Evaluation of architecture design with CIMOSA

Citation for published version (APA):

Zwegers, A. J. R., Fang, S. G., & Pels, H. J. (1997). Evaluation of architecture design with CIMOSA. *Computers in Industry*, 34(2), 187-200. <https://doi.org/10.1016/S0166-3615%2897%2900054-7>, [https://doi.org/10.1016/S0166-3615\(97\)00054-7](https://doi.org/10.1016/S0166-3615(97)00054-7)

DOI:

[10.1016/S0166-3615%2897%2900054-7](https://doi.org/10.1016/S0166-3615%2897%2900054-7)
[10.1016/S0166-3615\(97\)00054-7](https://doi.org/10.1016/S0166-3615(97)00054-7)

Document status and date:

Published: 01/01/1997

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Evaluation of architecture design with CIMOSA

Arian J.R. Zwegers^{a,*}, Shu-Guei Fang^b, Henk-Jan Pels^a

^a Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands

^b Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK), Pascalstraße 8-9, D-10587 Berlin, Germany

Abstract

This paper evaluates the suitability of the CIMOSA modelling framework for the specification of control architectures for manufacturing systems. An architecture can be seen as a set of specifications that express the functions of components and their interfaces. Based on the characteristics of architectures, several requirements on architecture modelling techniques are defined. These requirements are used in the evaluation, which is illustrated by an industrial application. CIMOSA offers adequate constructs to specify concurrent processes and their interactions, and to specify process behaviour. The framework separates architectural concerns from implementation matters and provides multiple views. Dynamic constraints, which are used to non-deterministically specify system behaviour, can not be represented. In addition, the semantics of the formal model reveal some inconsistencies. The CIMOSA modelling framework is now accompanied by some methodologies that aim to guide the user in the application of the framework. © 1997 Elsevier Science B.V.

Keywords: Control architecture; Architectural design; CIMOSA

1. Introduction

The purpose of an architecture is to structure a complex system. It largely determines the effectiveness as well as the flexibility of a system. Therefore, the design and analysis of an existing or perceived architecture is essential for system operation. Currently, a number of modelling frameworks is available that aim to provide the necessary formalisms for sound design of architectures.

The objective of this paper is to evaluate the suitability of the CIMOSA architectural framework for the specification of control architectures for manufacturing systems. CIMOSA — computer integrated manufacturing open system architecture —

consists of a modelling framework and an integrating infrastructure [1,2]. The latter is used for model execution support, and is not considered in this paper. CIMOSA aims to support system designers with descriptive modelling of enterprise operation which allows the identification of the needs for operational information as well as the information produced during the operation itself. It is exactly this kind of modelling that is used for defining a control architecture.

The architectural framework CIMOSA has inspired many researchers e.g., [3–7]. An overview of CIMOSA applications is given in [8]. Research on the modelling framework includes an extension with the economic view by Neuscheler and Spath [9], and a critical review of the theoretical aspects by Kotsiopoulos [10]. The European Committee for Standardisation (CEN) has made an evaluation of

* Corresponding author. Tel.: +31-40-2472671; fax: +31-40-2436492; e-mail: azw@tm.tue.nl.

CIMOSA's constructs for modelling enterprise operational processes with the ultimate goal of monitoring and controlling the daily enterprise operation [11]. Williams et al. investigated CIMOSA and two other enterprise reference architectures [12]. However, none of these researchers has validated the CIMOSA modelling framework for architectural design.

To illustrate the framework's suitability, we use its application in a re-engineering process at Traub, a German machine tool manufacturer. For more information on this re-engineering process, the reader is referred to [13]. Results were obtained by modelling, document analysis, and literature survey.

This paper is organised as follows: In Section 2, the types of architectures are described, and the characteristics of (control) architectures are given. After this, we present the requirements for architecture modelling formalisms. Then, the CIMOSA modelling framework is shortly introduced. Furthermore, we illustrate its application in the Traub case, and we make an evaluation of the framework's suitability for the specification of architectures on the basis of the defined requirements. We conclude this paper with a discussion on our findings.

2. Architectures

2.1. Types of architectures

Although the term 'architecture' is widely used in publications, it is generally assumed that the reader knows what this term means [14]. However, a lot of confusion arises since the word is used for two different meanings which deal with the integration of manufacturing entities or enterprises, namely:

- The structural arrangement (design) of a physical system such as the computer control system part of an overall enterprise integration system (Type 1).
- The structural arrangement (organisation) of the development and implementation of a project or program such as a manufacturing or enterprise integration or other enterprise development program (Type 2) [15,16].

