborough University, England, 1993.

[Emond 1988] EMOND, J. C. CIM-OSA: Key concepts overview and DEMONSTRATION. *Proceedings of the ESPRIT '88 Conference*. Amsterdam: North-Holland, 1988, vol. 2, pp.1509-1527.

[Endres 1991] ENDRES, A.; WEBER, H. eds. Lecture notes in computer science. *Proceedings of the European Symposium on Software Development Enviroments and CASE Technology*. Espringer-Verlag, 1991, pp.1-255.

[Espejo 1989] ESPEJO, R; HARNDEN, R. ed. *The viable system model - interpretations and applications of Stafford Beer's VSM*. UK: John Wiley & Sons, 1989. ISBN 0-471-92288-9 [003 VIA].

[ESPRIT/AMICE 1987a] ESPRIT/AMICE. *CIM-OSA: a primer on key concepts and purpose - Project 688*. USA: AMICE & APT, 1987.

[ESPRIT/AMICE 1987b] ESPRIT/AMICE. *CIM-OSA: strategic management and design issues - Project 688*. USA: AMICE, 1987.

[ESPRIT/AMICE 1989] ESPRIT/AMICE. *CIM-OSA research reports, project 688, volume 1. Open system architecture for CIM*. Springer-Verlag, Netherlands, 1989.

[ESPRIT/AMICE 1991a] ESPRIT/AMICE. Integrated manufacturing - a challenge for the 1990s. *Computing & Control Engineering Journal*, 1991, vol. May, pp.101-125.

[ESPRIT/AMICE 1991b] ESPRIT/AMICE. *CIM-OSA Architecture Description, AD 1.0*. Belgium: ESPRIT Consortium AMICE, 1991.

[ESPRIT/AMICE 1991c] ESPRIT/AMICE. *CIM-OSA releases: project proposal - Project 5288*. Belgium, 1991.

[ESPRIT/AMICE 1991d] ESPRIT/AMICE. *CIM-OSA: aerospatiale F.M.S. case study. Function view at requirements definition modeling level - Project 5288*. Belgium, 1991.

[ESPRIT/AMICE 1991e] ESPRIT/AMICE. *CIM-OSA seminar. IMechE - Computing and Data Communications Group*. 1991.

[ESPRIT/AMICE 1991f] ESPRIT/AMICE. *Computer-Aided Enterprise Engineering (CAEE) tool - user guide. Document R0320/0 - version 2.1*. 1991.

[ESPRIT/AMICE 1992a] ESPRIT/AMICE. *CIM-OSA user guide - system requirements definition - Project 5288*. 1992.

[ESPRIT/AMICE 1992b] ESPRIT/AMICE. *CIM-OSA formal reference base. version 0.3*. 1992.

[ESPRIT/AMICE 1993a] ESPRIT/AMICE. *CIM-OSA formal reference base*. 1993.

[ESPRIT/AMICE 1993b] ESPRIT/AMICE. *CIM-OSA research reports, project 688, volume 1. Open system architecture for CIM*. Springer-Verlag, 1983.

[ESPRIT/VOICE 1992] ESPRIT/VOICE. *Validating OSA in industrial CIM enviroments. Objetives and results - EP 5510*. 1992.

[ESPRIT/VOICE 1993] ESPRIT/VOICE. *Validation of CIM-OSA (Open Systems Architecture) - A joint ESPRIT projects report*. 1993.

[Fairthorne 1993] FAIRTHORNE, B. *Security - OPENframework Systems Architecture*. England: Prentice Hall, 1993.

[Farines 1992] FARINES, J. M.; CURY, J. E. R.; CARDOSO, J. Um sistema de coordenação para ambientes fabris baseado no modelo rede de Petri. *Anais do 9º Comgresso Brasileiro de Automática*. Vitória - ES, setembro de 1992.

[FIAT 1994a] FIAT. *CIMOSA-WP1: The model of a gearbox production system. Fiat Internal Paper*, 1994.

[FIAT 1994b] FIAT. *WP1: An industrial application of CIMOSA: Fiat gearbox production system. Copies of overheads*, 1994.

[Fowler 1964] FOWLER, H. W.; FOWLER, F. G. *The concise Oxford dictionary of current english*. 5ed., England, 1964.

[Fox 1992] FOX, M. S. The TOVE project: towards a common-sense model of the enterprise. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Fox 1993] FOX, M. S. and Gruninger, M. Ontologies for enterprise integration. *Proceedings of the Second Conference on Cooperative Information Systems*. Canada, 1994.

[Fritsch 1989] FRITSCH, C. A. Information dynamics for computer integrated product realization. In: S. Y. NOF and C. L. MOODIE, ed. *Advanced Information Technology for Industrial Material Flow Systems*. Germany: Springer-Verlag, 1989.

[Frizelle 1991] FRIZELLE, G. D. M. Deriving a methodology for implementing CAPM systems. *International Journal of Operations and Production Management*, 1991, vol. 11, no. 7, pp.6-26.

[Fulton 1992] FULTON, J. A. Enterprise integration using the semantic unification meta-model. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Gaches 1994] GACHES, R. *CIMtool presentation. Cim tool overview. Catalog*, 1994.

[Gale 1993] GALE, T. *Systems Management - OPENframework Systems Architecture*.

England: Prentice Hall, 1993.

[Garnousset 1989] GARNOUSSET, H. E.; FARINES, J. M.; CANTU, E. Efficient tools for analysis and implementation of manufacturing systems modelled by Petri Net with objects: a production rules compilation-based approach. *Proceedings of IECOM 89*. IEEE Publisher, 1989, pp.543-549.

[Gascoigne 1992] GASCOIGNE, J.D. *CIM-BIOSYS, its purpose and functional overview, SI Group Internal Document, available from Manufacturing Eng. Dept., Loughborough University of Technology*, England, England, 1992.

[Gilders 1991a] GILDERS, P.; LOMAS, A. *Requirements specification for production planning and control demonstration system - internal report version 1.* Loughborough University of Technology, England, 1991.

[Gilders 1991a] GILDERS, P. *Guide to writing CIM-BIOSYS system applications - internal report v1*. Loughborough University of Technology. England. 1991.

[Gilders 1995] GILDERS, P.J., WESTON, R.H. A mechanism for the rapid generation of Interaction Functionality for CIM Systems based on Estelle. (To be submitted to the) *Computing and Control Engineering Journal*, 1995.

[Goldratt 1984] GOLDRATT, E. *The goal - the process of on going improvement.* USA, 1984.

[Goodwin 1994] GOODWIN, C. Mapping middleware. *Unix News*. England, 1994.

[Goranson 1992] GORANSON, H. T. Services in the SIRIUS-BETA inter-integration domain. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Graefe 1989] GRAEFE, U.; THOMSON, V. A reference model for production control. *Int. J. Computer Integrated Manufacturing*, 1989, vol. 2, no. 2, pp. 86-93.

[GRAI 1984] GRAI Laboratory. *Study of the conceptual information model for an advanced factory management system defined for two work centres: inspection and tooling. Final Report - R-84-FM-01*. France, 1984.

[Hales 1989] HALES, H. L. *CIMPLAN The Systematic Approach of Factory Automation. CIM Strategies.* Cutter Information Corp., USA, 1989.

[Hales 1990] HALES, H. L. *CIM Strategy for Implementation.* SME Course. USA, 1990.

[Hammer 1994] HAMMER, M.; CHAMPY, J. *Re-engineering the corporation - a manifesto for business revolution.* England: Nicholas Brealey Publishing, 1994.

[Harhalakis 1989] HARHALAKIS, G., LIN, C. P., MARK, L. A knowledge-based prototype of a factory level CIM system. *Computer Integrated Manufacturing Systems*, 1989, vol.2, no.1, pp.11-20.

[Hars 1991] HARS, A., et al. Concept of current data modelling methodologies. Report on CODE - ESPRIT Project. Germany, 1991.

[Hatono 1989] HATONO, I.; KATOH, N.; YAMAGATA, K. et al. Modeling of FMS under uncertainty using stochastic Petri Nets - An application to rule-based on-line scheduling. *Proceedings of the 3rd. International Workshop Petri Nets and Performance Models - PNPM 89*. IEEE Publisher, 1989, pp.122-129.

[Heller 1991] HELLER, D. Motif programming manual. The definitive guides to the X Windows System vol. 4. USA: O'Reilly & Associates, Inc.,1991. ISBN 0-937175-70-6.

[Hinch 1988] HINCH, S. W. *Handbook of surface mount technology*. England: Longman Scientific & Technical,1988.

[Hou 1993] HOU, W-N.; TRAUBOTH, H. An approach to the development of the machine front-end services in a CIM-OSA enviroment. *Proceedings of the 3rd. Joint Int. FAIM'93 Conf.* Ireland,1993.

[Howard 1994] HOWARD, P.; POTTER, C. CASE and methods based development tools: an evaluation and comparison. Ovuum and Butler Bloor (independent

consultants), England, 1994.

[Hsu 1990] HSU, C.; RATTNER, L. Information modelling for computerized manufacturing. *IEE Transactions on Systems, Management and Cybernetics*, 1990, vol. 20, no. 4, pp. 758-776.

[Huhns 1992] HUHNS, M. N. et. al. Enterprise information modeling and model integration in CARNOT. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Hutchison 1991] HUTCHISON, D. DTI/SERC open distributed systems architecture research programme. *Computing and Control Engineering Journal*, 1991, vol.2, no.6, pp.250-252.

[Hutt 1993a] HUTT, A. *Usability - OPENframework Systems Architecture*. England: Prentice Hall,1993.

[Hutt 1993b] HUTT, A. *User Interface - OPENframework Systems Architecture*. England: Prentice Hall, 1993.

[ICL 1988] ICL. *Kidsgrove Manufacturing Operations Systems Strategy - Issue no. 3 - Revised*. February 1988.

[ICL 1993] ICL. *ProcessWise. Technical catalogues*. 1993.

[ICL 1994] ICL. *DAIS. Technical catalogues*. 1994.

[ICL/OFD 1993] ICL/OPEN FRAMEWORK DIVISION. *OPENframework - overview of methods* - issue 1.2. England, 1993.

[ICL/OFD 1994] ICL/OPEN FRAMEWORK DIVISION. OPENframework General Practicitioners Course - handouts. England, 1994.

[ICS 1991a] INTEGRATED COMPUTER SOLUTIONS, INC. *The Builder Xcessory User's Guide. version 2.0*. USA, 1991.

[ICS 1991b] INTEGRATED COMPUTER SOLUTIONS, INC. *The Builder Xcessory Reference. version 2.0*. USA, 1991.

[Ipsys 1992a] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder version 1.3 - Methods and Concepts Manual*. England, 1992.

[Ipsys 1992b] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder versio. 1.3 - Instruction Manual*. England, 1992.

[Ipsys 1992c] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder version1.3 - Implementor's Reference Manual*. England, 1992.

[Ipsys 1992d] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder version 1.3 - Guide to a Generated Tool*. England, 1992.

[Ipsys 1992e] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder version 1.3 - Handbook*. England, 1992.

[Ipsys 1992f] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder version 1.3 - Data Transformation Reference Manual*. England, 1992.

[Ipsys 1992g] IPSYS SOFTWARE P.C. *IPSYS ToolBuilder version 1.3 - Migration Guide*. England, 1992.

[ISO 1988] ISO 9074. *Estelle: A Formal Description Technique Based on an Extended State Transition Model*. 1988

[ISO 1993] ISO. *Framework for enterprise modeling - ISO TC184 SC5 WG1*. 1993.

[ISO 1994] ISO. *Framework for enterprise modeling - ISO TC184 SC5 WG1. 1994*.

[ISO/IEC 1993] *ISO/IEC JTC1/SC21/WG7. Reference Model for Open Distributed Processing - Project 21.34 - Draft recommendation X.901 - Document N838*. 1993.

[Jain 1990] JAIN, H. K.; BU-HULAIGA, M. I. E-R approach to distributed heterogeneous database systems for integrated manufacturing. *Information Resources Management Journal*, 1990, vol. 3, no. 1, pp. 29-40.

[Jayaraman 1990] JAYARAMAN, S. Design and development of an architecture for computer-integrated manufacturing in apparel industry. Part I: Basic concepts and methodology selection. *Textile Research Journal*, 1990, vol.60, no.5, pp.247-254.

[Jochem 1989] JOCHEM, R.; RABE, M.; SUSSENGUTH, W. et al. An object oriented analysis and design methodology for computer integrated manufacturing systems. *Proceedings of Tools 89*. 1989, pp. 75-84.

[Jones 1989] JONES, A.; BARKMEYER, E; DAVIS, W. Issues in design and implementation of a system architecture for computer integrated manufacturing. *International Journal of Computer Integrated Manufacturing*, 1989, vol.2, no.2, pp.65-76.

[Jones 1990] JONES, A. ed. *Proceedings of CIMCOM' 90*. USA: NIST Special Publication 785, 1990.

[Johnson 1991] JOHNSON, B. C. A distributed computing enviroment framework: an OSF perspective. *Proceedings of The European Forum for Open Systems*. 1991, pp.69-87.

[Jorgenson 1992] JORGENSON, B. R. Model repository technology for model integration. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Johnson 1993] JOHNSON, G. B.; SCHOLES, H. K. *Exploring corporate strategy*. 3rd. ed. London: Prentice-Hall International Inc., 1993.

[Jorysz 1990a] JORYSZ, H. R.; VERNADAT, F. B. CIM-OSA Part 1: total enterprise modelling and function view. *International Journal of Computer Integrated Manufacturing*, 1990, vol. 3, no.3-4, pp.144-156.

[Jorysz 1990b] JORYSZ, H. R. &VERNADAT, F. B. CIM-OSA Part 2: information view. *International Journal of Computer Integrated Manufacturing*, 1990, vol.3, no.3-4, pp.157-167.

[Juanole 1989] JUANOLE, G.; ROUX, J. L. On the pertinence of the extended time Petri Net model for analysing communication activities. *Proceedings of the 3rd. International Workshop Petri Nets and Performance Models - PNPM 89*. IEEE Publisher, 1989, pp.230-235.

[Kay 1993] KAY, M. H. *Information Management - OPENframework Systems Architecture*. England: Prentice Hall,1993.

[Kobayashi 1990] KOBAYASHI, Y. A perspective of OSI standardization - object oriented architecture for distributed processing. *Proceedings of the INFOJAPAN*. Tokyo: IPSJ,1990, pp.513-520.

[Kosanke 1992] KOSANKE, K.; VERNADAT, F. CIM-OSA: a reference architecture for CIM. *Proceedings of the 8th International IFIP WG 5.3 Conference PROLAMAT '92*. Japan, 1992.

[Kosanke 1994a] KOSANKE, K. ed. *Workshop on Business Re-engineering at ADITEC*. Germany, 1994.

[Kosanke 1994b] KOSANKE, K. ed. *Proceedings of the Workshop on CIM-OSA*. CIM-OSA Club. Germany, 1994.

[Kourie 1989] KOURIE, D. G.; VAN DEN HEEVER, R. J. Distribuited systems in ISO-context. *Proceedins of The Int. Sympp. on Parallel Processing*. Amsterdam, 1989, p100-112.

[Kramer 1990a] KRAMER, J. Configuration programming - a framework for development of distributable systems.*Proceedings of the Conf. on Distributed Computing Systems*, 1990.

[Kramer 1990b] KRAMER, J.; MAGEE, J.; FINKELSTEIN, A. A construtive approach to the design of distributed systems. *Proceedings of the Conf. on Distributed Computing Systems*, 1990.

[Kramer 1992] KRAMER, J.; SLOMAN, M. *Proceedings of The International Workshop on Configurable Distributed Systems*. England: IEE, 1992.

[Kwikkers 1992] KWIKKERS, R. AND WORTMANN, J. C. The FOF modelling framework. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992. p 259-265.

[Leva 1987] LEVA, A. D.; VERNADAT, F.; BIZIER, D. Information system analysis and conceptual databese design in production enviroment with M*. *Computers in Industry*, 1987, vol.9, no.3, pp.183-217.

[LCMI 1989] LCMI (Laboratorio de Controle e Microinformatica). *ARP (Petri-Net Analyser)*. Brazil, 1989.

[Linington 1991] LININGTON, P. F. Open distributed processing and open management. *Proceedings of The Symposium on Integrated Network Management*. Amsterdam: Elsevier Science Publishers B. V., 1991, p553-56,.

[Liu 1990] LITT. The development of a CIM architecture for the RAMP program. *Proceedings of CIMCON'90*. USA: NIST special publication 785,1990.

[Longman 1990] LONGMAN, T. *Route Map - Line 3. Slides with a general description of ICL's Line 3*. 1990.

[Longworth 1992] LONGWORTH, G. *Introducing SSADM v. 4*. Oxford: NCC Blackwell, 1992.

[Magee 1989] MAGEE J.; KRAMER J.; SLOMAN M., Constructing Distributed Systems in Conic. *IEEE Transactions on Software Engineering*, 1989, vol 15, no 6, pp.663-675.

[Maji 1988] MAJI, R. K.; STEVENSON, I. A. *Design methodologies for CIM*. Kingston Pollytecnic, United Kingdom, 1988.

[Malhotra 1990] MALHOTRA, R.; JAYARAMAN, S. Design and development of an architecture for computer-integrated manufacturing in apparel industry. *Textile Research Journal*, 1990, vol.60, no.6, pp.351-360.

[Maloubier 1985] MALOUBIER, H.; et al. Use of GRAI method to analyse and design production management system. *Proceedings of the Conference on Advances in Production Management Systems-IFIP/84*. Netherlands: Elsevier Science Publisher B. V., 1984, pp. 127-140.

