Events in CIMOSA and the CCE platform

S. Messina, P. Pleinevaux
Industrial Computing Laboratory, Department of Computer Science, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland
Tel: +41.21.693.67.97 Fax: +41.21.693.47.01, e-mail: silvia.messina@di.epfl.ch

Abstract

Event management is a mechanism useful for the specification of the system behaviour when specified conditions occur. The ability to represent reality in manufacturing systems may be enhanced by the availability of a well structured and powerful event management facility. We compare concepts like process, event, object, used to manage events, in the CIMOSA and CCE frameworks, two manufacturing modelling environments. We identify the relationship between such concepts. The two event management models are analysed in detail, they are compared and their differences are identified.

Keywords

Event management, manufacturing process modelling, integrating infrastructure, objects, information elements

1 INTRODUCTION

The aim of the present paper is to study the relationship between the CIMOSA and the CCE event models. We analyse the definition of their basic concepts, such as object, process and event in the CIMOSA framework and in the CCE platform. We compare their event management models and we identify the possible mappings between the two.

CIMOSA (Open System Architecture for CIM) (AMICE, 1994) provides a Reference Architecture for the modelling of a manufacturing enterprise. Its integration infrastructure offers a set of generic services for the execution of the enterprise model. The CCE (CIME Computing Environment) (CCE-CNMA, 1995) platform is an environment for the development, integration and execution of industrial applications. It has been developed as integration infrastructure of the CIMOSA platform. However, the two models have some differences and no exact mapping exists between their concepts. In the present paper we concentrate our attention on the event management models and their properties in CIMOSA and CCE.

Event management is a mechanism useful for the specification of the system behaviour under certain conditions. These conditions may be linked to physical devices, describing real events, or they may be triggered by user applications with the aim of synchronisation with or signalling to other applications. The ability to represent reality in manufacturing systems may be enhanced by the availability of a well structured and powerful event management facility. The need for such a facility in manufacturing environments is proved by the specification of event management mechanisms provided by existing industrial protocols (i.e., MMS (ISO/IEC, 1990)) and integrated manufacturing infrastructures (i.e., CIMOSA). CCE supports a limited event management.

The contribution of this paper consists in the comparison of concepts like process, event, object, used to manage events, in the two frameworks, CIMOSA and CCE, and the identification of the relationship between such concepts. The two event management models are analysed in detail, they are compared and their differences are identified.

Exceptions and exception handling should also be considered here for the sake of completeness, as extension of the present work, because of their commonalities with the event concept. Due to space limitation, we do not address here this issue and we recommend to the interested reader a preliminary study presented in (Messina, Pleinevaux, 1996).

The content of the paper is organised as follows: first an overview of the CIMOSA architecture and its event management is presented; the overview of the CCE platform and its event model follows, and a the relationship between CIMOSA and CCE is discussed. The second part of the paper presents the comparison and the mapping of the basic concepts of event management in CIMOSA and CCE: process, event, object, operation and attribute concepts are analysed in detail. The last section concludes the paper.

2 CIMOSA OVERVIEW

The CIMOSA Modelling Framework (AMICE, 1994) provides the necessary guidance to enable end users to model the enterprise and its associated CIM system in a coherent way. The CIMOSA modelling approach is based on a Reference Architecture composed of reusable generic building blocks, which are aggregated to describe the enterprise model.

The CIMOSA model development is composed of three phases, starting with the Requirements Definition Modelling phase (AMICE, 1991). This model is described by the end-user that provides his view of the business needs. He gives his knowledge about the function, information and resources of the system. The next phases are the Design Specification and the Implementation Description. The example given below concentrates on the Requirements Definition Level, and we develop our proposal referring to the Requirements Definition Model of the enterprise.

The first step in the Requirement Definition Model development consists in the definition of the Domain to be modelled, its objective and constraints.

The **Domain** describes a part of the enterprise relevant for achieving a defined set of business objectives. Examples of domains of activity in a real scenario are the Engineering Department or the Flexible Manufacturing System (Siemens, 1990), (Storr, et al., 1993).