Type 1, CIM architecture, is any method (drawing, model, description, etc.) for giving the structure

or framework showing the interrelationship of all of the parts and/or functions of a device, system or enterprise. A CIM architecture comprises those aspects involved in the computer-based information technology of the enterprise concerned. A control architecture is that part of the CIM architecture that is concerned with the enterprise's production control system.

Type 2, Enterprise reference architecture, is a collection of the overall generic functions, descriptions, or behaviours of many types of systems and their associated structures or frameworks. It presents a life-cycle development methodology for all aspects of the enterprise concerned.

In this paper, we reserve the term 'architecture' for the first type. We consider an architecture of a system or product as the result of a design process. It is a set of specifications that express the functions of components and their interfaces. The abstraction level of the specifications is higher than that of the specifications that are the result of later design activities. Drafting an architecture is concerned with the specification of components, their interfaces, their interactions, and constraints. This specification serves as a basis for further design and implementation activities, and should satisfy the imposed requirements. Thereby, architectural design succeeds the definition of requirements, and precedes the detailed design of components.

2.2. Characteristics

Architectural design consists of the specification of logical components and their interfaces. The architecture specifies functions that are assigned to the components, allowed inputs and outputs, and the logical relations between components. Furthermore, an architecture defines constraints on the functions and their interactions.

For instance, an architecture might specify a component 'fine planning' that plans jobs, and which operates in parallel with other components. The input for this component might for instance be a set of jobs and information about the status of executing resources; the output might be a list of production resources with allocated jobs. Such a list might be part of the interface between two components. For example, the set of jobs that are allocated to an

executing function, might be the output of a function Fine Planning and part of the input of the function that controls that specific executing element. The interaction between these two functions is the one-way exchange of jobs and — possibly — the feedback of monitoring information. A rather obvious example of a constraint is that the set of jobs allocated to production resources must be similar to the set of jobs that is input to the fine planning component. The following is an example of a dynamic constraint, which originates from a desire to fully utilise a specific executing element, say M :

For each system state i must hold that, if in state i jobs are available that might be executed on M , and if M is idle, it is not allowed that in state $i + 1$ resource M is still idle. This constraint specifies the priority of resource M over other resources [17].

The design of the solutions that embody these constraints should not be part of the architecture design.

An architecture can be looked at from different points of view. Apart from looking at a control architecture from a functional viewpoint, a designer has to take into account other aspects as well. For instance, a designer wants to be able to concentrate on the structure of information objects — such as the jobs in the previous example — and the relationships between them. It follows that several aspects of architectures are relevant to the overall design task, and that a designer has to construct a number of models. Each of these models omits those aspects of the system under study which are considered not to be relevant for the present purposes, and tends to concentrate on one single aspect of the architecture in question (i.e., aspect models). The use of a set of models allows the designer to deal with a number of relations between the essential elements of the system under construction.

3. Requirements on modelling techniques

Based on literature and on our own experience, we have identified a number of requirements a modelling technique has to fulfil in order to support a designer in the specification of control architectures [18,19]. In Section 5, these requirements are illustrated and explained in more details. Note that the following set is not pretended to be exhaustive.

The main requirement is that the technique should be able to appropriately express the architectural characteristics of a system as described in the previous section. This means that the technique has to provide the ability to specify concurrent processes, the interactions between these processes in terms of exchange of messages and products, and the process behaviour. In addition, constraints on system states (static constraints) as well as constraints on changes of states (dynamic constraints) have to be specified. The technique should provide multiple views on an architecture, and modelling should result in architectural specifications that are independent of implementation concerns. The technique should be based on a formal model to unambiguously specify the architecture under design, and it should be supported by a software tool. Finally, an accompanying methodology should guide a user in the modelling process.

4. CIMOSA modelling framework

In Fig. 1, the modelling framework is represented by the CIMOSA cube. The cube offers the ability to model different aspects and views of an enterprise [1,2]. This three-dimensional framework has a dimension of genericity, a dimension of enterprise models, and a dimension of views:

- The genericity dimension is concerned with the degree of particularisation. It goes from generic building blocks to their aggregation into a model of a specific enterprise domain
- The modelling dimension provides the modelling support for the system life cycle, starting from

Fig. 1. CIMOSA modelling framework.

statements of requirements to a description of the system implementation.

- The view dimension offers the possibility to work with sub-models representing different aspects of the enterprise.

For more details, the reader is referred to the appendix.