[Marechal 1987] MARECHAL, P. Method for systems modelization: The benefits of using IDEFO in the development of a software to program tape laying machines. *Proceedings of L'Automatique la producique 87*. Paris: Giipra, 1987, pp.27-32.

[Mayer 1991] MAYER, R. J.; PAINTER, M. K. Roadmap for enterprise integration. *Proceedings of Autofact 91*. USA, 1991, pp. 7.1-7.26.

[Mayer 1992] MAYER, R. J.; PAINTER, M. K.; DEWITTE, P. S. Appendix A - IDEF family of methods overview and pratical guidelines for IDEF use (internal report).

[McVitie 1993] McVITIE, D. *Platforms - OPENframework Systems Architecture*. England: Prentice Hall, 1993.

[Meer 1992] MEER, J.; HEYMER, V.; ROTH, R. ed. *Proceedings of the IFIP TC6/ WG6.4 International Workshop on Open Distributed Processing*. Amsterdam: North-Holland, 1992.

[Mertins 1992] MERTINS, K.; ALBRECHT, R.; STEINBERGER, V. et al. Flexible software system for production-adequate control. *Production Planning & Control*, 1992, vol.3, no.2, pp. 183-198.

[Mertins 1994] MERTINS, K; JOCHEM, R.; HOFMANN, M. J. *Methodology of business process modelling - QCIM arbeitskreis QUM. Contribution of QCIM project to ISO TC 184/SC4/WG8*. Germany, 1994.

[META 1990] META Software Corporation. *Design/IDEF User's Manual*. USA. 1990.

[Miller 1983] MILLER, D. Configurations of strategy and structure: towards a synthesis. *Strategic Management Journal*, 1983, vol. 7, pp. 233-249.

[Millis 1992] MILLIS, B. G. *Towards an overall information technology standartization framework or framework of framework. Internal document of the UK DISC Business Strategy Forum - Framework Working group*, draft 0.2, 1992.

[Mize 1992] MIZE, J. H.; BHUSKUTE, H. C.; PRATt, D. B. et al. Modeling of integrated manufacturing systems using an object-oriented approach. *IIE Transactions*, 1992, vol.24, no.3, pp.14-26.

[MSI 1994] MSI Research Institute. *Methods to structure business architecture and I.T. strategy development in manufacturing enterprises - application project A2. Case for support submitted to CDP*. 1994.

[Murgatroyd 1993] MURGATROYD, S.; EDWARDS, J.; WESTON, R.H. Tools to support the design of integrated manufacturing systems: an object oriented approach. *Proceedings of IEPM '93*. Mons, Belgium

[Naccari 1994]b NACCARI, F. et al. Business re-engineering at FIAT. *Workshop on Business Re-engineering at ADITEC*. Germany, 1994.

[Naeger 1993] NAEGER, G. *An integrated approach to software systems planning and selection base on CIMOSA-models*. Department of Computer Science, University of Karlsruhe, Internal Paper, 1993.

[Nof 1994] NOF, S. Y. Critiquing the potential of object orientation in manufacturing. *Int. J. Computer Integrated Manufacturing*, 1994, vol.7, no.1, pp.3-16.

[OMG 1991] OBJECT MANAGEMENT GROUP (OMG). *The Common Object Request Broker: Architecture and Specification. OMG Document number 91.12.1. - Revision 1.1*. USA 1991.

[Ostler 1991] OSTLER, G. ed. *The little Oxford dictionary of current English*. 6 ed., England: Clarendon Press,1991.

[Pan 1991] PAN, J. Y. C.; TENENBAUM, J. M. An intelligent agent framework for enterprise integration. *IEEE Transactions on Systems, Man, and Cybernetics*, 1991, vol.21, no.6, pp.1391-1407.

[ParcPlace 1990] ParcPlace Systems. *Objectworks/Smalltalk release 4 - user's guide*. USA, 1990.

[Parunak 1987] Parunak, H. V .D.; WHITE, J. F. A synthesis of factory reference models. *IEEE Workshop on Languages for Automation*. USA: IEEE Computer Society Press, 1987, pp.109-112.

[Pawling 1987] PAWLING, J. F. Surface Mounted Assemblies. Electrochemical Publications Ltd. 1987.

[Pereira 1982] PEREIRA, F. C *Prolog System.EdCAAD, Dept of Architecture*, University of Edinburgh, Scotland 1982.

[Peterson 1981] PETERSON, J. L. *Petri net theory and the modelling of systems*. USA: Prentice Hall, 1981.

[Petrie 1992a] PETRIE Jr., C. J. (editor). *Enterprise integration modeling. Proceedings of the First International Conference on Scientific and Engineering Computation Series*. USA: The MIT Press, 1992.

[Petrie 1992b] PETRIE Jr., C. J. (editor) The Working Group 3 of the ICEIMT Workshop III - Execution environment framework integration. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992, pp. 72-77.

[Petrie 1992c] PETRIE Jr., C. J. (editor) The Working Group 1of the ICEIMT Workshop I - the notion of a model. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992, pp. 72-77.

[Pidd 1992] PIDD, M. *Computer similation in management science*. 3rd. ed., Chichester: John Wiley & Sons, 1992. ISBN 0-471-93462-3.

[Pirron 1994] PIRRON, J. Product and process quality through CIM: 'QCIM' - a German reserach project. *Proceedings of the 10th International Conference on CAD/ CAM, Robotics and Factories of the Future*. Canada, 1994.

[Pleinevaux 1994] PLEINEVAUX, P. Integration of industrial applications: the CCE-CNMA approach. Submitted to the Proceedings of the *First IMSE*. France, 1994.

[Porter 1985] PORTER, M. E. *Competitive advantage - creating and sustaining superior performance*. USA: The Fee Press, 1985.

[Pratten 1993] PRATTEN, G. *Potential for Change - OPENframework Systems Architecture*. England: Prentice Hall, 1993.

[Querenet 1991] QUERENET, B. The CIM-OSA integrating infrastructure. *Computing & Control Engineering Journal*, 1991, vol. May, pp. 118-125.

[Ranky 1991a] RANKY, P. G. A general solution to the FMS design problem within CIM. *Factory Automation and Information Management*, 1991, pp.158-171.

[Ranky 1991b] RANKY, Pp. G. A systematic approach to the FMS design problem with CIM. *The Journal of Applied Manufacturing Systems*, 1991, vol.4, no.2, pp.39-45.

[Rembold 1991] REMBOLD, U.; NNAJI, B. The role of manufacturing models for information technology of the factory of the 1990s. *Journal of Design and Manufacturing*, 1991, vol.1, pp.67-87.

[Roberts 1984] ROBERTS, E. B. *Managerial application of system dynamics*. USA: The MIT Press, 1984. ISBN - 0-262-68035-1.

[Roberts 1991] ROBERTS, S. *SPEAR-Module Specification: Process Planning*. ICL. February 1991.

[Rogers 1989] ROGERS, P. *Object-oriented modelling of flexible manufacturing cells*. Ph.D. Thesis, University of Cambridge, 1989, pp. 7-18.

[Russel 1991] Russel, Peter. Modelling with CIMOSA. *Computing & Control Engineering Journal*, 1991, vol. May, pp.109-117.

[Sarkis 1994] SARKIS, J.; LIN, L. An IDEFo functional planning model for the strategic implementation of CIM systems. *Int. J. Computer Integrated Manufacturing*, 1994, vol.7, no.2, pp.100-115.

[Scheer 1991] SCHEER, A-W.; HARS, A. Enterprise modelling: basis for information systems designo. *Proceedings of The Conference of The GFQL*. Salzburg, 1991.

[Scheer 1992] SCHEER, A-W. *Architecture of integrated information systems: Foundations of enterprise-modelling*. Germany: Springer-Verlag, 1992.

[Schiel 1990] SCHIEL, U.; MISTRIK, I. Using object-oriented analysis and design for integrated systems. *Proceedings of the First International Conference on Systems Integration*. USA: IEEE, 1990, pp. 125-134.

[Shaharoun 1994] SHAHAROUN, A. M. A new approach to the life-cycle support of costing systems for advanced manufacturing environments. PhD Thesis. Loughborough University, England, 1994.

[Siemens 1987] SIEMENS. An Introduction to Surface Mounting. Product Catalogue. 1987.

[Siemens 1994] SIEMENS. *CIMOSA/ReMo - Tool for enterprise modeling on requirements level. Siemens Catalog*, 1994.

[Sijelmassi 1991a] SIJELMASSI, R., STRAUSSER, B. *The Portable Estelle Translater: An overview and user guide. NIST Technical Report*, 1991.

[Sijelmassi 1991b] SIJELMASSI, R., STRAUSSER, B. *The Distributed Implementation Generator: An overview and user guide. NIST Technical Report*, 1991.

[Singh 1994] SINGH, V. *Software Interoperability within Manufacturing Control Systems. PhD Thesis*. Loughborough University, England, 1994.

[Smethurst 1993] SMETHURST, R.; Wharton, P. *Availability - OPENframework Systems*

Architecture. England: Prentice Hall, 1993.

[SofTech 1976] SOFTECH Inc. SADT: structured analysis & design technique - Reader course. 1976.

[Stacey 1993] STACEY, R. D. *Strategic management and organisational dynamics*. England: Pitman Publishing,1993.

[Sun 1990a] SUN MICROSYSTEMS. *SunOS Reference Manual version 2. Part 2 - System Calls and Part 3 Library Functions*. USA, 1990.

[Sun 1990b] SUN MICROSYSTEMS. *Programmer's Overview Utilities & Libraries. Part 2 - Programming Utilities & Libraries*. USA, 1990.

[Sutcliffe 1993] SUTCLIFFE, S. ed. *Performance - OPENframework Systems Architecture*. England: Prentice Hall, 1993.

[Tenembaum 1992] TENEMBAUM, J. M.; WEBER, *J. C. Enterprise integration: lessons from SHADE and PACT. Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Thacker 1989] THACKER, R. M. *A new CIM model - A blueprint for the computer-integrated manufacturing enterprise*. USA: Society of Manufacturing Engineers, 1989. ISBN 0-87263-337-3.

[Tonshoff 1989] TONSHOFF, H. K.; HORNS, A.; SCHAELE, M. Integrated model hierarchy for factory automation. *Proceedings of the Conference on Software for Factory Automation*. Netherlands: Elsevier Science Publishers B. V., 1989, pp.207-232.

[Turney 1991] TURNEY, P. B. B. *Common cents - The ABC performance breakthrough*. USA: Cost Technology, 1991. ISBN 0-9629576-0-7

[Tzafestas 1989] TZAFESTAS, S. Petri-net and knowledge-based metodologies in manufacturing systems modelling simulation and control. *Proceedings of 5th CIM European Conference*. Belgium: ECSC-EAEC Publishers, 1989, pp. 39-50.

[Verheijen 1982] VERHEIJEN, G. M. A.; VAN BEKKUM, J. NIAM: an information analysis method. *Proceedings of The Confernece on Information Systems Design Methodologies: A comparative Review, IFIP 82*. Netherlands: North-Holland, 1982, pp. 539-587.

[Vernadat 1992] VERNADAT, F. CIM-OSA - a European development for enterprise integration part 2: enterprise modelling. *Proceedings of the International Conference on Enterprise Integration Modelling (ICEIMT 92)*. USA, 1992.

[Vervoort 1988] VERVOORT, W. A. Evaluation of the YOURDON methodology used to design a CIM system. *Proceedings of ISCIS III. Third International Symposium on Computer and Information Sciences*. Turkey, 1988, pp.697-703.

[Waldrop 1994] WALDROP, M. M. *Complexity - The emerging science at the edge of order and chaos*. UK: Penguin Books, 1994. ISBN 0-1401-7968-2.

[Waskiewicz 1994] Waskiewicz, F. (editor) OMG manufacturing SIG minutes (held in October 1994 in Nshua/NH), USA, 1994.

[Weaver 1994] WEAVER, A. et al. A soft systems approach to manufacturing re-design. *Proceedings of the Tenth National Conference on Manufacturing Research*. England:Taylor & Francis, 1994, p360-364.

[Weston 1993]# WESTON, R. H. Steps towards enterprise-wide integration: a definition of need and first generation open solutions. *International Journal of Production Research*, 1993

[Weymont 1987] WEYMONT, N. P.; HONEYAGER, J. S. Developing a CIM Architecture. *Proceedings of the Digital Equipment Computer Users Society*. USA: Anaheim, 1987, pp.45-65.

[Wheatley 1992] WHEATLEY, M. J. *Leadership and the new science: learning about*

organization from ordely universe. San Francisco: Berrett-Kohker Publishers, 1992. ISBN 1-881052-01-X.

[Wiendahl 1991] WIENDAHL, H. -P.; GARLICHS, R. Trends in CIM. *Future Generation Computer Systems*, 1991, vol. 7, pp. 97-107.

[Wiener 1948] WIENER, N. *Cybernetics*. Paris: Herman & Cie, 1948.

[Williams 1989] WILLIAMS, T. J. ISA. *A reference model for CIM implementation - a description from the viewpoint of industrial automation*. USA: Instrument Society of America, 1989.

[Williams 1993] WILLIAMS, T. ed. *Architectures for integrating manufacturing activities and enterprises. IFAC/IFIP Task Force on Architectures for Integrating Manufacturing Activities and Enterprises*. USA: IFAC/IFIP,1993.

[Williams 1994] WILLIAMS, T. J. *Contributions of the Purdue Enterprise Reference architecture and Methodology (PERA) to the development of a General Enterprise Reference Architecture and Methodology (GERAM). (submitted to the) IFIP/IFAC Task Force on Architectures for Enterprise Integration*. USA, 1994.

[Wu 1992] WU, B. *Manufacturing systems design and analysis*. Germany: Chapman & Hall, 1992. ISBN 0-412-40840-6.

[Yeomans 1984] YEOMANS, R. W. Design rules for computer integrated manufacturing systems. *Proceedings of the ESPRIT'84: Status Report of Ongoing Work*. Netherlands: Elsevier Science Publishers B. V.,1985, pp. 457-493.

[Yoder 1990] YODER. Toward a new CIM architecture for Sandia laboratories. *Proceedings of CIMCON'90*. USA: NIST special publication 785,1990.

[Yourdon 1989] YORDON, E. *Modern structured analysis*.Yourdon Press computing series. Englewood Cliffs. USA, 1989.

[Zachman 1986] ZACHMAN, J. A. *A frameworkk for information systems architecture. IBM Los Angeles Scientific Center, G320-2785*, 1986.

[Zachman 1987] ZACHMAN, J. A. A framework for information systems architecture. *IBM Systems Journal*, 1987, vol.26, no.3, pp.276-292.

List of Appendices

Appendix 1: Definition of the constructs used in the life-cycle model

- **Business analysis.** Strategy definition is achieved using methods and techniques to guide and support an analysis of strategic factors. Some of these methods include those discussed in Section 2.2.7.

- **Enterprise system.** This defines the overall scope of the system under consideration in respect of the activities which need to be considered when carrying out business analysis. Such a system can be described as embracing all functions or business processes of the enterprise (i.e. its **strategic domains**). Business analysis will normally be achieved by viewing the enterprise as a system whose composition can be described as macro-resources (i.e. **enterprise resources**, e.g. manufacturing resources, human resources, financial resources, etc.) which the enterprise requires to operate within the system environment (i.e. within the **enterprise environment**, comprising for example of customers, suppliers and competitors, each of which will be subject to social, political and economic factors).

- **Business models.** These collectively represent the business knowledge required to accomplish business analysis; they include business rules (if they exist) and the accumulated experience of those performing the analysis.

- **Required improvements.** These are achieved as a result of the business analysis, and identify key issues for improvement, so that the enterprise system can more successfully cope with the demands of its environment. The issues raised can be associated with one or more enterprise domains.

- **System analysis.** This can be structured and supported using the languages and frameworks discussed in Chapter 2 (e.g. CIM-OSA, IDEF, GRAI, Yourdon, etc.).

- **Enterprise domain.** Here, the scope can be that of a sub-system within the enterprise system (see Figure 1), where modelling is achieved to a level of granularity that identifies key functional components. The boundaries of an enterprise domain are defined according to specifics of the enterprise under consideration, in as much that they will be chosen to reflect issues requiring attention to realise identified improvements.Typically, enterprise domains are either a **business process** (e.g. production introduction process, order-flow process, etc.) or an **enterprise area** (e.g. shop-floor, engineering, etc.).

- **System models.** These represent knowledge of the way domains are currently organised or should be organised to achieve improved performance. Additionally, these models encapsulate the experience of system analysts, and can include archetype descriptions of how systems pertaining to a given type of industry are

typically organised (e.g. system models can be viewed as partial models in CIM-OSA terms [ESPRIT/AMICE 1993a]).

- **Suitable scenarios.** These represent the result of the conceptual analysis meta-phase, and identify alternative ways of organising the domain (i.e. 'should-be' scenarios), as well as more constrained solutions which might prove to be more suitable to address the domain under consideration (i.e. 'to-be' scenarios).

- **System build.** This comprises design and implementation processes which may be structured and supported via the application of formalised methods and techniques. Such methods should embody empirical knowledge required to transform a requirements specification into a physical implementation, along with system design methods to guide the process (e.g. the use of CIM-OSA and object-oriented design methods as a means of manipulating the designer's knowledge of available technological resource components)

- **System solution.** This corresponds to the scope of a working system which can realise the 'to-be' scenario identified for a certain domain during conceptual analysis. A system here is viewed as typically comprising: **physical components** that could either be active (i.e. machines, human beings, application programs) or passive (i.e. jigs and fixtures, tools, data elements, etc.); **infrastructural components** (i.e. those IT service elements of the system that are generic to all enterprises, e.g. an integrating infrastructure), **integrating elements** (i.e. those elements of the system that 'glue' together physical components[1] and infrastructural elements, which can usually take the form of a model.