Domains communicate among each other through **events** and describe the enterprise activities through Enterprise Objects and processes acting on them. An **Enterprise Object** is a generic entity of the enterprise that can be described by many Object Views. One Enterprise Object may be viewed from different points of view, thus it may correspond to several Object Views.

The functionality and the behaviour of a Domain is defined by **Domain Processes**. A Domain Process is a stand-alone process triggered by events and governing the execution of Enterprise Activities (the basic functionality) according to the so called Procedural Rules. Each Domain Process is decomposed into Business Processes and/or Enterprise Activities. That is the Domain Process is decomposed into hierarchically structured functions, that are elementary functions. A set of **Procedural Rules** define the sequence of activation of Business Processes and/or Enterprise Activities.

An Enterprise Activity is detailed by describing its functionality, composed of several components, some of which are:

- the Function Input (FI): set of Object Views to be processed and transformed by the activity;
- the Function Output (FO): set of Object Views produced or returned by the activity;
- the Resource Input (RI): is the possibly empty set of resources needed for the execution of the activity;
- the Resource Output (RO): textual statements indicating information to be recorded on the usage of the resources after the activity execution;
- the Control Input (CI): information used to control or constrain the activity execution;
- the Control Output (CO): set of Events generated by the activity;
- the Ending Status (ES): non-empty set of the possible termination statuses of the activity;
- the Activity Behaviour: finite algorithm specifying the functionality and behaviour of the activity; it is specified in terms of Functional Operations.

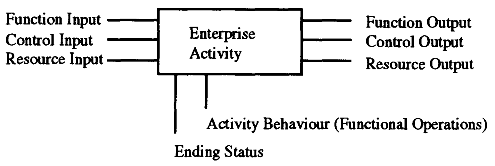

Figure 1 Enterprise Activity definition.

The functionality of an Enterprise Activity is further decomposed at Design Specification Modelling level into a set of Functional Operations to be executed by Human, Machine or Application resources. The concept of Enterprise Activity corresponds to the concept of process, that may eventually be distributed on several hosts and remotely executed.

CIMOSA event model

Events in CIMOSA may be generated by Enterprise Activities, resources and external components in order to trigger Domain Processes. The CIMOSA event model is further analysed and compared with the CCE event model in Section 5.

3 CCE OVERVIEW

CCE (CIME Computing Environment) is an *open* environment for development, integration and execution of industrial applications. Its aim is to simplify the task of integration of applications in heterogeneous environments (CEC, 1993). This platform hides to the users the diversity in communication protocols, databases and access methods.

The CCE consists of an intermediate software layer between the operating system and the end-user application, a so-called middleware, available on various hardware and software environments, providing a complete platform for the development, integration and operation of manufacturing applications. CCE is aimed at making the applications independent from the hardware and software environment in which they run: this environment is composed of computers, operating systems, networks, industrial devices, databases, proprietary and standard applications, etc.

The CCE architecture follows a client/server model: a client requests a CCE service through a CCE application programming interface (API), and a dedicated CCE server executes the service and sends the response back to the requesting application. The roles of client and server of the CCE components may change during the provision of the service: whereas CCE applications or the CCE administration are client applications when calling the CCE services, a CCE server itself may need to call another CCE server in order to provide the required service. A CCE server may also behave as a client with respect to servers outside the execution environment, such as MMS servers on automation systems, database servers or file servers.

The CCE object model

An object-oriented approach has been chosen to hide the differences and the complexities of the various data accesses and services to the application developer. An object represents something that has a counterpart in the real word (a device, a program, a tool, a pallet, etc.). It is specified by three sets of features (see Figure 2): *attributes* which characterise the object, *operations*

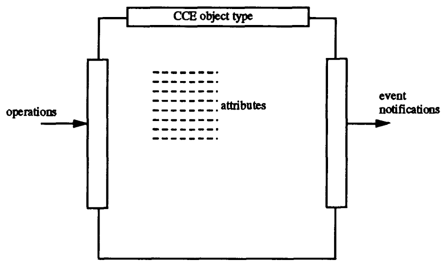

Figure 2 The CCE object description.

which can be applied on the object and *event notifications* which are sent by the object.
For example (see Figure 3), a 'program' object can have the attributes 'name', 'state', 'list of domains',