5. Design of a control architecture with CIMOSA

In this section, we indicate how CIMOSA fulfils the requirements we defined for an architecture modelling technique. We illustrate its suitability by means of examples from the application of the modelling framework in a re-engineering process at Traub.

Traub, a machine tool manufacturer located in Germany, wanted to optimise their order planning, tool management, and order execution processes. An important requirement is to have its machine tools available for production in time. Changing order information by the customer makes it frequently necessary to reschedule manufacturing jobs. Often problems arose because manufacturing devices were not available. One of the main reasons was that the tools, necessary to fulfil the jobs, were not prepared.

To solve this problem, a tool logistic system had to be implemented, which caused some changes in the production control system [13].

Traub's control elements can be classified into several types according to their control tasks. General control elements are area control, tool management, cell control, and machine handling. Area control is responsible for dispatching orders to tool management and cell control. Tool management sets up tools and consigns tool data for each order. The tasks of cell control are order fine planning and process monitoring. Machine handling is used to directly control the NC machines and to collect machine data for process monitoring.

Traub's new, hierarchical control architecture was modelled with CIMOSA's requirements definition level and design specification level. The principal area and cell control functions are captured in CIMOSA *domains*, the constructs that embrace the main enterprise processes. Fig. 2 shows the identified domains and their relations in the form of *domain relationships*. Non-CIMOSA domains are parts of the enterprise which have not been considered for the moment and which will be detailed in the future such as 'purchase' and 'DNC node', or which are closed legacy applications that can not be described such as the 'area control' application.

Fig. 2. Identified domains and their relations.

5.1. Concurrent processes and their interactions

Control systems are usually distributed systems, consisting of a set of interacting processes. These processes are under separate control and concurrently operational, but they interact with each other by the exchange of messages and products. By interaction, the processes affect each other's behaviour; they cooperate to realise the goal of the complete system.

CIMOSA represents main enterprise functionalities as *domain processes*. By means of the domain process construct, CIMOSA offers a modular approach for architecture modelling; the system can be easily extended with new domain processes, and system modifications can be limited to few domain processes. Enterprise operation is structured into a set of interoperating domain processes exchanging results and requests. These stand-alone processes are triggered by events and produce defined end results. They encapsulate a set of enterprise functionality and behaviour to realise certain business objectives under given constraints. In the Traub case, examples of concurrent processes are the processing of tool data

and the preparation of tools. These processes are represented in domain 'tool management' by two independent domain processes, namely 'tool data processing' and 'prepare tools'. Fig. 3 is a partial refinement of the previous figure, showing domains and domain processes. Note that the domain processes in Fig. 3 influence each other's behaviour.

For the exchange of messages and products, CIMOSA offers the *event* and *object view* constructs. Object views, which might be attached to events, have an informational or physical nature. In the Traub model, events that are sent and received by domain processes are either requests or, with attached object view, information reports. An occurrence of a domain process uses requests to trigger the execution of occurrences of other domain processes and uses information reports to inform other domain processes about status changes.

In the hierarchical control architecture of Traub, requests are only sent by control elements at a certain level to control elements at lower levels. Conversely, information reports are sent from a control element at a certain level to control elements at a higher level. In Fig. 3, the arrows that point down-

Fig. 3. Partial refinement of Traub's domains.

wards are requests; the arrows that point upwards are information reports.

5.2. Process behaviour

In addition to capabilities to represent processes, a modelling technique needs to be able to describe the internal behaviour of these processes. To realise the overall system's goal, processes cooperate and influence each other's behaviour by the exchange of messages and products. The description of this process behaviour involves the decomposition of processes and the identification of subprocesses. By functional decomposition, overall system complexity is suppressed and system design is made more manageable.

For the specification of process behaviour, CIMOSA offers the *business process* and *enterprise activity* constructs. In the previous section, it is stated that CIMOSA structures the enterprise operation into a set of interoperating domain processes which exchange results and requests. Domain processes are the root of the decomposition tree; they

employ *business processes* which are in the middle of the decomposition tree. The leaves are named *enterprise activities* and are employed by business processes or, if there are no business processes, by domain processes. The behaviour of a certain business or domain process is defined by rules, according to which enterprise activities belonging to this process are carried out. Enterprise activities represent the enterprise functionality as elementary tasks, and they are defined by their inputs, their outputs, their function and their required capabilities. Comparable to the messages and products of domain processes, the inputs and outputs of enterprise activities are described by means of object views.