- **Resource models.** These models capture information describing resources and their possible inter-relationships which will be required to realise a physical system; this should include information typically considered by a system designer when deciding to select a certain resource to perform a defined function in a system.

- **Integration models.** These models encapsulate information about how the various component elements of a system should be put together (i.e. how integration among system components can be achieved using infrastructural elements).

- **Scenario realisation.** The result of the processes carried out in the design and implementation meta-phase will be a system ready to be deployed. Here, it is important to note that before any system solution is deployed, it should be tested within the context of the enterprise system as a whole (i.e. this is an essential part of

---

1. The terms "system components", "resource components", "physical components", "physical resources" and "active resource components" are used interchangeably in this thesis.

the support required from a CASSE environment).

- **Enterprise management.** This includes management of the operational activities of an enterprise.
- **Operational system.** The scope of this meta-phase is the whole enterprise system at work, comprising **operational components** which use **infrastructural services** in order to perform **actions** (i.e. atomic transactions, e.g. data access, material movement and transformation, etc.) which, as a coordinated whole, lead to the achievement of the system's objectives.
- **Run-time models.** These models encapsulate information concerning the way in which the system works; this includes operational procedures defining task execution and data which supports those tasks and the actual code generated for the components and elements manipulated in the previous phase.
- **Flexible operation.** The deliverable of this meta-phase is an working system which has inherent flexibility to enable maintenance and change during its working life.

Appendix 2: Overview of Model-Driven CIM

"Model-Driven CIM" (a framework and toolset for the design, implementation and management of open CIM systems) is a research grant funded by ACME-SERC Directorate, which aims at providing a collection of tools to support the life cycle of integrated manufacturing systems, by addressing their functional, information and resource aspects.

A main research thrust of the MSI Research Institute, in which this grant is place, has been to seek ways of "bridging the gaps" in formalism and support for the Integrated Manufacturing System life cycle. Here, particular emphasis has been on the gap between the "modelling" and "physical" worlds. Means of filling the "gap" have been realised through study of ways of achieving "model-enactment" (i.e. model execution). Means of filling such a gap will inevitably be related to the models, methods and tools used in the modelling and physical worlds. At MSI the research methodology adopted has been as follows:

(I) Assess the public domain literature on established and evolving models, methods and tools commonly used in each world.

(II) Select potentially cognate groupings of models and methods and to seek means of extending and unifying them to support the various life-cycle phases of an IMS.

(III) Create software tools which support the life-cycle by unifying and underpinning the operation of chosen sets of models and methods.

(IV) Use and evaluation of the software tools created in (III).

Figure A-1 categorises the main research thrusts to-date, where project activity has been organised within 6 workpackages.

Workpackages 1 and 2 have their centre of gravity in the modelling world and seek to extend state of the art generally applicable IMS design and modelling methods, by building respectively on (i) the use of a collection of architectures centred on the CIM-OSA reference architecture and (ii) Object-oriented design methods (originally conceived to enable and structure the creation of software). Much of the "extension" of these methodologies is focused on establishing means of enacting the various models created using the two approaches, thereby beginning to fill the "gap" in a top to bottom direction. Both approaches have led to the creation of new prototype modelling tools where implementation of the underlying methodologies has been facilitated via a Meta CASE tool.

Workpackage 1, which coincides with the scope of the author's research, "Reference Architecture Based Modelling Tool", is implementing and extending a combination of state-of-the-art Reference Architectures centred on CIM-OSA. This workpackage is providing means of rapidly prototyping a system through the integrated

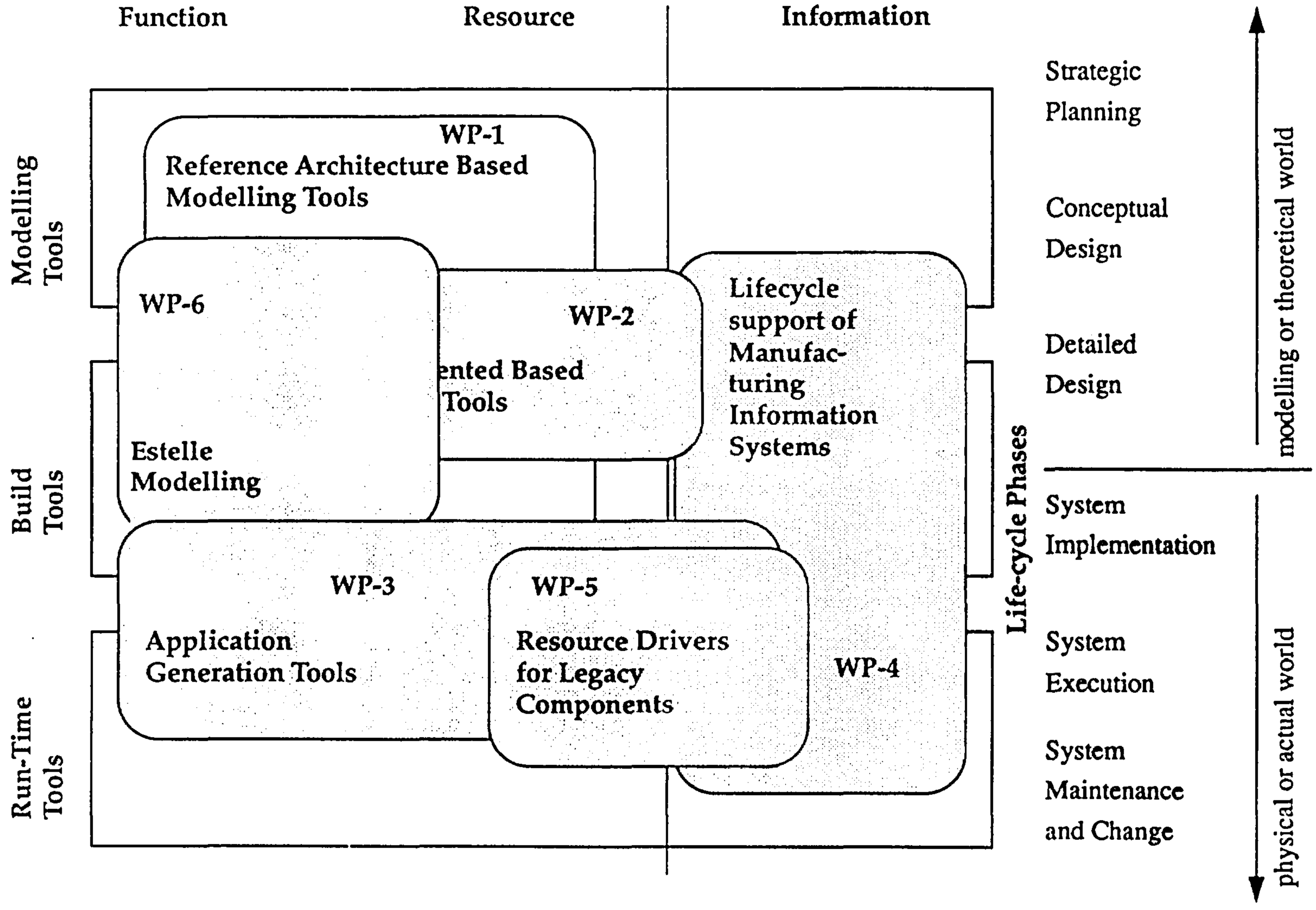


Figure A-1 - Model Driven CIM - Research Framework

use of a modelling, analysis, simulation, rapid-prototyping, configuration and operation of an IMS driven by models. Particular emphasis to date has been placed on enabling analysis of functional properties of both static and dynamic nature of integrated manufacturing systems. The work embraces the development of SEW-OSA - System Engineering Workbench for CIM-OSA which will be used to realise the structure of an IMS so that it can run across an integrating infrastructure.

Workpackage 2, which is conducted by I. S. Murgatroyd [Murgatroyd 1993] is focused on the extension of state-of-the-art Object Oriented design methods. The work consists of developing a CASE tool based on the Booch method which can be used to model and build the components of an IMS.

The tools of workpackages 1 and 2 have been created in an interactive manner, with the use of successively enhanced versions being evaluated through the case study modelling of "as-is" integrated manufacturing systems.

Workpackage 3, which is conducted by I. Coutts [Coutts 1994], has focused on defining and implementing tools to support a standard interface between software applications and an integrating infrastructure such as Misses CIM-BIOSYS, IBM's DAE and OSF/DCE].

The research of workpackage 4, which is conducted by P. Clements [Clements 1993], focuses on information issues by creating a unified software toolset which supports various forms of abstract modelling of manufacturing information, through automatically manipulating information and populating different forms of data storage device, to the provision of consistent and configurable information support services during system run time.

Workpackage 5, which is conducted by M. Leech, embraces methods and tools which have been derived to deal with legacy systems, where the tools produced enable the integration of current and previous generations of "closed" manufacturing system components. Such components have invariably been designed in a proprietary stand-alone manner, without reference to a "big picture" of what is required to achieve inter-working with other system components.

Finally, workpackage 6, which is conducted by P. Gilders [Gilders 1995], focuses on the development of an environment for modelling the interactions amongst system components from a 'bottom-up' perspective, based on the Estelle language.

Appendix 3: Details on Tools and Technologies

A-3.1.CASE Tools

CASE (computer-aided software system engineering) is an approach to the design of software systems in which the design method is formalised in a software tool. There exist a number of categories of CASE tools. According to Brathwaite [Brathwaite 1990], "an individual CASE tool automates one small focused step in the life cycle process [...] and individual tools fall into the following general categories:

- "Diagramming tools for pictorially representing system specifications;
- "Screen and report painters for creating system specifications and for simple prototyping;
- "Dictionaries, information management systems, and facilities to store, report, and query technical and project-management system information;
- "Specification-checking tools to detect incomplete, syntactically incorrect, and inconsistent system specifications;
- "Code generators to be able to generate executable code from pictorial system specifications;
- "Documentation generators to produce technical and user documentation required by structured methodologies.

A-3.1.1.ToolBuilder Meta-CASE tool

The modules of the ToolBuilder Meta-CASE tool depicted in Figure 18 are defined as follows:

a. Data Model:

This is a detailed description of attributes, rules and updates associated with each construct or diagram. Here, in addition to refining the definition of entities and relationships identified in the CASE Tool Structure, segments of code associated with operations to be executed when an entity, attribute or relationship is created, updated or deleted are defined.

b. Frames Model:

This is the definition of the content of each diagram or hyper-text template created to enable the CASE tool user to input his (or her) model. Navigations between diagrams and templates are also defined. The constructs used in each template or diagram are defined and related to their representation which can be on a text or graphical format.

c. Graphic Catalogue:

This catalogue defines the shape and style of each object[1] manipulated in the diagrams. That is, the catalogue is a drawing tool to create icons which represent constructs manipulated in the diagrams.

d. Catalogue of Subsections and Objects:

This is a library of functions and text objects used to define the content of each text template and its associated hypertext operations.

e. User/External Library:

The definitions of each building block in Figure 18 is ultimately converted into fragments of code. The language adopted by ToolBuilder for the description of that code is called Easel (i.e. a high-level interpreted language resembling Pascal). The library of functions are also written in Easel by the user and linked to the main code generated by the tool. This library is used to define the code of operations on constructs and diagrams which are triggered when they are involved in specified constructs. Examples of such operations are to: tide a diagram up, update attributes, create additional entities, etc.

f. Document Configuration:

The Document Configuration is a report generator. It enables to create the operations required to generate code based on information extracted from diagrams and constructs defined in the Data Model. These reports are created with a separate tool, the **Publisher** (see Figure 18) which may also make use of the User/External Library in order to manipulated the Data Model for the sake of code generation.

A-3.1.2.BuilderXcessory

CASE technology was also used to aid the development of a business entity for SEW-OSA (see Figure 9), namely BuilderXcessory (BX) [ICS 1991a] [ICS 1991b] a user interface generator (i.e. a 'screen painter' in Brathwaite's terminology [Brathwaite 1990]). BX enables rapid definition of user interfaces for X-Windows/Motif environment, by providing a graphical tool for the construction of user interfaces. BX comprises of:

- a catalogue of widgets (i.e. text windows, scrolled lists, buttons, etc.) which can be used and combined to construct a particular user interface;
- a resource interface for customising each widget and defining its associated call

---

1. In ToolBuilder terminology, an object is a graphical or text representation of a modelling construct.

back functions;

- a main interface from which the structure of the user interface can be visualised and changed.

BX enables rapid prototyping of an application program, with generation of "C" code organised in three files:

- creation-c.c. This is where the interface definition is held;
- callbacks-c.c. This is where call-back functions, associated with actions performed by the user through the interface, are placed. BX generates the function definition and X-Windows commands required for gracefully executing these functions. Software code for the tasks associated with those actions must be defined by the user in this file;
- main-c.c. This is the main program which contains user-interface initialisations.

A-3.1.3.The CIM-BIOSYS infrastructure

The main elements of the CIM-BIOSYS integrating infrastructure depicted in Figure 19 are defined as follows:

- the **service manager** provides a consistent interaction mechanism which enables transparent application interaction (i.e. application services) and information access (i.e. information services). Typical application services include (1) "to establish a link with another application" and (2) "to pass a message to an application". A typical information service is "to open a file in a logical file store";
- the **driver manager** provides facilities similar to the service manager, in the form of a consistent interaction mechanism for device drivers to provide communication between CIM-BIOSYS and remote non-compliant devices (e.g a robot, a placement machine, or any active device which is unable to comply with the access mechanism provided by CIM-BIOSYS);
- the **run-time manager** supports the integration services, by controlling the processes[1] external to and registered with CIM-BIOSYS which use its services. It also monitors error conditions within the infrastructure and provides human interface facilities through an Engineers Interface. This interface enables full manual control of the Run-time Manager, if required;
- the **configuration manager** maintains all internal system configuration data and external configuration files via an Administration Interface.

---

1. A CIM-BIOSYS process is a run-time occurrence of a CIM-BIOSYS application or device driver.

Appendix 4: General Introduction to Petri-Nets

A Petri-net is a bipartite graph [David 1994]. That is, it comprises two types of nodes, namely: places and transitions. Arcs link places and transitions alternately on a path made up of consecutive arcs. A Petri-net is marked, in the sense that an integer number (positive or zero) of tokens is assigned to every place. The union of marks of all places in a net defines the net marking. That is, the system state is defined by a given configuration of tokens distributed among places (marking). The flow of tokens (i. e. transitions from input places to output places) defines the state transition (or event) and, consequently, the system dynamics. Arcs can be weighted (this configuring a "generalised Petri-net"), which define the number of tokens that they carry when their associated transition is fired. This process can be illustrated with the example shown in Figure A-1.

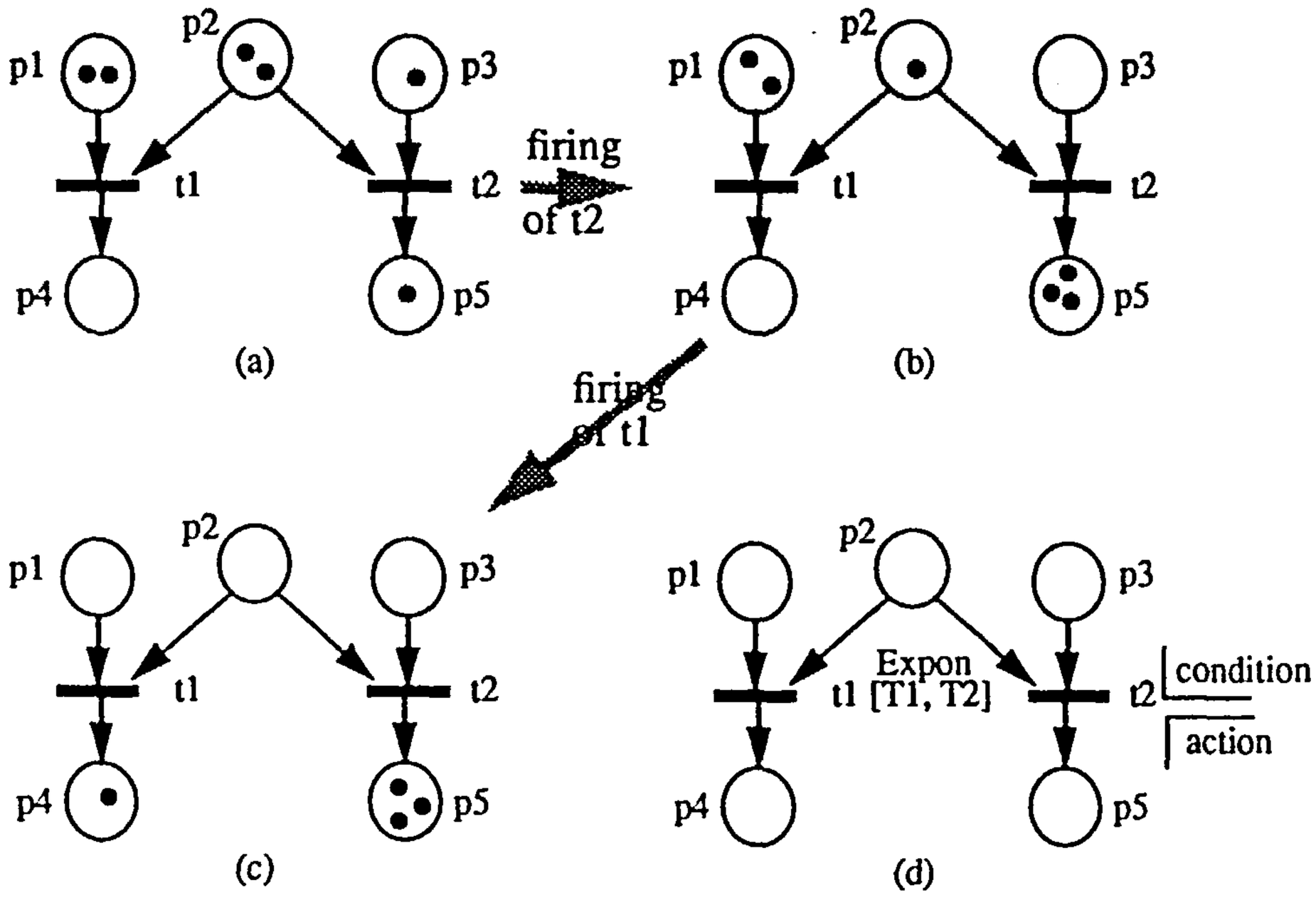


Figure A-1 - Example of Petri-net evolution

Here, the initial state of the net (see Figure A-1.a) is one in which the transitions **t1** and **t2** are enabled (i.e. they are able to "fire", this to provoke a change of state in the net).This is the case because, for example, in order for **t1** to fire, it requires two tokens in **p1** and one token in **p2** (this is determined by the weight of its input arcs - see Figure A-1). The firing of **t2** provokes the extraction of one token from **p2**, another from **p3**, and the inclusion of two tokens in **p5** (see Figure A-1.b). Notice that when no weight is associated to an arc, a weight of one is assumed. In the state described in Figure A-1.b, **t1** is still enabled whereas **t1** is not (i.e. there is no token left in **p3**). If there were, **t1** and **t2** would be both enabled and competing for the only remaining

token in **p2** (i.e. the situation of a conflict). When **t1** fires, two tokens are extracted from **p1**, one token from **p2**, and one token is added to **p4**.