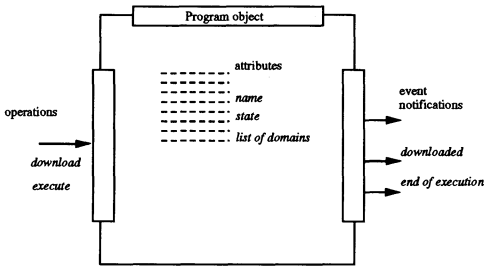

Figure 3 The PROGRAM object description.

the operations 'download', 'execute', and the event notifications 'downloaded' and 'end of execution'. The object interface allows to access and modify the object attributes, to invoke the operations and to subscribe to the event notifications sent by the object. An object is implemented by a server which provides all the services defined at its interface and detects and notifies events specific to this object.

The CCE event model

In the CCE model, events are transmitted as *notifications* and are associated with the CCE objects. They are sent to the client applications by CCE objects. Notifications are part of the object description, they are not modelled as a distinct object. The client application must subscribe to the notifications it wants to receive.

A possibly empty set of pre-defined notifications is specified for each object type. Each notification message has a fixed format. No new notifications can be defined on existing CCE objects (Silicomp, 1996).

The triggering of CCE event notifications is only internally monitored. Client applications may subscribe to the notifications they are interested in. Functional servers, access servers, information servers and processes may subscribe to notifications as well. The object that sends the notification takes care of sending it to all the subscribing CCE components (applications and servers), following a producer/consumer model. Thus, the producer must know the identity of all the consumers.

4 RELATIONSHIP BETWEEN CCE AND CIMOSA

The CIMOSA model (AMICE, 1994) is produced through a process composed of several phases, referred as "stepwise generation": the model is generated by identifying successively the requirements, design and implementation needs, in any appropriate order and iterating as necessary to achieve optimal solutions.

Figure 4 shows the system development cycle defined according to the software engineering terminology (Pfleeger, 1987). The CIMOSA model generation process covers all the four phases, while the CCE platform is used only at implementation phase. CCE and CIMOSA concentrate on the design of the soft-

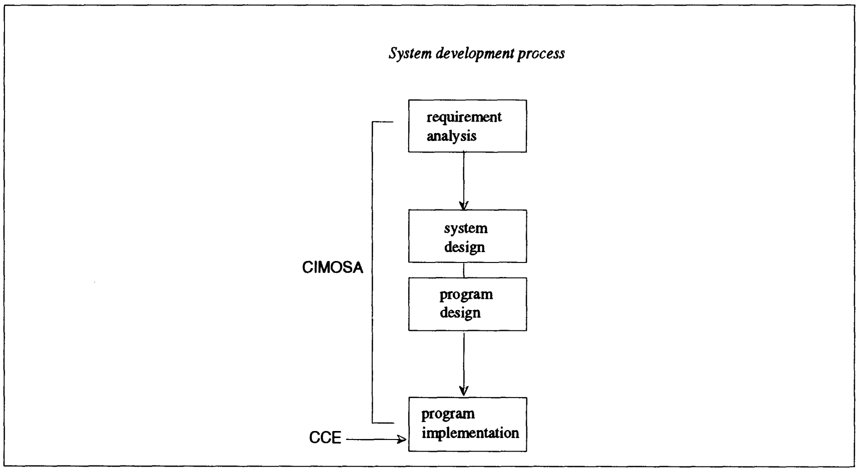

Figure 4 The CIMOSA stepwise generation process

ware architecture for manufacturing applications. CCE was inspired by CIMOSA, and intends to provide implementations of the physical and application integration framework.

CIMOSA identifies four enterprise views to model the major aspects of an enterprise independently of each other, namely the function view, the information view, the resource view and the organisation view. These views are not all present in the CCE model and are not so clearly separated. The basic component of the CCE model is the object, that can represent an information or resource entity, and at the same time it can partially express functional information, represented by the operations defined on the object. The CCE operations do not represent entirely the function view, since the function view includes not only the functionality description but also the control information for the execution of the operations. Thus the function, information and resource views appear as intrinsically, even if partially, supported by the CCE object model. To our knowledge the organisation view has not yet been considered by CCE (see Table 1).