Fig. 4 presents the behaviour of domain process 'order processing'. For clarity some relationships have been deleted. Events, which are either received from or sent to other domain processes, are represented by a Z-shape; enterprise activities are shown as boxes labelled 'EA'. Note that there are no business processes specified. The large triangles indicate behavioural rules according to which enterprise activi-

Fig. 4. Process behaviour of domain process 'order processing' (partially).

ties or business processes are carried out. Fig. 4 was made by the GtVOICE tool (see Section 5.7).

5.3. Constraints

Constraints are needed to non-deterministically specify process behaviour. In the previous sections, it is stated that control systems are usually distributed systems, consisting of a set of interacting processes. It must be possible to specify the behaviours of the processes non-deterministically. Intuitively speaking, this means that process behaviour may be specified only partially. Some constraints are specified and any actual behaviour is permitted as long as these constraints are satisfied. A designer wants to express only those constraints in the architecture that are necessary at the architectural level of the system specification. In this way, the greatest possible freedom can be left for the detailed design and implementation to optimise the design against all kinds of (situation-dependent) costs. This is a departure from current practice that, instead of specifying the constraints, supplies specific solutions that embody those desired constraints [18,19].

For instance, a constraint for which Traub specified a solution, is related to the domain 'NC machine handling' (see Fig. 3). Domain process 'manufacturing' may not be executed for a specific NC machine, while at the same time this machine is being set up by the execution of domain process 'set up NC machine'.

CIMOSA does not provide adequate constructs for sufficient specification of constraints. It does offer some constructs that aim to specify constraints, such as the *constraint* construct, which describes limits imposed on functional elements at a qualitative or quantitative way. *Declarative rules* define imposed business rules as formal combinations of objectives and constraints together with the conditions under which they apply. *Integrity rules* express user-defined constraints on *information elements* i.e., elementary, atomic pieces of information, to ensure their consistency. Declarative rules and integrity rules are constructs at the requirements definition level (see Fig. 1). At the design specification level, they are transformed into *integrity constraints*, which are data-oriented representations of specified rules concerning information items. Some additions have been

made to the CIMOSA modelling language that was used by Traub (for instance, see [20]). However, none of these constructs are capable of formally specifying some types of constraints. Especially dynamic constraints, where time aspects are involved, can not be formally specified.

5.4. Multiple views

Multiple views of an architecture are required in order to deal with several relations between elements of the system under construction. Different aspects of the architecture need to be described in an appropriate view. Consider the analogy of a building architect: a building architect works with the customer and the builder by means of a number of different views in which some particular aspect of the building is emphasised. The architect may provide the customer with scale models with the look of the building in its context. For the builder, the architect provides structural views that provide an immense amount of detail about various explicit design considerations such as electrical wiring, plumbing, heating, etc.

Similarly, a manufacturing system architect needs a number of different views of the control architecture for the various uses and users. While some particular aspects are emphasised, other detail is suppressed to avoid obscuring the real issues at stake. Two viewpoints that an architect definitely needs to consider are the functional viewpoint and the information viewpoint. The function view regards processes and their interactions, whereas the information view displays the objects that are exchanged between processes.

The CIMOSA modelling framework provides views, which it considers as 'windows' through which selective aspects of an enterprise can be observed and manipulated. The set of views is open, and extensions have been suggested. For example, Neuscheler and Spath proposed the addition of an economic view [9]. The original framework, as shown by Fig. 1, contains four views:

- Function view: allows observation of the enterprise behaviour and functionality for operation planning, control and monitoring.
- Information view: allows observation of the structure of business information that is used during

enterprise operations for planning, control and decision-making processes.

- Resource view: allows observation of the enterprise's assets needed for carrying out the enterprise processes, including the use of the model to manage (control and monitor) these assets.
- Organisation view: allows observation of the decision making responsibilities in the enterprise for function, information, resources and for the management of exceptions and decision-making [2].

During the model creation process in the Traub case, the function and information view were extensively used, whereas the resource and organisation view were barely addressed. It was not meaningful for Traub to consider too many factors such as responsibility before Traub was satisfied with the new control structure regarding functionality and information flow. An example of a construct of the information view is the *conceptual schema*. The conceptual schema is linked with constructs of the function view by means of the *external schema* construct. External schemata describe how the information used by an enterprise is presented to a particular user, and therefore specifies how an object view can be implemented [1]. Fig. 5 shows a part of Traub's specification of the conceptual schema.