A generalised stochastic time Petri-net (GSTPN) [Juanole 1989] is an extension of the ordinary Petri-net [Peterson 1981] which associates a time interval to the firing of each transition and a probability distribution function relating to each time interval (see transition **t1** in Figure A-1.d). In addition to inheriting the properties of an ordinary Petri Net, a GSTPN allows timing considerations to be attached to system models to facilitate description of the evolution of states. Here, the time interval associated with each transition defines the lower and the upper time limit at which that particular event in the system is to occur. The stochastic nature of GSTPN allows the definition of a probability distribution to each time interval, which makes GSTPN particularly appropriate for simulation, when seeking to determine average values of system parameters (performance measures).

A predicate-action Petri-net (also termed as "interpreted Petri-net" [David 1994]) is an extension of an ordinary Petri-net which associates the firing of a transition with the occurrence of events external to the system which the Petri-net is modelling. Such events can be a condition to enable the firing of a transition or an action that is taken when an transition is fired (see transition **t2** in Figure A-1.d). Predicate-action Petri-nets are quite useful to model relationships and dependencies between the internal behaviour of a system and its external environment.

Appendix 5: Example of a Petri-Net Model in the ARP Syntax

```
NET smt;

NODES
  EV-1 : TRANSITION[0, 0] NORMAL(10);
  pEV-1 : PLACE (1);
  pib3 : PLACE (0);
  EV-2 : TRANSITION[0, 0] NORMAL(10);
  pEV-2 : PLACE (0);
  poEV-2 : PLACE (0);

          •
          •
          •
;
  ARC-6pFE-6_1 : PLACE (1);
  ARC-6pFE-6_2 : PLACE (0);
  ARC-6tFE-6_1 : TRANSITION [0, 0] NORMAL(10);
  ARC-6tFE-6_2 : TRANSITION [50, 60] NORMAL(10);
  placem : PLACE (3);
  convey : PLACE (3);
  finish : PLACE (6);

STRUCTURE
  EV-3 : (1*pEV-3),(1*pib5);
  b3 : (1*pib3),(1*piEA-1,1*piDP-1EA-1);
  b4 : (1*poDP-1EA-1),(1*piFn1);
  Fn1 : (1*piFn1),(1*pFn1);

          •
          •
          •

  ARC-5tFE-5_1 : (1*ARC-5pFE-5_1,1*pFO-15),(1*ARC-5pFE-5_2);
  ARC-5tFE-5_2 : (1*ARC-5pFE-5_2),(1*ARC-5pFE-5_1,1*pFO-16);
  ARC-5tFE-5_3 : (1*ARC-5pFE-5_1,1*pFO-19),(1*ARC-5pFE-5_3);
  ARC-5tFE-5_4 : (1*ARC-5pFE-5_3),(1*ARC-5pFE-5_1,1*pFO-20);
  ARC-6tFE-6_1 : (1*ARC-6pFE-6_1,1*pFO-17),(1*ARC-6pFE-6_2);
  ARC-6tFE-6_2 : (1*ARC-6pFE-6_2),(1*ARC-6pFE-6_1,1*pFO-18);
  EV-1 : (1*pEV-1),(1*pib3);
  EV-2 : (50*pEV-2),(1*poEV-2);


ENDNET.
```

Appendix 6: Example of ARP Reports

A-6.1.Report on a Performance Evaluation Run

Performance Evaluation Oriented to EVENTS of Net smt.

Inicial Marking : {pEV-1, FE-1, FE-2, FE-3, FE-4, FE-5, FE-6, ARC-1pFE-1_1,
ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey,
6* finish}
Desired Precision : 0.10 %
Max. of Fires : 10000
Num. of Reaching : 518
Improdutive Interac.: 0 (0.00 %)

Fire probability attributed to conflict groups:

Group1: (b12: 20%) (b13: 80%)
Group2: (b16: 90%) (b17: 10%)

Average number of fires from cycle transitions :
(EV-1: 1.00) (EV-2: 1.00) (EV-3: 50.00) (b3: 1.00) (b4: 1.00)
(Fn1: 1.00) (iDP-1EA-1: 1.00) (tin_EA-1: 1.00) (tout_EA-1: 1.00)
(tEA-1_81: 1.00) (tEA-1_82: 1.00) (tEA-1_83: 50.00) (b5: 50.00)
(b6: 50.00) (b7: 50.00) (b8: 49.01) (Fn2: 49.01) (iDP-2BP-1: 50.00)
(iDP-2BP-2: 50.00) (iDP-2BP-3: 49.03) (b9: 50.00) (b10: 50.00) (b11: 10.06)
(b12: 10.06) (b13: 39.94) (St1: 50.00) (Fn3: 50.00) (iBP-1EA-2: 50.00)
(iBP-1EA-3: 50.00) (iBP-1EA-4: 10.06) (b14: 50.00) (b15: 50.00) (b16: 44.96)
(b17: 5.04) (b18: 5.04) (b19: 50.00) (St2: 50.00) (Fn4: 50.00)
(iBP-2EA-5: 50.00) (iBP-2EA-6: 50.00) (iBP-2EA-7: 5.04) (iBP-2EA-8: 50.00)
(b20: 50.00) (b21: 50.00) (b22: 49.11) (St3: 50.00) (Fn5: 49.06)
(iBP-3EA-9: 50.00) (iBP-3EA-10: 49.23) (tin_EA-2: 50.00) (tout_EA-2: 50.00)
(tEA-2_86: 50.00) (tEA-2_87: 50.00) (tin_EA-3: 50.00) (tout_EA-3: 50.00)
(tEA-3_100: 50.00) (tEA-3_101: 50.00) (tEA-3_102: 50.00) (tEA-3_103: 0.00)
(tin_EA-4: 10.06) (tout_EA-4: 10.06) (tEA-4_106: 10.06) (tEA-4_107: 10.06)
(tin_EA-5: 50.00) (tout_EA-5: 50.00) (tEA-5_110: 50.00) (tEA-5_111: 50.00)
(tin_EA-6: 50.00) (tout_EA-6: 50.00) (tEA-6_124: 50.00) (tEA-6_125: 50.00)
(tEA-6_126: 50.00) (tEA-6_127: 0.00) (tin_EA-7: 5.04) (tout_EA-7: 5.04)
(tEA-7_130: 5.04) (tEA-7_131: 5.04) (tin_EA-8: 50.00) (tout_EA-8: 50.00)
(tEA-8_134: 50.00) (tEA-8_135: 50.00) (tin_EA-9: 50.00) (tout_EA-9: 50.00)
(tEA-9_138: 50.00) (tEA-9_139: 50.00) (tin_EA-10: 50.00) (tout_EA-10: 49.49)
(tEA-10_146: 50.00) (tEA-10_147: 50.00) (tEA-10_148: 50.00)
(ARC-1tFE-1_1: 1.00) (ARC-1tFE-1_2: 1.00) (ARC-1tFE-1_3: 10.06)
(ARC-1tFE-1_4: 10.06) (ARC-2tFE-2_161: 50.00) (ARC-2tFE-2_162: 50.00)
(ARC-2tFE-2_163: 50.00) (ARC-2tFE-2_164: 50.00) (ARC-3tFE-3_83: 50.00)
(ARC-3tFE-3_84: 50.00) (ARC-4tFE-4_1: 50.00) (ARC-4tFE-4_2: 50.00)
(ARC-4tFE-4_3: 50.00) (ARC-4tFE-4_4: 50.00) (ARC-5tFE-5_1: 5.04)

(ARC-5tFE-5_2: 5.04) (ARC-5tFE-5_3: 50.00) (ARC-5tFE-5_4: 50.00)
(ARC-6tFE-6_1: 50.00) (ARC-6tFE-6_2: 50.00)

Average time of fire :
(ARC-1tFE-1_2: 39.75) (ARC-1tFE-1_4: 15.00) (ARC-2tFE-2_162: 1.32)
(ARC-2tFE-2_164: 20.00) (ARC-3tFE-3_84: 67.99) (ARC-4tFE-4_2: 3.32)
(ARC-4tFE-4_4: 3.32) (ARC-5tFE-5_2:271.56) (ARC-6tFE-6_2: 55.01)

Average marking in places :
(pEV-2: 23.14) (pFn1: 0.99) (piDP-1EA-1: 0.01) (pEA-1_132: 0.01)
(pFn2: 23.14) (piDP-2BP-1: 7.80) (poDP-2BP-1: 14.30) (piDP-2BP-2: 3.12)
(piDP-2BP-3: 1.12) (piBP-1EA-2: 5.64) (piBP-1EA-3: 2.13)
(piBP-1EA-4: 0.02) (piBP-2EA-5: 2.81) (piBP-2EA-6: 0.05)
(piBP-2EA-7: 0.22) (piBP-2EA-8: 0.05) (piBP-3EA-9: 0.86)
(piBP-3EA-10: 0.25) (piEA-2: 5.63) (pEA-2_142: 0.02) (piEA-3: 1.85)
(pEA-3_157: 0.28) (pEA-4_167: 0.02) (piEA-5: 1.85) (pEA-5_176: 0.96)
(pEA-6_191: 0.05) (pEA-7_201: 0.22) (pEA-8_210: 0.05) (piEA-9: 0.09)
(pEA-9_219: 0.78) (piEA-10: 0.25) (FE-1: 0.95) (FE-2: 0.56)
(ARC-1pFE-1_1: 0.95) (ARC-1pFE-1_2: 0.01) (ARC-1pFE-1_3: 0.03)
(ARC-2pFE-2_264: 0.56) (ARC-2pFE-2_265: 0.03) (ARC-2pFE-2_266: 0.41)
(ARC-5pFE-5_2: 7.05)

Inicial Event: EV-1

Ed 1: EV-2
Average Time: 3698.89 Deviation: 196.42 Probab: 100.00 % Reach: 518
Minimum time of reaching : 3485.89 Maximum time of reaching : 4715.45

A-6.2.Report on a Verification of the Petri-net Properties

State Enumeration : net smt (103 reachable states).

Verified properties :

Net under analisys is limited.
Null places (M = 0): {piFn1, pFn1, poDP-1EA-1, poEA-1, poBP-1EA-4, poEA-4,
pEA-4_166, pEA-4_167, pEA-4_168, pFO-7, pFO-8,
ARC-1pFE-1_3}
Binary places : {pEV-1, pib3, pEV-2, poEV-2, pEV-3, pib5, piDP-1EA-1,
piEA-1, pEA-1_131, pEA-1_132, pEA-1_133, pEA-1_134,
piFn2, pFn2, piDP-2BP-1, poDP-2BP-1, piDP-2BP-2,
poDP-2BP-2, piDP-2BP-3, poDP-2BP-3, piBP-1, poBP-1,
poSt1, piFn3, piBP-1EA-2, poBP-1EA-2, piBP-1EA-3,
poBP-1EA-3, piBP-1EA-4, piBP-2, poBP-2, poSt2, piFn4,
piBP-2EA-5, poBP-2EA-5, piBP-2EA-6, poBP-2EA-6,

piBP-2EA-7, poBP-2EA-7, piBP-2EA-8, poBP-2EA-8,
piBP-3, poBP-3, poSt3, piFn5, piBP-3EA-9, poBP-3EA-9,
piBP-3EA-10, poBP-3EA-10, piEA-2, poEA-2, pEA-2_141,
pEA-2_142, pEA-2_143, piEA-3, poEA-3, pEA-3_156,
pEA-3_157, pEA-3_158, pEA-3_159, piEA-4, piEA-5,
poEA-5, pEA-5_175, pEA-5_176, pEA-5_177, piEA-6,
poEA-6, pEA-6_190, pEA-6_191, pEA-6_192, pEA-6_193,
piEA-7, poEA-7, pEA-7_200, pEA-7_201, pEA-7_202,
piEA-8, poEA-8, pEA-8_209, pEA-8_210, pEA-8_211,
piEA-9, poEA-9, pEA-9_218, pEA-9_219, pEA-9_220,
piEA-10, poEA-10, pEA-10_233, pEA-10_234, pEA-10_235,
pEA-10_236, pFO-1, pFO-2, pFO-3, pFO-4, pFO-5, pFO-6,
pFO-9, pFO-10, pFO-11, pFO-12, pFO-13, pFO-14,
pFO-15, pFO-16, pFO-17, pFO-18, pFO-19, pFO-20, FE-1,
FE-2, FE-3, FE-4, FE-5, FE-6, ARC-1pFE-1_1,
ARC-1pFE-1_2, ARC-2pFE-2_264, ARC-2pFE-2_265,
ARC-2pFE-2_266, ARC-3pFE-3_137, ARC-3pFE-3_138,
ARC-4pFE-4_1, ARC-4pFE-4_2, ARC-4pFE-4_3,
ARC-5pFE-5_1, ARC-5pFE-5_2, ARC-5pFE-5_3,
ARC-6pFE-6_1, ARC-6pFE-6_2}

k-Bounded places : {3* placem, 3* convey, 6* finish}
Unbounded places : {}

Net under analisys is not strictly conservative.

Multi-enabled Tr. : {}

Net under analisys is not live.

Live Tr. : {}
"Almost-live" Tr. : {EV-1, EV-2, EV-3, b3, tin_EA-1, tEA-1_81,
tEA-1_82, tEA-1_83, b5, b6, b7, b8, Fn2,
iDP-2BP-1, iDP-2BP-2, iDP-2BP-3, b9, b10, b12,
b13, St1, Fn3, iBP-1EA-2, iBP-1EA-3, b14, b15,
b16, b17, b18, b19, St2, Fn4, iBP-2EA-5,
iBP-2EA-6, iBP-2EA-7, iBP-2EA-8, b20, b21, b22,
St3, Fn5, iBP-3EA-9, iBP-3EA-10, tin_EA-2,
tout_EA-2, tEA-2_86, tEA-2_87, tin_EA-3,
tout_EA-3, tEA-3_100, tEA-3_101, tEA-3_102,
tin_EA-5, tout_EA-5, tEA-5_110, tEA-5_111,
tin_EA-6, tout_EA-6, tEA-6_124, tEA-6_125,
tEA-6_126, tin_EA-7, tout_EA-7, tEA-7_130,
tEA-7_131, tin_EA-8, tout_EA-8, tEA-8_134,
tEA-8_135, tin_EA-9, tout_EA-9, tEA-9_138,
tEA-9_139, tin_EA-10, tout_EA-10, tEA-10_146,
tEA-10_147, tEA-10_148, ARC-1tFE-1_1,
ARC-1tFE-1_2, ARC-2tFE-2_161, ARC-2tFE-2_162,
ARC-2tFE-2_163, ARC-2tFE-2_164, ARC-3tFE-3_83,
ARC-3tFE-3_84, ARC-4tFE-4_1, ARC-4tFE-4_2,
ARC-4tFE-4_3, ARC-4tFE-4_4, ARC-5tFE-5_1,
ARC-5tFE-5_2, ARC-5tFE-5_3, ARC-5tFE-5_4,

ARC-6tFE-6_1, ARC-6tFE-6_2}
Non-fired Tr. : {b4, Fn1, iDP-1EA-1, tout_EA-1, b11, iBP-1EA-4, tEA-3_103,
tin_EA-4, tout_EA-4, tEA-4_106, tEA-4_107, tEA-6_127,
ARC-1tFE-1_3, ARC-1tFE-1_4}

None state can have another beginning.

No "live-locks" detected.