Table 1 CIMOSA views and CCE components relationship

CIMOSA views	CCE components
function	partially supported by CCE operations + CCE notification
information	CCE object
resource	represented by CCE object
organisation	not represented

The CIMOSA integrating infrastructure hides the location, storage and physical placement of information and resources from the requestor of the information and manages the link with the underlying communication infrastructure. The CCE platform has similar objectives and can be used as CIMOSA integrating infrastructure (Pleinevaux, 1996).

CCE offers services that comply with the definition of CIMOSA integrating infrastructure services, namely common, presentation, information and business services (Pleinevaux, 1994). CCE covers the requirements of common, presentation and information services. Only the business services are not covered by the CCE platform. These services are defined in CIMOSA for the execution of the models of the enterprise. In CCE, application knowledge is mainly provided to the system in the form of programs, not in executable models.

5 CCE AND CIMOSA: A COMPARISON OF THE BASIC CONCEPTS

In this section we analyse the definition and use of events and of concepts such as processes and objects, as they are defined in CCE and CIMOSA. We discuss their differences and similarities.

5.1 Events and processes

The events defined in CIMOSA can be compared with the CCE event notifications. Their apparent differences are mainly due to a simplification of the event model realised at implementation phase.

CCE events are sent by objects to the applications that have subscribed to them. CIMOSA events are sent by Enterprise Activities, resources or external components to Domain Processes. Thus, apparently, a difference exists between the event producers and consumers in CCE and CIMOSA.

Let us consider first the *event consumers*. In the CCE model the applications are the only active entities that can send operation requests and receive notifications from the objects.
The CIMOSA concepts of Enterprise Activity, Business and Domain Process are implemented in CCE in the application programs. We call 'CCE process' the execution of a CCE application program. We consider a process an active entity, able to communicate with other processes, and asynchronously exe-

cuted. A CCE process is composed of sub-processes (and sub-sub-processes, at several levels of decomposition) or subprograms and simple operations. This hierarchy may be mapped onto the CIMOSA Enterprise Activity, Business and Domain Process hierarchy in several ways: no unique mapping exists and the implementor decides how to structure his application programs. The most natural mapping is the one that establishes a match between the Domain Process and the CCE process. As consequence of this choice, a mapping exists between the Business Processes and the sub-processes or subprograms composing the CCE process; and a mapping exists between the Enterprise Activities and the sub-sub-processes or the simple operations composing the CCE sub-processes. With this approach the Domain is mapped to a set of CCE processes. Table 2 summarises this example of mapping between CCE and CIMOSA.

Table 2 One possible mapping between CIMOSA and CCE process concepts

CIMOSA concept	CCE concept
Domain	set of CCE processes
Domain Process	CCE process (application execution)
Business Process	application sub-process or subprogram
Enterprise Activity	sub-sub-process or subprogram or operation

However, as we said, the implementor may choose to map in a different way its application programs: he can represent the entire Domain as a single CCE process, the Domain Processes as the sub-processes of this CCE process, the Business Processes as the sub-sub-processes and the Enterprise Activities as the procedures or operations. To the opposite extreme, an example of mapping is shown in Table 3, as third option.

Table 3 Mappings between CIMOSA and CCE process concepts

CIMOSA concept	Mapping 1	Mapping 2	Mapping 3
Domain	CCE process	set of CCE processes	set of sets of CCE processes
Domain Process	sub-process	CCE process	set of CCE processes
Business Process	sub-sub-process	sub-process	CCE process
Enterprise Activity	operation	sub-sub-process/operation	process/sub-process/operation

CIMOSA designers propose one possible mapping of IDL (Implementation Description Language) constructs onto corresponding CIMOSA constructs, where Business Processes are mapped onto parallel processes, Enterprise Activities onto sequential statements and Functional Operations onto expressions (AMICE, 1994).