5.5. Independence of implementation concerns

The modelling technique must be independent of methods of implementation so that the resulting architectural specifications do not unduly constrain implementers. A common system life cycle consists of the following phases: determination of objectives, requirements definition, architectural design, detailed design, implementation, and operation. The set of specifications that come out of the architectural design phase should fulfil the requirements, and serve as the foundation from which the desired system can be built. On the other hand, practice shows that designers, while defining the architecture, at the same time make some implementation choices. Nevertheless, architectural specifications constrain later design choices; these constraints should be imposed by the architect, and not by the modelling technique.

CIMOSA, a descriptive rather than a prescriptive framework, clearly separates implementation choices from architectural specifications. Fig. 1 shows that the cube covers the requirements definition, the design specification, and the implementation description phases of a system life cycle. The requirements definition level of the CIMOSA modelling framework is used at architectural design activities and not

Fig. 5. Conceptual schema (partially).

for the definition of requirements. The reason is that CIMOSA does not support a 'true' definition of requirements. Instead, the CIMOSA requirements definition level offers support in structuring a manufacturing system at a high level i.e., in defining the system architecture.

In the Traub case, the specification of both the architecture and the physical system went hand in hand. When Traub initially defined its control architecture as it is reflected by Fig. 3, it specified that the area control function had to be implemented on a workstation. In addition, personal computers were chosen for the cell control and the tool management function. Finally, communication between the area controller and the cell controller would be done with Ethernet, using the MMS communication standard. At an early stage in the system life cycle, Traub made some implementation choices. These choices were not imposed by CIMOSA, that postpones such decisions to the implementation description level, although they can be recorded as declarative rules in the requirements definition model.

5.6. Underlying formal model

The modelling technique should be based on a formal model suitable for proving the correctness and equivalence of architectural specifications. An architecture is an abstraction of a real (concrete) system, and may be specified by means of informal descriptions, graphical models, a mathematical model, etc. From all description methods, mathematics is recognised as the most appropriate one, because it permits the system under design to be specified unambiguously. In addition, the formal model should support the verification of whether a certain implementation conforms to an architectural specification [19].

In essence, the CIMOSA modelling framework is based on a formal underlying model. However, some inconsistencies appear in CIMOSA's formal specification as it is described in [1]. Kotsiopoulos gives some theoretical shortcomings of the framework, for example the missing capability of parent processes to re-assign the ending status values of processes and activities during model execution [10]. Furthermore, he points out that a consistent language specification and a verification method are not available. These

deficiencies hamper the fulfilment of CIMOSA's ultimate goal, namely the monitoring and control of the daily enterprise operation by model execution. Nevertheless, the functional, structural, and behavioural aspects of all constructs except the constraints are sufficiently specified for architectural design.

5.7. Tool support

A modelling technique should be supported by a software tool. During the model creation process, problems might occur regarding the size and complexity of the architectural models. Models become too big to be overlooked by the user, and they become inconsistent. Furthermore, the modelling process without tool support is tardy, and maintainability and extendibility of the models are jeopardised.

Some tools have been developed for modelling with CIMOSA, such as 'GtVOICE', which is described in [21]. This tool ensures model consistency and reduces modelling time by easy model modification and extension. In addition, it provides non-ambiguous interpretation of CIMOSA constructs and a uniform way to present models among CIMOSA users. Finally, GtVOICE makes communication with other tools feasible, such as an SQL data server and a rapid prototyping tool kit.

5.8. Methodology support

Together with a suitable framework for architectural design, guidelines for the model creation process should be available. A designer must be guided to navigate through the modelling framework in a consistent and optimised path, in order to ensure complete, consistent and optimal models.

During the modelling process for the Traub application, no modelling guidelines were available. However, at this moment CIMOSA provides a methodology to guide users in application of its modelling framework. Zelm et al. describe this so-called CIMOSA business modelling process [22]. For modelling at the requirements definition level, the top-down modelling process starts with the identification of domains, the definition of domain objectives and constraints, and the identification of rela-

tionships between the domains. Afterwards, the domains are decomposed into domain processes, which are further decomposed in business processes and enterprise activities. The modelling process at requirements definition level ends with some analyses and consistency checking. Most existing tools that support the design of CIMOSA models, such as GtVOICE [21] and SEW-OSA [3], adopt this modelling approach.