States (and fire sequencies) in "dead-lock":
C29 :EV-1 b3 tin_EA-1 tEA-1_81 ARC-1tFE-1_1 ARC-1tFE-1_2 tEA-1_82
tEA-1_83 EV-3 b5 St1 b9 tin_EA-2 tEA-2_87 ARC-2tFE-2_161
ARC-2tFE-2_162 tEA-2_86 tout_EA-2 iBP-1EA-2 b10 tin_EA-3 tEA-3_101
ARC-2tFE-2_163 ARC-2tFE-2_164 tEA-3_100 tEA-3_102 tout_EA-3
iBP-1EA-3 b12
C87 :EV-1 b3 tin_EA-1 tEA-1_81 ARC-1tFE-1_1 ARC-1tFE-1_2 tEA-1_82
tEA-1_83 EV-3 b5 St1 b9 tin_EA-2 tEA-2_87 ARC-2tFE-2_161
ARC-2tFE-2_162 tEA-2_86 tout_EA-2 iBP-1EA-2 b10 tin_EA-3 tEA-3_101
ARC-2tFE-2_163 ARC-2tFE-2_164 tEA-3_100 tEA-3_102 tout_EA-3
iBP-1EA-3 b13 Fn3 iDP-2BP-1 b6 St2 b14 tin_EA-5 tEA-5_111
ARC-3tFE-3_83 ARC-3tFE-3_84 tEA-5_110 tout_EA-5 iBP-2EA-5 b15
tin_EA-6 tEA-6_125 ARC-4tFE-4_1 ARC-4tFE-4_2 tEA-6_124 tEA-6_126
tout_EA-6 iBP-2EA-6 b16 tin_EA-8 tEA-8_135 ARC-4tFE-4_3 ARC-4tFE-4_4
tEA-8_134 tout_EA-8 iBP-2EA-8 b19 Fn4 iDP-2BP-2 b7 St3 b20 tin_EA-9
tEA-9_139 ARC-6tFE-6_1 ARC-6tFE-6_2 tEA-9_138 tout_EA-9 iBP-3EA-9
b21 tin_EA-10 tEA-10_147 ARC-5tFE-5_3 ARC-5tFE-5_4 tEA-10_146
tEA-10_148 EV-2 tout_EA-10 iBP-3EA-10 b22 Fn5 iDP-2BP-3 b8 Fn2

A-6.3.Report on a Step-by-Step Execution (for batch-size = 1)

Evolution Register of Net Simulation smt.

----> Register on.
Depth : 0 (Memorized as Inicial State).
Marking : {pEV-1, FE-1, FE-2, FE-3, FE-4, FE-5, FE-6, ARC-1pFE-1_1,
ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1, ARC-5pFE-5_1,
ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {EV-1}

-> EV-1 fire at t = 0.
Depth : 1.
Marking : {pib3, FE-1, FE-2, FE-3, FE-4, FE-5, FE-6, ARC-1pFE-1_1,
ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1, ARC-5pFE-5_1,
ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {b3}

-> b3 fire at t = 0.

Depth : 2.
Marking : {piDP-1EA-1, piEA-1, FE-1, FE-2, FE-3, FE-4, FE-5, FE-6,
ARC-1pFE-1_1, ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {tin_EA-1}

-> tin_EA-1 fire at t = 0.
Depth : 3.
Marking : {piDP-1EA-1, pEA-1_131, FE-2, FE-3, FE-4, FE-5, FE-6,
ARC-1pFE-1_1, ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {tEA-1_81}

•
•
•

-> ARC-1tFE-1_2 fire at t = 30.
Depth : 6.
Marking : {piDP-1EA-1, pEA-1_132, pFO-2, FE-2, FE-3, FE-4, FE-5, FE-6,
ARC-1pFE-1_1, ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {tEA-1_82}

-> tEA-1_82 fire at t = 0.
Depth : 7.
Marking : {piDP-1EA-1, pEA-1_133, FE-2, FE-3, FE-4, FE-5, FE-6,
ARC-1pFE-1_1, ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {tEA-1_83}

-> tEA-1_83 fire at t = 0.
Depth : 8.
Marking : {pEV-3, piDP-1EA-1, pEA-1_134, FE-2, FE-3, FE-4, FE-5, FE-6,
ARC-1pFE-1_1, ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {EV-3}

-> EV-3 fire at t = 0.
Depth : 9.
Marking : {pib5, piDP-1EA-1, pEA-1_134, FE-2, FE-3, FE-4, FE-5, FE-6,
ARC-1pFE-1_1, ARC-2pFE-2_264, ARC-3pFE-3_137, ARC-4pFE-4_1,
ARC-5pFE-5_1, ARC-6pFE-6_1, 3* placem, 3* convey, 6* finish}
Bounds : {b5}

•
•
•

Appendix 7: "1993 Donald Julius Groen Prize" for the article of the year in the area of communications and control, awarded by the IMechE

The Institution of Mechanical Engineers

I MECH E

Computing and Data Communications Group

Donald Julius Groen Prize

We hereby certify that the 1993 Donald Julius Groen Prize was awarded by the Computing and Data Communications Group of the Institution to

M W C Aguiar and Dr R H Weston

for their paper entitled

CIM-OSA and Stochastic Time Petri Nets for Behavioural Modelling and Model Handling in CIM Systems Design and Building

which was published in the Proceedings Volume 207, 1993

President

Director General and Secretary

ON BEHALF OF
Chairman
Computing and Data Communications Group

22 April 1994

Appendix 8: Description of the Business Entity Scenario Diagram

The following description details the execution of the model shown in Figure 51 by the business entity, according to the scenario diagram depicted in Figure 52.

In Figure 52 a thread of business model execution starts when EV-1 is generated. EV-1 can be either generated from an exhogenous happening coming from an external domain (e.g. a change in a certain object stored on a database) or simulated by the designer (i.e. Event Handler in manual mode at debugging time) through the interface depicted in Figure 47. In which case, the Event Handler generates an occurrence number (i.e. field oc in Figure 52) and a sequence number associated with the event (e.g. EV-1 1); composes a message and sends it across to the Process Controller (i.e. message number 1). Here, it is important to emphasise that more than one event can be sent by the Event Handler to the Process Controller at the same time (regardless to whether they have been generated automatically or by the designer) and they comprise a single thread of the business model (i.e. a single "oc" number), but they are identified by unique numbers (e.g. EV-1 1 and EV-1 3). Another related feature of the business entity at this stage is its ability to generate unique identifiers (which are never duplicated) for occurrences of events and threads of executions of the business model.

The event received from the Event Handler is queued up by the Process Controller and examined in conjunction with events already in the queue, in order to identify domain processes that are triggered by any event or combination of events already latched in the queue. Event occurrences remain in the queue until they have triggered each and every domain process that consumes them. Here, it is important to observe that a certain event may be consumed by two or more domain processes in different ways. For example, in Figure A-2, EV-3 triggers two domain process. DP-3 can be triggered by a simple occurrence of EV-3, whereas DP-4 is only triggered by a combination of EV-3 and EV-4 (in this case, it is assumed that they are combined by an AND logical operation). However OR operations are also possible). This means that EV-3 will remain in the queue until EV-4 occurs, so that for the same occurrence of EV-3 two domain process are triggered.

Coming back to the scenario in Figure 52, by scanning the process model, the Process Controller identifies that DP-1 can be triggered. Then, it spawns a local process and configures it to execute the procedural rule set of DP-1. The Process Controller assigns a unique identifier to this local process (which is generated by the process itself) and communicates with it via a Unix socket. Message number 2 characterises the spawning process performed by the Process Controller on the process termed **dp**. A **dp** process identifies a class of processes which are able to execute a procedural rule set of

a domain process. In message 2, **dp** is identified between square brackets because it is not a field that is formally passed by the Process Controller to **dp** but is simply implied by the way **dp** is spawn (i.e. through a command "popen()" in C). The remaining fields convey the following information:

- **oc**: occurrence number for the thread of execution of the business model;
- **PC**: socket name to which the DP-1 process will have to write in order to send messages to the Process Controller;
- **DP-1**: identifier of the domain process to be executed;
- **m**: row of the procedural rule set matrix which was enabled by an event (or combination of events). Here, it is important to notice that the actual event identification that caused the domain process to be triggered is not formally passed to the **dp** process. Due to PC and **dp** share the same file in which the matrix that describes the procedural rules of DP-1 are defined (see Figure 51), **dp** only needs to know which of the procedural rules was triggered, therefore, shortening this message. In this case, the value 1 is assigned to **m** is set to 1 (see Figure 51);
- **sch**: boolean variable indicating whether DP-1 has already been scheduled for execution (**sch** is usually set to 0, but it was left as a variable in this case, as a provision for a more flexible mode of operation).

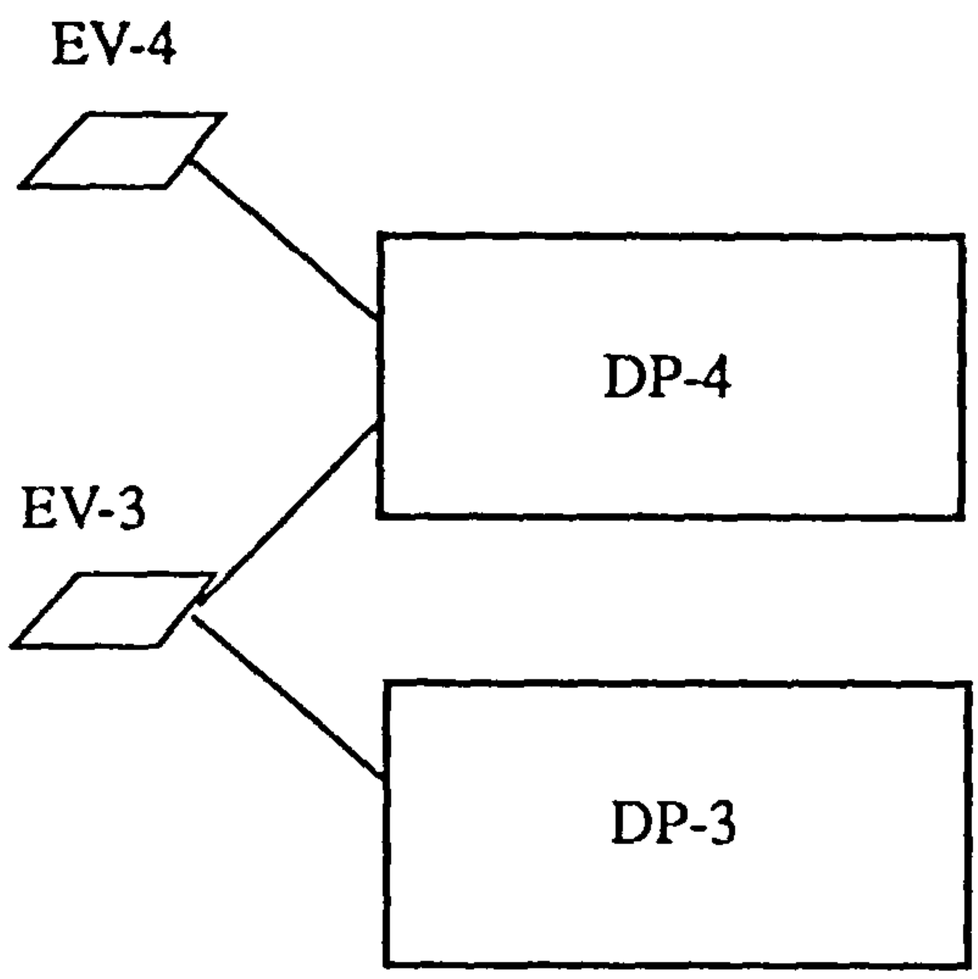


Figure A-2 - Illustration of Event Queuing Up

On creation, **dp** reads its procedural rules from the file named **pc** (see Figure 44), generates its unique identifier (based on the Unix process identification: **pid** [Back 1986]) and returns such an identification to the Process Controller (message number 3: "Report PID" - PID - see Figure 52), where:

- **DPxx** is the unique identifier which individualises this particular occurrence of DP-1 within the business entity; and
- **PID** is the message identification (see Table 7.1).

Then, **dp** starts executing its procedural rule set. The first rule to be executed is m = 1, which means that the triggering of BP-1 (i.e. "DO" part of the first procedural rule of DP-1 - see Figure 51). Before BP-1 can be triggered by **dp**, **dp** has to check with the Process Controller whether this step of the business model can be executed. This is done through message number 4: "Step Request" (STEP_REQ - see Figure 52), where:

- **n** is the number of Enterprise Functions that are about to be triggered followed by a list of their identifiers. In this case, only one Enterprise Function is about to be triggered: BP-1, thus, **n** is set to 1;
- **STEP_REQ** is the message identification (see Table 4 on page 151).

On receipt of a STEP_REQ, the Process Controller checks its operating status. If it is in automatic mode and not interrupted (see Figures 48 and 49), it sends back to **dp** the message number 5: "Step Release" (STEP_REL). Otherwise, it waits for a command from the designer (see Figure 49) to either: execute the function (if it is in step-by-step model) or resume execution (if the execution was interrupt). The pair of messages, STEP_REQ and STEP_REL, is the basic means by which the debugging functions of the Process Controller are implemented. These messages are required in order to allow the Process Controller to interrupt the execution of a procedural rule set of its children processes.

On receipt of a STEP_REL, **dp** checks whether the functions that are about to be triggered require scheduling (i.e. whether the variable **sch** is set to "0"). If so, **dp** sends a "Resource Request" to the Process Controller (message number 6 - RES_REQ - see Figure 52). The Process Controller, in turn, forwards this request to the Resource Manager (message number 7 - Figure 52). The Resource Manager then access its scheduling functionality to check whether it can allocate the resources for the whole duration of the function (or functions) about to be executed. If it can, a "Resource Release" (message number 8 - RES_REL - Figure 52) is issued by the Resource Manager to the Process Controller with its **sch** variable set to 1. Otherwise, **sch** is set to 0. The RES_REL information is passed back to **dp** by the Process Controller.

sch indicates whether there is a need to schedule the functions contained within a certain function. For example, a **sch** set to 1 means that the functions contained within DP-1 do not need to request resource allocation to the Resource Manager, in the same way that in the description above **dp** had request resource allocation for DP-1. **sch** set to 0 implies the allocation of resources by the Resource Manager was postponed. It is

important to notice in Figure 52 that messages 6 to 9 all carry the same related information in its fields, concerning the identification of the **dp** occurrence (i.e. DPxx) and the functions are being scheduled (i.e. BP-1).

In this example, sch = 0 is passed down to **bp**, when **dp** ultimately spawns BP-1 (message number 10). **bp**, likewise **dp** in the above description, reads in its procedural rule set from the **pc** file (see Figure 44) and starts executing it by the rule beginning with "ON START" (see Figure 51), which provokes the triggering of EA-1. The same procedure described for **dp** applies to **bp**, in regard to the execution of a step in the business model and the functional scheduling (i.e. messages 12 to 17 in Figure 52) for the function EA-1. This time **sch** is set to 0, which means that the execution of EA-1 will be subject to the availability of resources to execute its Functional Operations. In fact, in this implementation, the Resource Manager always returns **sch** set to 0 because no scheduler is attached to it, in order to make a definite decision about functional scheduling. Therefore, the decision about whether a function will be actually executed is only made when the finest level of granularity is reached (i.e. Functional Operations), based on the availability of an active resource component able to execute them.

Having received the RES_REL message for EA-1, **bp** issues a request for the execution of EA-1 to the Process Controller (i.e. "EA execute" - message number 18 - EA_EXEC - Figure 52), which is then passed by the Process Controller to the Activity Controller (i.e. message number 19 - Figure 52).

The Activity Controller then spawns an occurrence of EA-1 (i.e. EA1.i, where **i** identifies the occurrence number), by triggering a CIM-BIOSYS application to execute its functional content (i.e. message 20 - Figure 52). Message 20 consists basically of a command of the CIM-BIOSYS infrastructure to start a CIM-BIOSYS application by establishing a link with it. Once triggered, EA1.i starts executing its behavioural description, as defined in the activity behaviour diagram (see Figure 31). EA1.i executes the internal processing associated with the progressive execution of its predicate-action Petri-net and interacts with the remaining processes of the business entity via message exchanges attached to predicates and actions (see Object diagram in Figure 30). Message 21 is an instance of such interaction, whereby EA1.i requests the execution of the operation FO-1 to be passed to an available active resource component. On receipt of message 21, the Activity Controller forwards it to the Resource Manager, and updates its internal variables (i.e. **oc** and **EA1.i** identifier - message number 22). On receipt of a request to execute a functional operation from Activity Controller, the Resource Manager checks whether there an active resource component (ARC) is available which is able to execute it, based on the information defined in the Resource diagram (see Figure 33). If no ARC is available, the functional operation is queued up until an ARC is released. If an ARC exists which can be allocated for executing the functional operation, the Resource Manager passes the functional operation identifier

to the active resource component (message number 23 - Figure 52).

One should notice here that there is only a limited number of active resource components to serve enterprise activities whose quantity are, 'a priori', only limited by the maximum number of occurrences of the business model. Such a number is dependent upon run-time occurrences which can be unlimited. The manner by which active resource components can cope with the number of enterprise activities that they expected to execute is defined by their schedule. Additionally, it is envisaged that active resource components may be able to support the execution of functional operations issued by competing enterprise activities in the following alternative ways:

(1) Constraint on functional operations:

a. execution of only one functional operation at a time (i.e. the Resource Manager is only allowed to request the active resource component to execute a functional operation when the previous operation was already completed;

b. simultaneous execution of functional operations of different type at the same time (i.e. the Resource Manager is allowed to request the active resource component to execute one or more functional operations without waiting for a previous functional operation to be completed, as long as they all are of different types);

c. simultaneous execution of functional operations regardless of any restriction of type. This is a super-set of the previous one in which, in addition to supporting simultaneous execution of functional operations of different type, the execution of occurrences of functional operations of the same type are also supported by the active resource component.

(2) Constraint on enterprise activities:

a. execution of functional operations issued by the same enterprise activity to which the resource is currently allocated. That is an active resource component will not execute functional operations of an enterprise activity occurrence other than the one that it is currently serving. This means that when an enterprise activity finishes executing its functionality, it must report its completion to the Resource Manager;

b. no constraints on the execution of functional operations issued by distinct Enterprise activity occurrences which, therefore, can use the active resource component at the same time. Here, a further distinction could be made such that an active resource component could execute functional operations of enterprise activity occurrences of different types or of the same type.