A fundamental difference between the CCE and the CIMOSA models appears in this context. The CCE application programs are composed of sub-programs and operations, that are defined by the user as ASPI calls. The execution of a CCE application is controlled step-by-step by the user by calling the CCE interface, and getting the results of the execution from the platform. The CIMOSA model specification made by the user allows to define the Enterprise Activities, the Processes and the Domains, at a higher level of abstraction. The model should be mapped to executable processes. If this mapping is realised (this is a target not completely satisfied by the CIMOSA project), the CIMOSA infrastructure takes care

of the execution of the processes and their sequence, at all levels from Functional Operations up to the Domain Processes, without the user being responsible for controlling their execution.
In conclusion, we see that the consumer of CIMOSA events, the Domain Process, corresponds in mapping 2 to the CCE event consumer, the application program (CCE process). The mapping is shown in gray in Table 3.

Now let us consider the *event producers*. CIMOSA events can be generated by enterprise resources, Enterprise Activities or external components. These three types of event producers may trigger event conditions. In CCE, events are sent by objects when pre-defined and internally monitored conditions occur. These conditions may be modified by user application requests or by physical events. User applications require operations on the objects and these operations may modify the monitored object conditions and generate events. Thus one source of events in CCE is the operation that causes the triggering of the event condition. Objects cannot directly trigger their own event condition, nor the event conditions of other objects. CCE objects may model physical devices or physical entities, e.g., a CCE object variable may model the physical value measured by a sensor. The physical device or entity state changes are reflected into the object state and may modify the monitored object conditions. Thus sources of events in CCE may also be the physical events. External applications, not modelled by CCE, may be sources of events as well, and may send events to CCE applications. Thus, CCE event producers are operations on objects, physical devices or entities and external applications. Table 4 summarises the mapping. CCE Event conditions are only internally monitored, and are detected by polling.

Table 4 Mapping between CIMOSA event producer and CCE event producer

CIMOSA event producer	CCE event producer
resource	physical device
	operations executed by user applications
Enterprise Activity	
external component	external application

Another point that may seem to represent a difference is the *triggered action*. CIMOSA does not say explicitly anything about it, but the Domain Process, consumer of the event, acts as triggered action, because it is executed as consequence of the event occurrence. Also CCE allows to associate triggered actions with event occurrences, but these actions are limited to single operations on the same object where the event occurred and the actions are pre-defined in the platform definition. In the current version of CCE, the user cannot define his own actions. Thus we see that the CCE event model is also in this case more limited than the general specification provided by CIMOSA. Table 5 summarises the mapping of event concepts in CIMOSA and CCE.

Table 5 Mapping between CIMOSA and CCE event models

event concept	*CIMOSA RS Model*	*CCE Model*
producer	**enterprise resource +**	**physical entity +**
	Enterprise Activity +	**operation +**
	external component	external application
consumer	Domain Process	process (application execution)
event action	**Domain Process**	**operation on object**

We write in bold style the concepts where a difference exists between the two models. The difference always consists in a more limited functionality supported by CCE.

Other important event issues that must be discussed are: event conditions, event subscriptions, polling of conditions, priority, severity, suspend/resume monitoring. These issues are implementation dependent and they are not specified explicitly by CIMOSA. Another interesting feature is the acknowledgment of notifications, that allows checking the receipt by the subscribers. It is not specified by the CIMOSA model and it is not supported by CCE.
Event conditions have priorities that are used to schedule their processing according to the importance of the event relative to other events; they have severity to represent the effect of the event on the process which is being controlled. The event condition polling may be temporarily suspended and then resumed. CIMOSA provides event condition specification in the form of natural language sentences; subscription to events is implicit in the model, no explicit subscription action is defined by CIMOSA. The other issues are not taken into consideration by the CIMOSA Requirement Specification Model, since they are mainly design or implementation issues.

The CCE platform, on the other hand, provides a possible implementation of the flexible concepts of CIMOSA, thus it does implementation choices that limit the flexibility and generality of CIMOSA: no event condition definition is allowed by CCE, an explicit operation for event subscription is defined. Applications may specify the event polling period. Event notifications may have a specified priority, but no severity. No suspend/resume monitoring option is provided and no event pulling by the client application is allowed.

In conclusion, we think that the CCE platform implements a limited CIMOSA concept of event, imposing restrictions sometimes due to implementation constraints, other times due to a simpler event model.