Recently, new methodologies were proposed that aim to be improvements or alternatives to the CIMOSA business modelling process. For instance, Reithofer proposes a bottom-up modelling methodology for the design of CIMOSA models [23]. He claims that the CIMOSA business modelling process can hardly be focused on processes and activities that are relevant to solve a concrete problem. His bottom-up modelling approach should not have this disadvantage. In addition, Janusz asserts that existing CIMOSA models of particular enterprises are not complete, not consistent and not optimal [24]. Also, these models often describe only functions or sub-processes limited by department borders of an enterprise. Therefore, she developed an algorithm that filters out process chains of an existing CIMOSA model. Using the algorithm, the completeness and the consistency of the considered process chains can be checked.

6. Discussion

Most characteristics of control architectures can be specified by means of the CIMOSA modelling framework. Domain processes, events, and object views are adequate constructs to specify concurrent processes and the exchange of information and products between these processes. By defining domain processes and by the establishment of relations between them, any type of control architecture may be modelled. The modular nature of domain processes makes CIMOSA models extensible and modifiable. Moreover, the framework separates architectural choices from implementation concerns, and provides an open set which contains at the moment four views to focus on different enterprise aspects. Recently, some software tools have been developed that support CIMOSA modelling at the requirements definition and design specification level.

CIMOSA lacks the capability to adequately model a wide range of dynamic constraints. The ability to specify constraints is an essential characteristic of architecture modelling formalisms. Therefore, we believe that the current CIMOSA framework needs to be extended with constructs to model dynamic constraints. Furthermore, the semantics of CIMOSA's underlying formal model reveal some inconsistencies that hamper model simulation and execution.

Apart from the specification of constraints and the semantics of the CIMOSA meta-model, another research area is the modelling of human behaviour. Active human involvement is important for successfully dealing with unforeseen events and complex situations in daily operation. Even more, the integration of humans with the system is necessary to enable a 'closed loop' of continuous learning on the human side, and continuous improvement and adaptation on the system side. This topic is not addressed in this article, since it is a design issue rather than an architectural one.

Recently, some guidelines have been defined that navigate a designer through the modelling framework, in order to ensure complete, consistent and optimal models. However, the practical value of these rules for industry has not been established yet.

Perhaps even more important for industry than a suitable framework and accompanying guidelines for the model creation process are methodologies to support designers to make the right architectural choices i.e., choices that result in flexible, effective systems. Industry needs guidelines for the architectural design of systems in order to discard inadequate options early in the design process, and to enable the selection of the best options. Clearly, these methodologies are not present at the moment. Note that it is not an objective of CIMOSA to provide such a prescriptive methodology. The CIMOSA modelling framework aims to support system designers with descriptive modelling of enterprise operation; it is a descriptive rather than a prescriptive framework.

Acknowledgements

The authors would like to express their gratitude to the partners in ESPRIT project VOICE II (EP 6682), most notably to Traub, TPD-TNO, and FhG-

IPK. This study would not have been possible without them.

Appendix A

The aim of this appendix is to provide more information about modelling with CIMOSA. First, the three dimensions of the CIMOSA modelling framework are described. Then, the basic concepts of CIMOSA modelling are indicated. Note that this appendix can only give a global description of modelling with CIMOSA. For more details, the interested reader is referred to [1,2].

A.1. The CIMOSA modelling framework

In Fig. 6, the modelling framework is reprinted for convenience. The cube offers the ability to model different aspects and views of an enterprise. This three-dimensional framework has a dimension of genericity, a dimension of enterprise models, and a dimension of views.

The genericity dimension is concerned with the degree of particularisation. It goes from generic building blocks to their aggregation into a model of a specific enterprise domain.

The *generic level* consists of a collection of constructs, which are basic architectural building blocks that can be re-used in various architectural configurations. It includes Generic Building Blocks and Building Block Types for functions, objectives, constraints, services, and protocols. Constructs de-

scribed at this level have the widest application in CIM.

The *partial level* is populated with partial models. These reference models are incomplete skeletons of models for particular enterprises and are applicable to a wide range of industrial sectors, companies and/or manufacturing strategies. Partial models are the prime means by which CIMOSA encapsulates industry needs, and provide a realistic and usable tool for a particular enterprise.

The *particular level* is entirely concerned with one particular enterprise or part of it. The particular model embodies all necessary knowledge of the enterprise in a form that can be used directly for the specification of an integrated set of manufacturing technology and information technology components, such as machines, application programs, and human resources.

The dimension of enterprise modelling consists of three modelling levels. A modelling level is a level of abstraction corresponding to one of the main phases of enterprise model development.

The *requirements definition level* uses a simple language to identify the business requirements of the enterprise, reflecting the objectives of that enterprise. Business requirements are concerned with defining what has to be done to fulfil the objectives of the enterprise without considering the issues of the technology employed.