Combinations of these two sets of constraints are also envisaged, in which

constraints on enterprise activities superimpose constraints on functional operations. Although, these options have not been foreseen in the CIM-OSA specifications, it is a thrust of this research that they should be part of the business entity, for they impose important restrictions in the way system works. Nonetheless, in this implementation it was decided to adopt only constraints on enterprise activities of the type (2.a) above.

Therefore, coming back to the scenario shown in Figure 52, on receipt of message 23 (i.e. FO-1), the activity resource component executes the functional operation according to the model defined in the entity behaviour diagram (see Figure 32). In this particular case, the cyclic behaviour of ARCj consists of waiting for a command FO-1 and issuing a response FO-2 (i.e. message 24 in Figure 52). Conversely, the behaviour of its enterprise activity counterpart consists of issuing a command FO-1 and waiting for a response FO-2. Such a protocol can be observed in messages 21, 23, 24 and 26 (see Figure 52). In these messages, fields between square brackets (i.e. ARCj and EA1.i) identify particular occurrences of enterprise activities and active resource components. These identifiers are not visible by their counterparts. That is, the enterprise activity occurrence has no knowledge of which active resource component is executing its functionality. Likewise, the active resource component has no knowledge of which enterprise activity it is interacting with. This clearly illustrates the separation between functional and resource models, whose relationships are captured in the business model and jointly managed by the Activity Controller and the Resource Manager.

Once EA1.i has received FO-2 (message number 26 in Figure 52), it completes its internal processing, generates an occurrence of the event EV-2 (message 27), and terminates (message number 28). Events generated by the enterprise activities[1] are passed to the Event Handler (message number 29 in Figure 52). Termination of EA1.i is manifest through issuing an ending status (message number 28) and disappearance from the business entity. The ending status is calculated according to the state variables perceived by the internal behaviour of the enterprise activity (see Figure 31). Its default value is "done"[2]. On receipt of message 28, the Activity Controller updates its internal controls (which keeps track of all the information associated with the enterprise activity occurrence that has just been terminated), passes the ending status to the Resource Controller (i.e. message number 30 in Figure 52), which enables it to update its internal information as well, and reports the ending status to the Process Controller (message

---

1. Enterprise activities are the only constructs allowed to generate events within the Business Entity.
2. Here, one should observe that enterprise activities are the only constructs in the business entity which are able to generate an ending status other than "done".

number 31 in Figure 52). The Process Controller, in turn, forwards the ending status to **bp** (message number 32). **bp** then executes another step of its procedural rule which causes it to finish its procedural rule execution (see Figure 51). This is done when **bp** reports the ending status of BP-1 to its parent function (i.e. DP-1) via an "Enterprise Function Ends" (message number 33 - EF_END) and terminates itself. **dp**, in turn, passes the status of BP-1 to the Process Controller, in order to report it to the designer (message number 34: "enterprise function status" - EF_STATUS) and executes another step of its own procedural rule. This causes **dp** to complete its procedural rule execution with the issue of ending status, which is also passed back to the Process Controller (message number 35 - "domain process ends" - DP_END). **dp** then terminates itself. On receipt of a DP_END, the Process Controller updates its information about running domain processes.

On receipt of message 29, the Event Handler then checks whether EV-2 happens to trigger any domain process. If so, another occurrence of the business model is generated (i.e. field **new_oc** in message number 36 in Figure 52) and passed back to the Process Controller. If not, EV-2 is forwarded to the external world (i.e. non-CIM-OSA compliant domains. This is illustrated by the message number 35a which is an alternative to message 35 - see Figure 52).

Appendix 9: Relationship of SEW-OSA with Other D2D Systems

The discussion presented in Chapter 8, associated with the quantitative results discussed in Chapter 11, constitute the activities performed within the case study application of SEW-OSA to D2D shop-floor. This section aims to:

- situate the domain addressed in the case study in the context of other information systems of D2D;
- present some general (but unproven) ideas about how SEW-OSA can be an integral part of a wider-scope model-driven system at D2D.

The motivation for this section was to illustrate the role that SEW-OSA can play within the "big-picture" of D2D manufacturing operations.

A-9.1.Shop-Floor Scheduling

The architecture shown in Figure 64 does not itself address wider scope shop-floor scheduling issues. For that purpose, a hierarchy of schedulers could be used whose scope of decision might coincide with the groupings of line segments shown in Figure 58 (i.e. lines, cells and areas). Indeed, the four layer hierarchy shown in Figure A-3 is suggested by the author. The line segment schedulers shown in Figure 64 operate at the bottom level of this hierarchy. The second layer is responsible for either deciding which line segment of a cell should be allocated to a certain job (i.e. cell schedulers) or controlling the operation of a line (i.e. operate as a line controller). One should notice here that no scheduling decision is required within the scope of a line. Arguably, in this case line segments could be merged into a unique line segment.

Cell schedulers can then be integrated to an area scheduler whose scope coincides with the physical division of the shop-floor into areas (such as for high-volume production and high-mix production, as depicted in Figure 58). Finally, area schedulers are integrated to a shop-floor scheduler whose scope of control embraces the complete shop-floor.

The CIM-BIOSYS integrating infrastructure can realise a physical integration of the components of such a hierarchy of schedulers and it is not necessarily for this to be achieved via SEW-OSA. As the scheduling function is not included in the CIM-OSA modelling process, it does not account for interactions which occur within the scope of line segment schedulers with which it interacts. An architecture for hierarchical factory control usually envisages that their components operate as serves and clients during their life cycle. Indeed, if the communication protocols prescribed in such architectures use CIM-BIOSYS application services for their implementation, this being

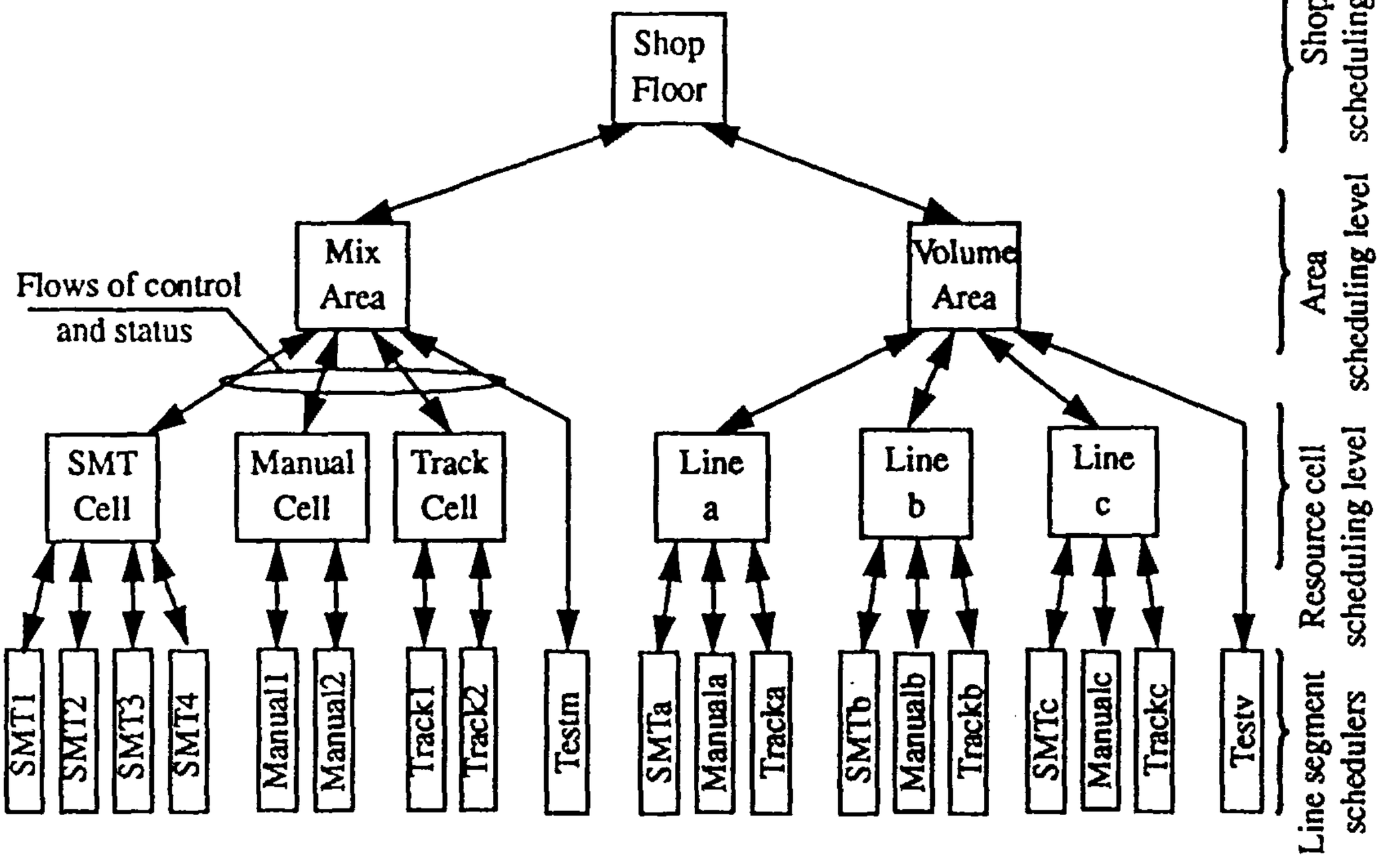


Figure A-3 - Hierarchical scheduling of the shop-floor

independent of SEW-OSA's entities, then the overall stack of services required should be organised in the manner shown in Figure A-4. Here, an essentially flat computer architecture would exist, in which every element of Figure A-3 will utilise similar services to communicate with peers, subordinates and supervisors. However, the resultant effect will need to be transformed by the internal functionality of each element as part of an organised hierarchy which provides scheduling services in a distributed manner.

In this context, these two types of architectures (i.e. the CIM-OSA and factory control architectures), which are of an inherently different nature and origin, can interoperate, as the line segment schedulers can function as the only actual interface between them at run-time. From the viewpoint of SEW-OSA, the scheduling services are provided locally by the line segment schedulers. From the view of the hierarchy of schedulers, the scheduling decisions are being used to directly control a shop-floor.

A-9.2. Integration of Shop-Floor, Process Planning and Production Planning and Control

Another thrust of this research was to investigate the use of SEW-OSA can as a bridge amongst the functions of process planning (i.e. SPEAR[1]), production planning and control and shop-floor. This section considers how this form of integration can be

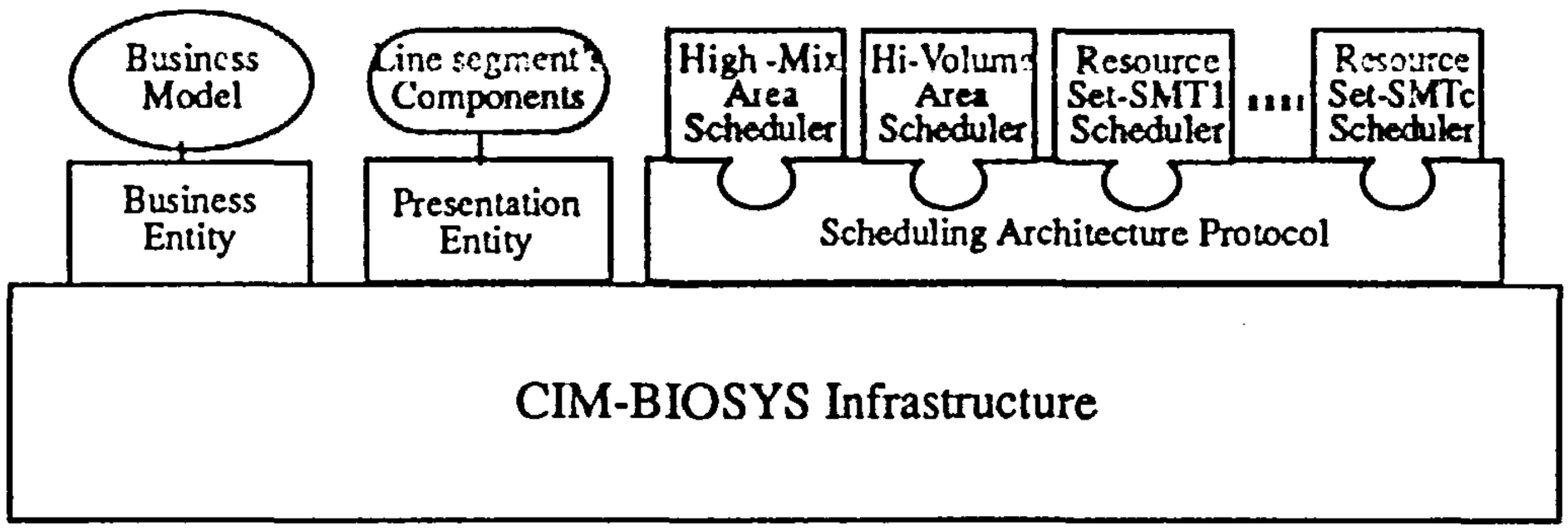


Figure A-4 - Organisation of services for integrating the architectures

achieved for current systems used by D2D.

Figure A-5 illustrates a relationship between SEW-OSA and these enterprise functions. SEW-OSA can import data from production planning about production orders (such as those issued by an Orders Book for a given production period). This could include data about due dates, production quantities, priorities, and time and motion information. From process planning, SEW-OSA can input manufacturing data associated with each PCB type. This could include data on: viable routes, resource capability, resource set-ups, etc. By combining these two sets of data a very useful model of the shop-floor can be constructed and tested within SEW-OSA (by means of model-enactment), before possible solutions are deployed on the shop-floor. The SEW-OSA studies could include analyses of:

- alternative allocations of line segments to produce certain PCB's;
- alternative configurations of line segments;
- various scheduling scenarios.

In order to support study of various scheduling scenarios, the hierarchy of schedulers must have access to information required for scheduling decisions. This could then be as delivered by the model-building capability of SEW-OSA, via the business model.

However, a more generalised discussion of how integration between the functions of Process Planning and Process Control can be unified to produce model-driven solutions is beyond the scope of this research.

---

1. SPEAR is a major D2D project which focuses on formalising process planning activities in the enterprise to enable the development of information systems which support an integrated design-to-manufacturing approach to product introduction, thereby reducing product introduction lead times.

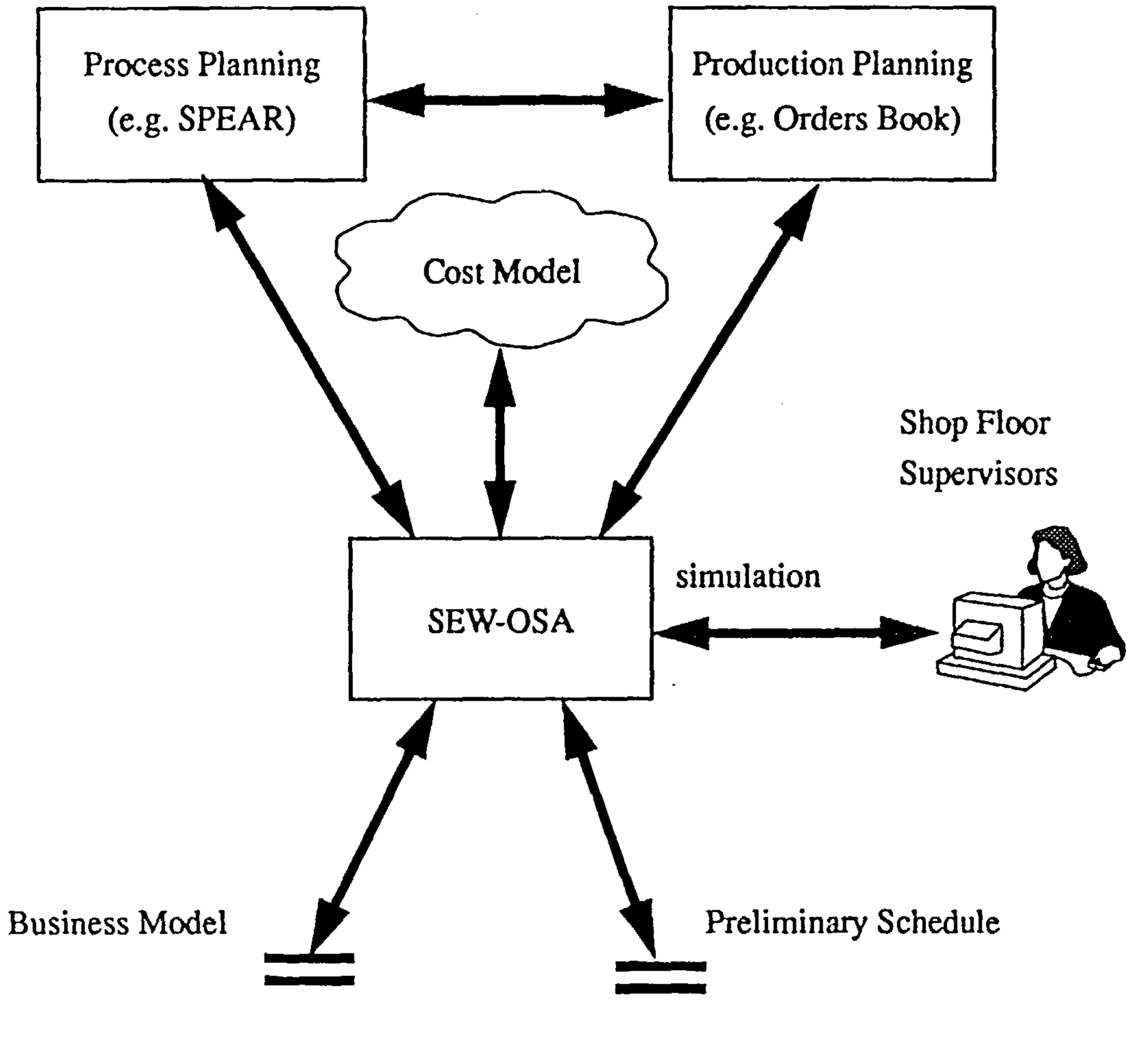


Figure A-5 - Integration to process planning, and production planning and control

D2D has progressively moved towards acting as a sub-contract manufacturer of PCB's. Hence, a greater proportion of their production is on assembling boards designed by other parties. As a result of much wider range of designs need to be manufactured and the shop-floor needs to be configured more frequently. This trend obviates a need for improved systems analysis and means of achieving flexible integration (and hence improved coordination of manufacturing processes), through use of an approach such as SEW-OSA.