5.2 Objects and information elements

The CIMOSA objects are generalised concrete or abstract entities of the enterprise. CIMOSA objects model enterprise resources (human, machine or application), and resources are one of the possible sources of events. Objects are defined by the users during the Information Analysis to identify the objects involved in all the Enterprise Activities. Objects may be viewed from different points of view by the users or applications. Thus each object may have several Object Views. Objects are described by either Information Elements or lower-level objects (also called sub-objects). Information Elements are the attributes of the objects. Integrity rules may be defined on Information Elements, in order to impose constraints used to ensure the validity and correctness of the Information Elements (e.g., domain constraints, consistency constraints, and so on). CIMOSA objects are defined as entities used for the execution of the Enterprise Activities and Business Processes, manipulated by them, shared by several Enterprise Activities and Processes.

CCE objects, like CIMOSA objects, model enterprise resources and can send notifications. But CCE object classes are pre-defined by the CCE platform: the user can only create objects belonging to the pre-defined classes. Each object has a unique view; no concept of several object views exists for CCE objects. The definition of CCE objects is provided by the CCE platform, in terms of attributes, operations and notifications. In the current version of CCE, the user cannot modify the object definition, nor define a new class. He cannot modify the event notification or define a new one. The reason is that all these operations require a good understanding of the platform internal.

In conclusion, CCE represents one possible implementation of the CIMOSA objects, but more limited, since it does not support object views, integrity constraints on object attributes and it does not allow the dynamic definition of events. Table 6 summarises the results of the comparison.

Table 6 Mapping of CIMOSA and CCE object concepts

CIMOSA	CCE	comparison result
Enterprise Object + Object Views	object	CCE supports only one Object View for each object
Information Element + Integrity Rule	attribute	no integrity constraints on attributes are supported by CCE
Enterprise Activity and Business Process share objects	operation	CCE operations act on the object on which they are defined
dynamic definition of events generated by resources	pre-defined notifications sent by objects	CCE does not support dynamic definition of events

6 EXAMPLE

Let us consider the following example. Within a real manufacturing scenario, we take into consideration a Flexible Manufacturing System Domain. Inside this domain, we consider the Part Production Domain Process, which deals with the manufacturing process. This Domain Process is composed of the following Business Processes and Enterprise Activities:

```
Domain Process: Part Production
    Business Process: Input Raw Material
    Business Process: Toolset Preparing
    Business Process: Machining
        Enterprise Activity: Support Preparing
        Enterprise Activity: Manufacture Part
    Business Process: Machining Inspection
    Business Process: Output Parts
```

The Business Process `Machining' represents the manufacturing operations that are executed on the raw material in order to obtain the manufactured parts. The component Enterprise Activities decompose the machining process into more elementary steps.
The Enterprise Activity functionality description is as follows:

```
Enterprise Activity: Manufacture Part

    FI: Tool Data, Raw Material
    CI: Part Program
    RI: Machine Tool, Tools, Raw Material
    FO: Part
    CO: Event Notification: End of Manufacturing
    RO: Tool Data
    ES: DONE
    Activity Behaviour:Download Part Program and Tool Data,
                  Start Part Program,
                  Send 'End of Manufacturing' event notification
                  Upload Tool Data
```

The Business Process functionality description is as follows:

```
Business Process: Machining
 Procedural Rules:
    WHEN Start DO BP Input Raw Material
    WHEN ES(Input Raw Material)=DONE DO BP Toolset Preparing
    WHEN ES(BP Toolset Preparing)=DONE DO BP Machining
    .......
    WHEN ES(BP Output Parts)=DONE DO FINISH
```

The 'Manufacture Part' Enterprise Activity functionality is described by the sequence of Functional Operations. These are mapped at implementation phase to a CCE process. This CCE process is composed of a sequence of CCE operations on objects, thus a mapping is defined between each Functional Operation of the Enterprise Activity and each CCE operation, as shown in Table 7. The shortness of this ex-

Table 7 One possible mapping between EA and CCE process

Functional Operations	CCE operations
Download Part Program and Tool Data	CCE_Download (Part Program; Tool Data)
Start Part Program	CCE_StartProgram
Send 'End of Manufacturing' event notification	CCE_Receive(Event Notification)
Upload Tool Data	CCE_Upload(Tool Data)

ample is imposed by space limitations.