The *design specification level* uses a computer processable language to identify and quantify, in an implementation independent format, the technology required to perform the identified processes. It structures and optimises the processes according to the overall enterprise constraints and the selected technology. The resulting model can be evaluated by simulation techniques.

The *implementation description level* defines in a computer executable form the means of process execution by selecting actual vendor products to provide the information technology and manufacturing technology components.

The dimension of views is composed of four views, which are 'windows' through which selective aspects of an enterprise can be observed and manipulated. Whilst some particular aspects are emphasised, other extraneous details are suppressed to avoid obscuring the real issues at stake.

Fig. 6. CIMOSA modelling framework.

The *function view* allows observation of the enterprise behaviour and functionality.

The *Information view* allows observation of the structure of business information used during enterprise operations.

The *Resource view* allows observation of the enterprise's assets needed to execute the enterprise processes.

The *organisation view* allows observation of the decision making responsibilities in the enterprise for functions, information objects, resources and for the management of exceptions and decision-making.

A.2. Modelling with CIMOSA

The structuring of particular enterprise domains follows the framework of modelling levels and views. In the CIMOSA approach an enterprise is seen as a collection of inter-related but non-overlapping domains. A CIMOSA domain contains one or more domain processes as opposed to non-CIMOSA domains, which are used to describe the relationships the CIMOSA domains have with the world outside the universe of discourse. Fig. 7 shows a set of domain processes cooperating on shared objects (results and events). *Domain processes* provide the functionality needed to satisfy some subset of the

enterprise's business objectives. One of those processes, the domain process 'smelting' is detailed in Fig. 7.

The functionality and behaviour of an enterprise are decomposed hierarchically. Fig. 7 shows that domain processes are the root of the decomposition tree and that they employ *business processes* which are in the middle of the decomposition tree. The leaves are named *enterprise activities* and are employed by business processes or, if there are no business processes, by domain processes. Decomposition can be either into business processes representing lower level behaviour or enterprise activities representing functionality.

The rules according to which enterprise activities belonging to a certain business or domain process are carried out, determine the behaviour of the process. For instance, Fig. 7 shows the behaviour of the business process 'melting'. Enterprise activities define the enterprise functionality as elementary tasks defined by their inputs, their outputs, their function and their required capabilities. The inputs and outputs of enterprise activities (information, control, and resources) are described by means of the *object view* construct of the information view. Fig. 7 gives an example of an enterprise activity with its inputs and outputs. For system design, enterprise activities are further decomposed into functional operations which

Fig. 7. CIMOSA modelling.

relate functionality to enterprise resources for execution.