Another opportunity indicated in Figure A-5 stems from the possibility to link models manipulated by SEW-OSA to cost information. The rationale for such a link stems from the need to rapidly respond to request from customers to produce new or existing designs of PCB's. At D2D such a response consists of informing a customer whether a design can be manufactured within the required time-scales and, if so, the effort (in terms of cost) required to design and manufacture the board. It has been observed that traditional ways of costing products often do not accurately account for the resources used and overhead consumed when in production [Shaharoun 1994] [Turney 1991].

Therefore, as part of this research, the author has considered (but not yet

implemented) ways of combining SEW-OSA with Activity Based Costing (ABC) methodologies for assessing cost related effort when producing a product. ABC facilitates a consideration of factors often disregarded in the traditional cost accounting practice. Indeed, it can be particularly useful when considering:

- the cost of specials[1] (e.g. when elaborate test procedures or specialised components);
- the impact on cost of producing a board in the context of a certain product mix (e.g. the cost of re-scheduling).

The use of ABC can provide a useful means of obtaining performance metrics from simulation models, which could be used to assess the merits of different shop-floor configurations. As a result, changes in the models manipulated by SEW-OSA could be analysed.

Combining the considerations outlined in this appendix would result in the configuration presented in Figure A-6.

---

1. Products which require special treatment.

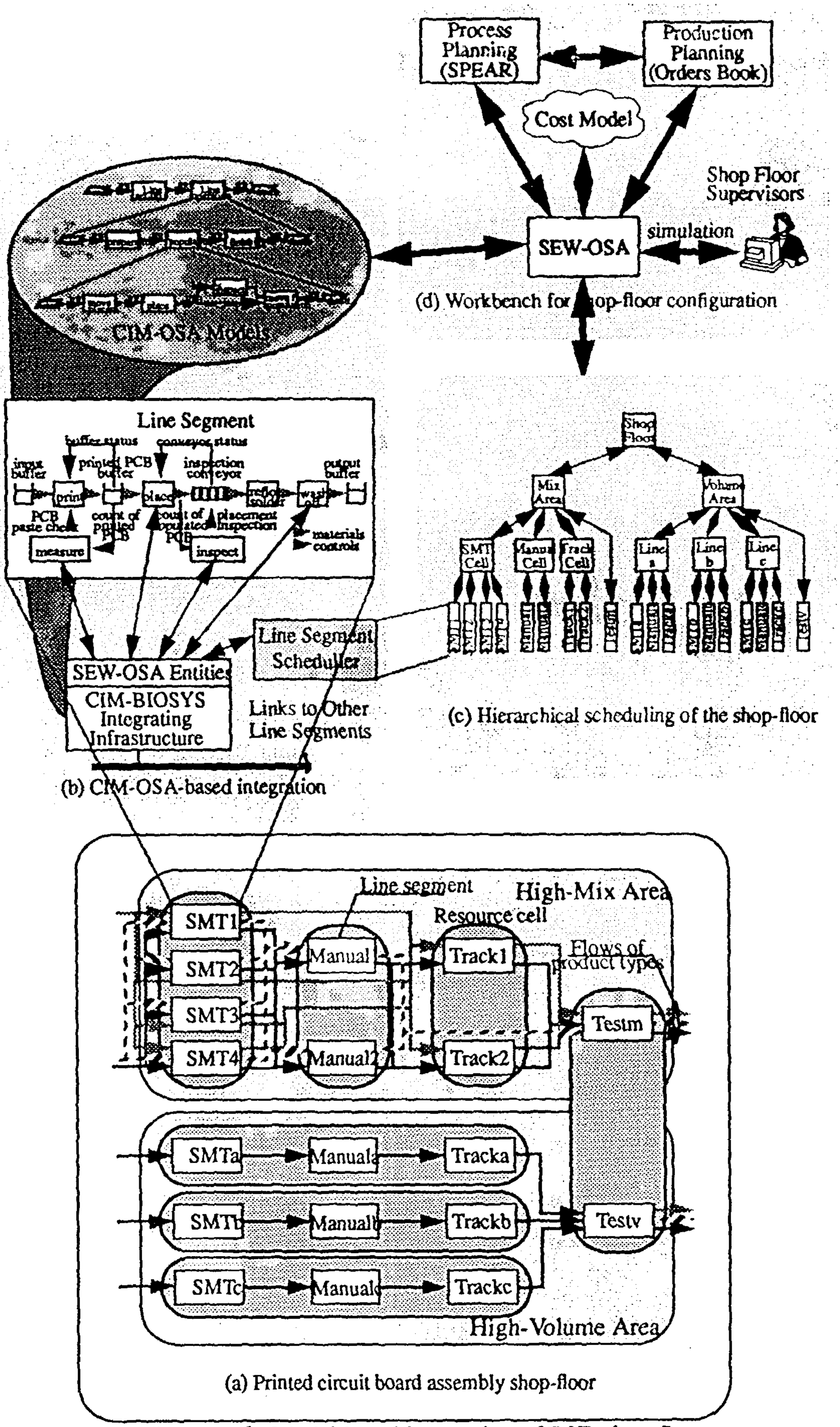


Figure A-6 - Model-based integration of D2D shop-floor

Appendix 10: Data Associated with Modelling and Simulation Studies

A-10.1.Definition of the Metrics Associated with Results of the Model-Building Work

The following definitions refer to the curves presented in Figure 85:

a. Number of constructs in the model.

This consists of the sum of the main constructs of interest at the requirements definition modelling level (i.e. domains, domain processes, business processes, enterprise activities, object views and events) with those at the design specification modelling level (i.e. functional entities, active resource components, functional operations and Petri-net transitions and places). This measure is a simplified representation of the complexity of a model, for it does not include the details about how the constructs are inter-related. Additionally, since the number of constructs at the design specification modelling level depends upon the number at the requirements definition modelling level, the latter could have also been used as a measure of complexity of a model. In this respect, the count of constructs at the design specification modelling level includes two levels of constructs, namely: functional entities, active resource components and functional operations, at a higher level and details of activity and entity behaviour diagrams (i.e. transitions and places) at a lower level. Table A-1 presents these values and the form in which they have been aggregated. It can be noticed from this Table that the different sets of measures of model complexity are related to one another in direct proportion, such that any of the proposed measures could have been used as an indication of complexity of a model.

Table A-1 - Number of constructs in a model

Models	**Total**	**Requirements Definition**	**Design Specification**	**FEs, FOs and ARCs**	**transitions and places**
MSI Institute	42	11	34	10	24
SMT assembly line	140	30	111	32	79
D2D shop-floor	253	68	185	58	127

b. Code for simulation:

This is a measure of the size of a Petri-net representing the behavioural aspects of a model in the ARP syntax (see Figure 41). This measure is directly related to the size and complexity of the models formalised in the behaviour diagrams (see Figure 27) and activity/entity behaviour diagrams (see Figures 31 and 32). This measure is also an indication of the number of relationships that exist between the

constructs manipulated in these diagrams. Therefore, this measure complements the previous one, which did not capture relationships between constructs.

As discussed in Chapter 6, simulation is enabled by transforming the models into a Petri-net. An indication of size and complexity of a Petri-net is directly related to its number of places and transitions. Table A-2 summarised these metrics for the three models in question. Here, one should notice that the model of the SMT assembly line is near the limit of maximum number of places supported by the ARP (i.e. 150 places), whereas the model of the D2D shop-floor is well beyond this limit. This limitation and the limitation of stemming from the overhead imposed by the business entity (discussed in Chapter 11) were the main reasons why the model-enactment capability was only tested with the model of the SMT assembly line.

Table A-2 - Code generated for simulation

Models	LSC	transitions	places
MSI Institute	102	27	39
SMT assembly line	373	110	144
D2D shop-floor	253	235	277

c. Code for rapid-prototyping:

This is a simplified measure of the size of a system prototype produced with the SEW-OSA CASE tool. This measure consists of adding the number of lines of code and structured text produced by the CASE tool, namely: list of events (ev), list of object views (ov), process behavioural definition (pr), activity behavioural definition (ac), resource definition (rc) and the configuration of the CIM-BIOSYS infrastructure (cf) - see Figure 44. Table A-3 presents the values obtained for these fragments of code.

Table A-3 - Code generated for rapid-prototyping

Models[a]	Total	ev	ov	pr	ac	rc	cf
MSI Institute	213	6	7	18	57	82	43
SMT assembly line	603	7	15	79	231	138	133
D2D shop-floor	1150	19	29	65	378	433	226

a. all units are in lines of code and structured text definition.

d. Model-building to code generation lead-time:

This is an indication of the usability of the CASE tool. Usability is of interest for it can impact the complexity of the model-building process. The values indicated in Figure 85 were obtained by measuring the total time required to input these models into the CASE tool. This time does not include any data gathering activity at D2D and assumes a considerable familiarity by the user with the SEW-OSA method encapsulated by its CASE tool[1].

In any case, it is important to note that these values include the definition of all diagrams of the SEW-OSA method (at the requirements definition and design specification stages - see Figure 22), being approximately half the total lead-time spent in each modelling level. This reinforces the argument presented in Chapter 10 in favour of reference models to support the model-building process, particularly the definition of functional entities, enterprise activities and their behaviour diagrams, in order to reduce considerably the total lead-time.

A-10.2.Impact of Number of Slots in the SMT Assembly Line

A study of the impact that different values of number of slots have upon the manufacturing (i.e. assembly lead time) for the parameters presented above is shown in Table A-4. The percent difference shown in this Table is calculated by having a distribution of infinite slots (i.e. first row) as a reference[1]. As one can observe the impact of this distribution is fairly small, although its consideration is important in order to reflect a physical constraint of the line.

Table A-4 - Code generated for rapid-prototyping

Distribution of slots	Lead-Time	Difference (%)
infinite capacity	3631.57	0
3 placement, 3 conveyor and 6 finishing	3709.84	2
3 placement, 3 conveyor and 3 finishing	3836.82	6

A-10.3.Details on the Complexity of SEW-OSA

Table A-5 presents a detailed analysis of the main content of the SEW-OSA CASE tool. In this Table, it is important to notice that:

- nearly half the size of the CASE tool contains libraries produced by ToolBuilder with functions common to all CASE tools;
- from the other half (i.e. the constructs of the SEW-OSA method - 15751 LSC), the greatest proportion of the code is dedicated to the requirements definition modelling level (i.e. 56%), being the domain diagram and the behaviour diagram the largest in the tool. This is the case due to the complexity of the CIM-OSA constructs formalised at this modelling level.

---

1. These measures were obtained with the author serving as the user.
1. Difference = [lead-time - (reference lead-time)] x 100/(reference lead-time).

Table A-5 - Size of the SEW-OSA CASE tool[a]

CASE tool modules	LSC	%
Complete CASE tool	**30221**	-
ToolBuilder libraries	14500	-
SEW-OSA constructs	**15751**	**100**
General modules	1575	10
Requirements definition tool	**8821**	**56**
- context diagram	1575	10
- domain diagram	2205	14
- structure diagram	788	5
- behaviour diagram	2205	14
- functional diagram	2048	13
Design specification tool	**5355**	**34**
- object diagram	1418	9
- resource diagram	945	6
- configuration diagram	1575	10
- entity/activity behaviour diagram	1418	9

a. The values of LSC were calculated as a percentage of the number of entities in the CASE tool data model.

Table A-6 summarises the main components of the business entity of SEW-OSA along with indications of their size in terms of number of lines of source code (LSC) as well as the size of their resulting executable code. It is important to note in this table that the total size of the business entity includes functions that are shared among components. These functions amount about 5429 LSC (i.e. 13% of the business entity). Hence, the total size of the business entity, excluding the functions shared among components, is 42761. As a measure of comparison the CIM-BIOSYS infrastructure contains 26427 lines of source code (i.e. 62% of the business entity) which result in 688kbyte of executable code.

In regard to a comparison amongst the business entity components (see Figure 52), it can be observed in Table A-6[1] that, with the exception of the Prolog based component (i.e. the enterprise activity occurrences and the active resource components), the remaining components possess a similar degree of complexity. However, if we consider the domain process and business process occurrences as an integral part of the process controller, this component becomes the largest of them all. In fact, the size resulting from the aggregation of these components reflects the

1. This table used the total size of the business entity, inclusive of shared functions, in order to enable the sum of percentages to total 100%.

complexity involved in the process of enacting behaviour diagrams (see Figure 52) which is the key functionality provided by this component.

Table A-6 - Size of the business entity components

Element	Executable code (kilobyte)[a]	LSC	% of LSC of the business entity
Complete Business Entity	-	42761	100
Event Handler	295	4177	10
Process Controller	360	7583	18
Activity Controller	311	5329	12
Resource Manager	311	5153	12
domain process occurrence	147	4623	11
business process occurrence	123	4657	11
enterprise activity/resource component	483	11239	26

a. With the exception of enterprise activity occurrences and active resource components, the size of executable code presented here is for components that use dynamic links to other "C" libraries.

As introduced in Chapter 7, the business entity components can be organised based on the types of functions that they contain, according to the modules presented in Table A-7. This table also provides an indication of the percentage of contribution the total size of a component (as shown in Table A-6) of each of its major building blocks. It should be noted in this table that:

Table A-7 - Content of the business entity components (%)

Functions/ Interfaces	Event Handler	Process Control-ler	Activity Control-ler	Resource Manager	Activity/ Resource	domain process	business process
Total	100	100	100	100	100	100	100
Core function	9	20	17	16	80	29	29
CIM-BIOSYS	24	13	20	20	20	-	-
X-Windows	34	20	25	26	-	-	-
Model input	33	40	38	38	-	39	39
Unix comm.	-	7	-	-	-	32	32

- not all components contain all the modules, namely: functions for inter-process communication in Unix are only used by between the Process Controller and its processes; domain and business processes do not communicate via CIM-BIOSYS and do not possess a user interface; and enterprise activity occurrences and active resource components do not possess a sophisticated user interface (see Figure 50) and aggregate model-input functions into their core functionality.

- X-Windows interface (or Unix communication) and model-input functions account for the greatest proportion of code in the component. Model-input comprise the functions required to transform the code generated by the case tool into "C" structures manipulated by the functionality of the component.

The modules that comprise enterprise activity occurrences and active resource components are developed in Prolog and "C". However, the greatest proportion of the code (i.e. 93%) is written in "C" (see Table A-8).

Table A-8 - Languages of enterprise activity occurrences and active resource components

Language	LSC	%
Total	11239	100
Prolog	771	7
"C"	10468	93

Appendix 11: Mathematical Study of Overhead

A mathematical model for LT_{rp} was obtained by examining all the happenings executed by the business entity in order to enact a model. This exam was based on the description presented in Chapter 7, specially with regard to the transactions depicted in Figure 52.