7 CONCLUSION

The goal of this paper was the analysis of the relationship between the event management models in the CCE and the CIMOSA architectures. We have first introduced the CIMOSA and CCE models, discussed their relationship and then analysed the mapping existing between the concepts defined in their respective event models.

From the above discussion, we can conclude the following:

- CCE objects are active (they may send notifications), as are CIMOSA resource objects; some differences in the two event models are mainly due to a simplification of the CIMOSA event model realised at implementation phase: the CCE platform implements a limited CIMOSA concept of event, imposing restrictions sometimes due to implementation constraints, other times due to a simpler event model. Main limitations are:
    - the mapping between CIMOSA event consumers and CCE event consumers is left to the implementor choice;
    - CCE limits the event generation to the indirect modifications acted by operations (or procedures) on object states;
    - CCE does not support event triggering;
    - CCE event actions are pre-defined; no event condition definition is supported by CCE;
    - CCE does not support object views and integrity constraints on attributes;
    - no dynamic creation/modification of CCE object classes is allowed.
- The only fundamental difference between the CCE and the CIMOSA models is the level of ab-

straction: the CCE application programs are considered as sequences of service calls to the platform interface, each call being a single CCE operation, while the Domain Processes, Business Processes and Enterprise Activities may model complex behaviours that should theoretically be transformed by the platform into executable processes.

The analysis presented in this paper represents the first step of a future study devoted to the identification of the commonalities and differences between the CIMOSA and CCE architectures. The future work aims at the definition of a mapping tool of all CIMOSA and CCE concepts.

8 REFERENCES

ESPRIT Consortium AMICE (1991) Integrated manufacturing - a challenge for the 1990s, *Computing and Control Engineering Journal*, Special Issue, 101-125.

ESPRIT Consortium AMICE (1994) *Open System Architecture for CIM*. Springer, Heidelberg.

ESPRIT Consortium CCE-CNMA (1995) *CCE: An Integration Platform for Distributed Manufacturing Applications*. Springer-Verlag.

Deliverable to CEC (1993) *CCE Implementation Guide* - Revision 1.0

ISO/IEC (1990) *Industrial automation systems - Manufacturing Message Specification*. ISO/IEC 9506-1,2.

Messina, S. and Pleinevaux, P. (1996) Enhancing CIMOSA with Exception Handling, in *Proceedings of the International Symposium on Robotics and Manufacturing ISRAM*, World Automation Congress'96, Montpellier, France.

Pfleger, S. L. (1987) *Software Engineering The production of quality software*. C. Macmillan, Canada.

Pleinevaux, P. (1994) Integration of Industrial Applications: The CCE-CNMA Approach, in *Proc. of IMSE'94*, European Workshop on Integrated Manufacturing Systems Engineering.

Pleinevaux, P. (1996) CCE: the CIME Computing Environment. to appear in *International Journal on CIM*.

Siemens (1990) *CIM: a management perspective*. Siemens Aktiengesellschaft.

Silicomp (1996) *Appli-Bus*. Grenoble, Version 3.00.

Storr, A. and Rembold U. and Nnaji B. O. (1993) *Computer Integrated Manufacturing and Engineering*. Addison Wesley.

9 BIOGRAPHY

S. Messina graduated in Computer Science at the University of Torino (Italy) in 1989. She received her Ph.D. in Computer Science from the Swiss Federal Institute of Technology in 1996. She has been an assistant/researcher there, in the Industrial Computing Laboratory, since 1990. Her interests include enterprise modelling, object-oriented approaches and business engineering.

P. Pleinevaux obtained a degree in Electrical Engineering in 1984 from the University of Brussels and a Ph.D. in Computer Science from the Swiss Federal Institute of Technology in 1990. He was technical manager of the ESPRIT CNMA and CCE-CNMA projects between 1991 and 1995. He is a lecturer in industrial computer engineering at the Swiss Federal Institute of Technology in Lausanne.