References

- [1] AMICE Consortium, CIMOSA: Open System Architecture for CIM — Technical Base Line, Version 2.0, February 1993.
- [2] AMICE Consortium, CIMOSA: Open System Architecture for CIM, Springer-Verlag, 1993.
- [3] M. Wilson C. Aguiar, I. Coutts, R.H. Weston, Model enactment as a basis for rapid prototyping of manufacturing systems, in: Proceedings of the European Workshop on Integrated Manufacturing Systems Engineering (IMSE '94), Grenoble, Dec. 1994, pp. 86–96.
- [4] M. Didic, F. Neuscheler, L. Bogdanowicz, McCIM: execution of CIMOSA models, in: C. Kooij, P.A. MacConaill, J. Bastos (Eds.), Proceedings of the 9th CIM Europe Annual Conference, Amsterdam, May, 1993.
- [5] M. Didic, CIMOSA model creation and execution for a casting process and a manufacturing cell, Computers in Industry 24 (2/3) (1994) 237–247, Special Issue on CIM Architectures.
- [6] B. Querenet, The CIMOSA integrating infrastructure, Computing and Control Engineering Journal 2 (1991).
- [7] R.H. Weston, J.M. Edwards, A model-driven toolset for flexibly integrating manufacturing systems, in: Proceedings of the European Workshop on Integrated Manufacturing Systems Engineering (IMSE '94), Grenoble, Dec. 1994, pp. 441–450.
- [8] VOICE Consortium, Computers in Industry 27 (2) (1995) Special Issue on Validation of CIMOSA.
- [9] F. Neuscheler, D. Spath, The economic view: a concept using benchmarks to analyze, evaluate and optimize business process, in: Proceedings of the European Workshop on Integrated Manufacturing Systems Engineering (IMSE '94), Grenoble, Dec. 1994, pp. 194–205.
- [10] I.L. Kotsiopoulos, Theoretical aspects of CIMOSA modelling, in: Proceedings of the 9th CIM Europe Annual Conference, Amsterdam, May, 1993.
- [11] CEN TC310 WG1, An evaluation of CIM modelling constructs: wvaluation report of constructs for views according to ENV 40 003, Computers in Industry 24 (2/3) (1994) 159–236, Special Issue on CIM Architectures.
- [12] T.J. Williams, P. Bernus, J. Brosvic, D. Chen, G. Doumings, L. Nemes, J.L. Nevins, B. Vallespir, J. Vlietstra, D. Zoetekouw, Architectures for integrating manufacturing activities and enterprises, Computers in Industry 24 (2/3) (1994) 111–139, Special Issue on CIM Architectures.
- [13] C. Schlotz, M. Röck, Reorganization of a production department according to the CIMOSA concepts, Computers in Industry 27 (2) (1995) 179–190, Special Issue on Validation of CIMOSA.
- [14] R.M.C. van Waes, Architectures for Information Management, Tinbergen Institute Research Series, No. 11, Thesis Publishers, Amsterdam, Netherlands, 1991.
- [15] T.J. Williams, J.P. Shewchuk, C.L. Moodie, The role of CIM architectures in flexible manufacturing systems, in: S.B. Joshi, J.S. Smith (Eds.), Computer Control of Flexible Manufacturing Systems, 1994, pp. 1–30.
- [16] P. Bernus, L. Nemes, T.J. Williams, Architectures for Enterprise Integration, Chapman and Hall, 1996.
- [17] P.J.M. Timmermans, Modular Design of Information Systems for Shop Floor Control, Eindhoven University of Technology, 1993.
- [18] Dewayne E. Perry, Alexander L. Wolf, Foundations for the study of software architecture, ACM SIGSOFT 17 (4) (1992).
- [19] Frank Biemans, Pieter Blonk, On the formal specification and verification of CIM architectures using LOTOS, Computers in Industry. 7 (1986) 491–504.
- [20] F.B. Vernadat, CIM business process and enterprise activity modelling, in: P. Bernus and L. Nemes (Eds.), Modelling and Methodologies for Enterprise Integration, 1996, pp. 171–182.
- [21] M.M. Didic, F. Couffin, E. Holler, S. Lampérière, F. Neuscheler, J. Rogier, M. de Vries, Open engineering and operational environment for CIMOSA, Computers in Industry 27 (2) (1995) 167–178, Special Issue on Validation of CIMOSA.
- [22] M. Zelm, F.B. Vernadat, K. Kosanke, The CIMOSA business modelling process, Computers in Industry 27 (2) (1995) 123–142, Special Issue on Validation of CIMOSA.
- [23] W. Reithofer, Bottom-up modelling with CIMOSA, in: Proceedings of the Second International Conference on the Design of Information Infrastructure Systems for Manufacturing (DIISM '96), Kaatsheuvel, Netherlands, Sept. 1996.
- [24] B. Janusz, Modelling and reorganizing of process chains using CIMOSA, in: Proceedings of the Second International Conference on the Design of Information Infrastructure Systems for Manufacturing (DIISM '96), Kaatsheuvel, Netherlands, Sept. 1996.

Henk-Jan Pels is an assistant professor in the Information Systems Group of the Department of Industrial Engineering at Eindhoven University of Technology. After he received his master's degree in electrical engineering at the same university in 1971, he joined Philips Industries, where he was engaged in the development of database applications for business as well as for the manufacturing area. In 1982 he moved to his current position, where he is researching and teaching databases and data modelling, especially for manufacturing applications. In 1988, he received his Ph.D. on a thesis on modular decomposition of data models. After that, he specialised in Architectures for Computer Integrated Manufacturing and Engineering Data Management. Since 1995, he is also part time consultant for M.I.S. Organisatie Adviseurs.

Arian Zwegers received his M.Sc. degree in Industrial Engineering and Management Science from the Eindhoven University of Technology in 1993. Currently, he is preparing his Ph.D. thesis at the same university. For two years, he worked for the VOICE II project in the Netherlands Organisation of Applied Scientific Research. His research interests include enterprise reference architectures and production control architectures.

Shu-Guei Fang is a research engineer in the Fraunhofer Institute for Production Systems and Design Technology (IPK) since 1991, where she is working on database systems, and where she is developing communication applications for process planning and simulation. Involved in the VOICE project since 1993, she was responsible for the application of CIMOSA modelling concepts for the industrial pilots and for the development of information services for distributed

manufacturing applications.