Basically, the actions and transactions required to execute each construct major construct formalised in the model were identified, as follows:

a. For each functional operation send by an enterprise activity occurrence to an active resource component:

- 1 CIM-BIOSYS message[1] from the enterprise activity occurrence to the Activity Controller (t_{CBS});
- 1 CIM-BIOSYS message from the Activity Controller to the Resource Manager (t_{CBS});
- delay in the Resource Manager due to resource allocation (t_{aloc});
- 1 CIM-BIOSYS message from the Resource Manager to the active resource component (t_{CBS});
- delay in the active resource component due to emulation of the manufacturing process time (TM);
- delays in the active resource component related to the required time interval between transitions due to limitations in the Prolog engine ($ntr_j.t_{tr}$);

b. For each functional operation send by an active resource component to an enterprise activity occurrence:

- delay in the active resource component due to emulation of the manufacturing process time (TM);
- delays in the active resource component related to the required time interval between transitions due to limitations in the Prolog engine ($ntr_j.t_{tr}$);
- 1 CIM-BIOSYS message from the active resource component to the Resource Manager (t_{CBS});
- 1 CIM-BIOSYS message from the Resource Manager to the Activity Controller (t_{CBS});
- 1 CIM-BIOSYS message from the Activity Controller to the enterprise activity occurrence (t_{CBS});

---

1. A CIM-BIOSYS message is a message exchanged between two CIM-BIOSYS applications.

c. For each functional operation send by an active resource component to an enterprise activity occurrence:

- 1 Unix message[1] (i.e. EA_EXEC - see Figure 52) from a domain process or a business process occurrence to the Process Controller (t_S);
- 1 CIM-BIOSYS message (i.e. EA_EXEC) from the Process Controller to the Activity Controller (t_{CBS});
- 1 CIM-BIOSYS message (i.e. EF_END) from the Activity Controller to the Resource Manager (t_{CBS});
- 1 CIM-BIOSYS message (i.e. EF_END) from the Activity Controller to the Process Controller (t_{CBS});
- nEV CIM-BIOSYS messages (i.e. event generation) from the Activity Controller to the Event Handler ($nEV.t_{CBS}$);
- 1 Unix message (i.e. EF_END) from the Process Controller to a domain process or a business process occurrence (t_S);
- procedure to start a CIM-BIOSYS application representing the enterprise activity occurrence and establishing a link with it (t_{link});
- delays in the enterprise activity occurrence related to the required time interval between transitions due to limitations in the Prolog engine ($ntr_j.t_{tr}$);

d. For each business process occurrence:

- procedure to start a Unix process representing the business process occurrence (t_{SP});
- 1 Unix message (i.e. PID) from the business process occurrence to the Process Controller (t_S);
- pr_{BPi} times the following set of actions and transactions:
  - procedure to trigger a procedural rule (tpr_{BPi});
  - 1 Unix message (i.e. STEP_REQ) from the business process occurrence to the Process Controller (t_S);
  - 1 Unix message (i.e. STEP_REL) from the Process Controller to the business process occurrence (t_S);
  - 1 Unix message (i.e. RES_REQ) from the business process occurrence to the Process Controller (t_S);

---

1. A Unix message is a message transferred between two Unix processes communicating via local sockets.

- 1 CIM-BIOSYS message (i.e. RES_REQ) from the Process Controller to the Resource Manager (t_{CBS});
- 1 CIM-BIOSYS message (i.e. RES_REL) from the Resource Manager to the Process Controller (t_{CBS});
- 1 Unix message (i.e. RES_REL) from the Process Controller to the business process occurrence (t_S);

- nEF_{BPi} Unix messages (i.e. EF_END) from the business process occurrence to the Process Controller (t_S);
- 1 Unix message (i.e. EF_END for its own) from the business process occurrence to the Process Controller (t_S);

e. For each domain process occurrence:

- 1 CIM-BIOSYS message (i.e. event) from the Event Handler to the Process Controller (t_{CBS});
- procedure to start a Unix process representing the domain process occurrence (t_{SP});
- 1 Unix message (i.e. PID) from the domain process occurrence to the Process Controller (t_S);
- pr_{DPi} times the following set of actions and transactions:
  - procedure to trigger a procedural rule (t_{PR});
  - 1 Unix message (i.e. STEP_REQ) from the domain process occurrence to the Process Controller (t_S);
  - 1 Unix message (i.e. STEP_REL) from the Process Controller to the domain process occurrence (t_S);
  - 1 Unix message (i.e. RES_REQ) from the domain process occurrence to the Process Controller (t_S);
  - 1 CIM-BIOSYS message (i.e. RES_REQ) from the Process Controller to the Resource Manager (t_{CBS});
  - 1 CIM-BIOSYS message (i.e. RES_REL) from the Resource Manager to the Process Controller (t_{CBS});
  - 1 Unix message (i.e. RES_REL) from the Process Controller to the domain process occurrence (t_S);
- nEF_{DPi} Unix messages (i.e. EF_END) from the domain process occurrence to the Process Controller (t_S);
- 1 Unix message (i.e. EF_END for its own) from the business process occurrence to

the Process Controller (t_S);

- 1 Unix message (i.e. DP_END for its own) from the domain process occurrence to the Process Controller (t_S).

The equation obtained for LT_{rp}, by organising these factors is:

$$LT_{rp} = k.\ \Sigma_i\ (noc_{DPi}.\ t_{DPi})\ (2)$$

where:

- $\mathbf{noc_{DPi}}$ is the number of occurrences of the domain process DPi which have been generated in a particular thread of model execution.
- **k** is a coefficient whose value is a fraction of 1 (i.e. **k** belongs within the interval [0; 1]) which reduces the total sum $\Sigma_i\ (noc_{DPi}.\ t_{DPi})$, in order to account for a level of parallelism in the execution of the all $\mathbf{noc_{DPi}}$ occurrences of any DPi. In a situation in which there is no parallelism, k = 1.
- t_{DPi}, the time to execute a complete occurrence of the domain process DPi, is given by:

$$t_{DPi} = t_{SP} + (2 + 4.pr_{DPi} + nEF_{DPi}).t_S + 2.nEF_{DPi}.t_{CBS} + pr_{DPi}.t_{PR} + k_{DPi}.\Sigma_{DPi}\ (t_{EFi})\ (3)$$

where:

- $\mathbf{t_{SP}}$ is the time required in the Unix operating system to spawn a new process representing a domain process (or business process) occurrence. This time is related to the first messages sent by the Process Controller to a domain process occurrence or by any domain process or business process to its child business process, in order to activate it (see messages number 2 and 10 in Figure 52).
- $\mathbf{nEF_{DPi}}$ is the number of enterprise functions used by the domain process at run-time, in order to describe its internal behaviour (see Figure 27). $\mathbf{nEF_{DPi}}$ varies according to the decisions that are made at run-time by the business entity with regard to the number of functions that are actually triggered by the procedural rules of a process. This, in turn, depends upon the ending-status of these functions. The same definition applies for $\mathbf{nEF_{BPi}}$.

  To illustrate this definition, one could examine the behaviour diagram shown in Figure 27. In this diagram, if no inspection is executed in a certain occurrence of BP-2, $nEF_{BP2} = 3$, otherwise, $nEF_{BP2} = 4$.
- t_S is the transmission time of a message between two Unix processes communicating via local sockets. This time relates to the communications between

the Process Controller and its domain processes, as well as between domain processes, business processes and their child business processes (see Figure 52).

- t_{CBS} is the transmission time of a message between two software applications communicating via the CIM-BIOSYS infrastructure. This is typically the transmission time of messages between the four components of the business entity, enterprise activity occurrences and active resource components (see Figure 52). t_{CBS} is a measure of how efficient the CIM-BIOSYS infrastructure is in transferring messages between its applications.

- pr_{DPi} is the number of procedural rules executed by the domain process DPi at run time. The same remarks made for nEF_{DPi} with regard to the number of procedural rules that are actually executed at run-time apply to pr_{DPi}. Likewise, the same definition also applies to pr_{BPi}.

- t_{PR} is the average time that a domain process or a business process occurrence takes to execute one of its procedural rules. This includes the time to process the ending-status received from its function, in order to select the next function to be triggered.

- k_{DPi} is a coefficient similar to k, which accounts for occurrences of functions used by DPi which are executed in parallel. A similar coefficient is also defined for business processes (i.e. k_{BPi}).

- Σ_{DPi} (t_{EFi}) is the sum of the execution times of all functions used by DPi, where t_{EFi}, is the time to execute a complete occurrence of the enterprise function EFi. This can be the time to execute either a business process (i.e. t_{BPi}) or an enterprise activity (i.e. t_{EAi}) belonging to DPi. t_{EFi} is given by:

$$t_{BPi} = t_{SP} + (2 + 4.pr_{BPi} + nEF_{BPi}).t_S + 2.nEF_{BPi}.t_{CBS} + pr_{BPi}.t_{PR} + k.\Sigma_{BPi}\ (t_{EFi}) \quad (4)$$

or

$$t_{EAi} = 2.t_S + t_{aloc} + t_{link} + k_{EAi}.(ntr_{EAi} + \Sigma_{ARCj}\ (ntr_j)).t_{tr} + (3.nFO_{EAi} + 3 + nEV_{EAi}).t_{CBS} + \Sigma_{EAi}\ (TM_i) \quad (5)$$

where,

- t_{aloc} is the time that the Resource Manager can take to bind an active resource component to an enterprise activity occurrence. This time can vary considerably, for it depends on the availability of active resource components to serve the enterprise activity occurrence. In a situation where active resource components are readily available, $t_{aloc} \sim 0$.

- t_{link} accounts for the time taken by the CIM-BIOSYS infrastructure to start an application and establish a link (i.e. a connection) between that application and any other applications wishing to communicate with it (i.e. the application protocol of the CIM-BIOSYS infrastructure is connection oriented). This time is only relevant for enterprise activity occurrences for their occurrences are represented by CIM-BIOSYS applications activated dynamically at run-time.

- $\mathbf{ntr_{EAi}}$ is the number of relevant transitions contained in a particular enterprise activity which are required to describe its behaviour (see Figure 31). This number includes only the transitions that are actually executed by a enterprise activity occurrence at run-time. Thus, the same remarks made with regard with the procedural rules for process apply to transitions for enterprise activities.

- Σ_{ARCj} $(\mathbf{ntr_j})$ is the sum of the relevant transitions executed by the active resource components (i.e. $\mathbf{ntr_j}$) with which the enterprise activity interacts, in order to perform its functionality. The same considerations made for $\mathbf{ntr_{EAi}}$ apply to this factor.

- $\mathbf{k_{EAi}}$ is a coefficient similar to **k**, which accounts for any parallelism in the execution of the transitions of an enterprise activity occurrence and its associated active resource components.

- $\mathbf{nFO_{EAi}}$ is the number of functional operations exchanged between the enterprise activity, its active resource components and the information entity (see Figure 30). As the models developed in this thesis concentrated on the functional and resource aspects, functional operations related to the information entity are not considered.

- $\mathbf{nEV_{EAi}}$ is the number of events generated by the enterprise activity at run-time. In the case of the SMT assembly line models nEV_{EA1} = BS (i.e. within EA-1, one occurrence of EV-3/Produce PCB is generated for each PCB in a batch of BS boards).

- t_{tr} is the elapsed time between two consecutive firings of transitions within an enterprise activity occurrence or active resource component. This is a design parameter whose minimum value is limited by the speed of the Prolog engine used to construct these components.

- Σ_{EAi} (TM_i) is the sum of the manufacturing process time intervals (i.e. TM_i) associated with the functional operations (see Table A-5). Here, it should noticed that:

$$LT_s = k_{model} \cdot \Sigma_{model} [\Sigma_{EAi} (TM_i)]$$

where,

- Σ_{model} is the sum of the manufacturing process times associated with functional operations used by the enterprise activities of a model; and
- k_{model} is, once again, a coefficient used similarly to **k**, in order to account for operations in the manufacturing processes which are executed in parallel.

then,

$$OH_s = LT_{rp} - LT_s \quad (6)$$

where LT_{rp} is given by equation (2), which aggregates equations (3), (4) and (5).

Even though the equations that lead to the determination of OH_s may seem quite complex, in fact, they constitute a simplification of the universe of factors which contribute to overhead. Some of the factors which have not been included in these relations are:

- the internal processing of the business entity components (e.g. the time for the Process Controller to respond to an event occurrence sent by the Event Handler - see Figure 52);
- the processing time related to the debugging functions of the business entity (e.g. data update on interfaces such as the one shown in Figure 48 and generation of log files).

Based on these considerations, the deducted equations were applied to the model of the SMT assembly line, which resulted in the following equations (these equations only compute overhead, i.e. they do not include the factor TM_i):

$$t_{EA1} = 2.t_S + t_{aloc} + t_{link} + 6.t_{tr} + 10.t_{CBS} + \Sigma_{EA1}\,(TM_1)$$

$$t_{EA2} = 2.t_S + t_{aloc} + t_{link} + 6.t_{tr} + 9.t_{CBS} + \Sigma_{EA2}\,(TM_2)$$

$$t_{EA3} = t_{EA4} = t_{EA5} = t_{EA6} = t_{EA7} = t_{EA8} = t_{EA9} = t_{EA2}$$

$$t_{BP1} = t_{SP} + 20.t_S + 22.t_{CBS} + 3.t_{PR} + 2.t_{aloc} + 2.t_{link} + 12.t_{tr}$$

$$t_{BP2} = t_{SP} + 27.t_S + 33.t_{CBS} + 4.t_{PR} + 3.t_{aloc} + 3.t_{link} + 18.t_{tr}$$

$$t_{BP3} = t_{SP} + 20.t_S + 23.t_{CBS} + 3.t_{PR} + 2.t_{aloc} + 2.t_{link} + 12.t_{tr}$$

$$t_{DP1} = t_{SP} + 13.t_S + 12.t_{CBS} + 2.t_{PR} + t_{aloc} + t_{link} + 6.t_{tr}$$

$$t_{DP2} = 4.t_{SP} + 83.t_S + 84.t_{CBS} + 14.t_{PR} + 7.t_{aloc} + 7.t_{link} + 42.t_{tr}$$

where OHs is given by:

$$OH_s = k.\ \Sigma_i\ (noc_{DPi}.\ t_{DPi})$$

$$OH_s = 5.t_{SP} + 101.t_S + 96.t_{CBS} + 16.t_{PR} + 8.t_{aloc} + 8.t_{link} + 48.t_{tr}$$

These equations were obtained for the following conditions:

- $k_{model} = k = k_{DPi} = k_{DPi} = k_{DPi} = 1$. That is, no parallelism is assumed to occur for the execution of a model occurrence. This is the case for BS = 1 and 0% inspection and check.
- nFO = 2 (i.e. two functional operations per enterprise activity);
- $nEV_{EA1} = BS = 1$;
- $ntr_{EA1} = BS + 2 = 3$, $ntr_{EAj} = 3$ (for j = 2 to 10);
- $ntr_{ARCj} = 3$ for all active resource components;
- $pr_{BP1} = 3$, $pr_{BP2} = 4$, $pr_{BP3} = 3$, $nEF_{BP1} = 2$, $nEF_{BP2} = 3$ and $nEF_{BP3} = 2$. These values stem from a condition in which 0% of the boards produced are either checked of inspected. This condition was selected for two reasons: (1) they enable the establishment of deterministic values to the parameters in question; and (2) they are nearest to value of lead-time obtained for the default operating conditions of the SMT assembly line.
- $pr_{DP1} = 2$, $pr_{DP2} = 4$, $nEF_{DP1} = 1$ and $nEF_{DP2} = 3$.

Appendix 12: List of Publications

A-12.1.Journal Publications

Workbench for rapid prototyping of open software systems. Submitted to the *International Journal of Manufacturing Systems Design*. England. 1994.

A Model-driven approach to enterprise integration. Submitted to the *International Journal of CIM*. England. 1993.

CIM-OSA and stochastic time petri nets for behavioural modelling and model handling in CIM systems design and building. Published in the *Proceedings of the IMechE Journal*. England. 1993 (Winner of the 1993 Donald Julius Groen Prize)

A-12.2.Conference Publications

'Model-Driven CIM' - Frameworks, Methods and Architectures to Support the CIM System Life Cycle. Published in the *Proceedings of the 2nd Workshop on CIM-OSA*. Germany. 1994.

Model Enactment as a Basis for Rapid Prototyping of Manufacturing Systems. Submitted to the *Proceedings of the First IMSE*. France. 1994.

Manufacturing Systems Design and Implementation Based on Formal Modelling methods and Tools. Published in the *Proceedings of the 27th International Symposium on Automotive Technology and Automation (ISATA)*. Germany. 1994.

Rapid prototyping of open software systems. Published in the *Proceedings of the European Simulation Multi-conference*. Spain. 1994.

Petri-nets in SEW-OSA - Systems Engineering Workbench centred on CIM-OSA. Published in the *Proceedings of the IEEE Conference on Systems, Management and Cybernetics*. USA. 1994.

The Business Entity of SEW-OSA - Systems Engineering Workbench for CIM-OSA. Published in the *Proceedings of the 10th National Conference on Manufacturing Research*. England. 1994.

Systems Engineering Workbench centred on CIM-OSA. Submitted to the *Proceedings of the 10th International Conference on Computer Aided Manufacturing, Robotics and Factories of the Future CARS-FOF/94*. Canada. 1994.

Model based approach supporting the life cycle of integrated manufacturing enterprises. Published in the *Proceedings of the International Conference on Computer Integrated Manufacturing ICCIM*. Singapore. 1993.

Reference architectures for enterprise integration. Published in the *Proceedings of the International Conference on Computer Aided Manufacturing, Robotics and Factories of the Future CARS-FOF/93*. USA. 1993.

CIM-OSA and time petri nets for CIM systems modelling and simulation.

Published in the *Proceedings of the International Conference on Factory Automation and Integrated Manufacturing - FAIM/93*. Ireland. 1993.

Case study on the application of the CIM-OSA methodology for manufacturing process modelling. *Published in the Proceedings of the European Simulation Symposium*. Germany. 1992.

A-12.3.Internal Reports

A proposal for linking SEW-OSA with a bottom-up modelling tool. Document n. 9 v. 1.0. 1995.

Preliminary review of methods and techniques to support business strategy development - internal report. England. 1994.

Notes from the case study on the application of SEW-OSA to D2D shop-floor. Document n. 8 - v. 1.0. 1994.

SEW-OSA (Systems Engineering Workbench centred on CIM-OSA) - CASE Tool documentation. Document n. 7 - v. 1.0. 1994

SEW-OSA (Systems Engineering Workbench centred on CIM-OSA) Business Entity - Printout of the software code. Document n. 6 - v. 1.01994.

Collection of five reports to CAPES on the progress of the PhD research work - isseud between 1991 and 1994. 1994.

Petri-net Case Tool Documentation. Document n. 5 - v. 1.0. October 1992.

Study on Reference Architectures for CIM Systems Design and Building Based on Executable Models. Master of Philosophy Report - Loughborough University of Technology. 1992.

Data Gathering on the SMD Assembly Line - Third Interview - Working Group 3 - ICL/LUT-SIG Joint Work On CIM-OSA. Document n. 4 - v. 1.0. May 1992.

Data Collection on the SMD Assembly Line - Working Group 3 - ICL/LUT-SIG Joint Work On CIM-OSA. Document n. 3 - v. 1.0. March 1992.

First Exercise on the Application of the CIM-OSA Modelling Methodology - Working Group 3 - ICL/LUT-SIG Joint Work On CIM-OSA. Document n. 2 - v. 1.0. January 1992.

On the Life Cycle of Integrated Manufacturing Systems - Research Plan Outline. Document n. 1 - v. 1.1. November 1991.