International Journal of Production Management and Engineering

http://dx.doi.org/10.4995/ijpme.2014.2326
Received 2014-04-30 - Accepted 2014-06-05

Enterprise Modeling in the context of Enterprise Engineering: State of the art and outlook

Vernadat, F.B.
Laboratory for Industrial Engineering, Production and Maintenance (LGIPM),
University of Lorraine, Ile du Saulcy, Metz F-57042 cedex 1, France.
Francois.Vernadat@eca.europa.eu

Abstract: Enterprise Modeling is a central activity in Enterprise Engineering which can facilitate Production Management activities. This state-of-the-art paper first recalls definitions and fundamental principles of enterprise modelling, which goes far beyond process modeling. The CIMOSA modeling framework, which is based on an event-driven process-based modeling language suitable for enterprise system analysis and model enactment, is used as a reference conceptual framework because of its generality. Next, the focus is on new features of enterprise modeling languages including risk, value, competency modeling and service orientation. Extensions for modeling collaborative aspects of networked organizations are suggested as research outlook. Major approaches used in enterprise modeling are recalled before concluding.

Key words: Enterprise Engineering, Enterprise modeling, Process modeling, Capability/competency modeling, Risk modeling, Value modeling, Collaborative networked organization, CIMOSA .

1. Introduction

Nowadays, companies are facing drastic competition, unstable business conditions and serious efficiency problems. They must rationalize and optimize their daily operations in a productive and cost-effective way. They must be reactive and agile to quickly face changing conditions. Their operations must be highly interoperable with the partners' ones. To achieve this, these operations must be properly defined, described and put under control. Thus, precise and up-to-date models of these operations are necessary to understand how they work or are organized.

Enterprise Modeling (EM) is concerned with representing and describing the structure, the organization and the behavior of a business entity to evaluate its performances, reengineer its various internal and external flows or optimize them in order to make the enterprise more efficient and effective. A business entity is whole or part of an enterprise or of a group of enterprises (e.g., an extended enterprise, a virtual enterprise, a networked enterprise or a supply chain). Over the last two decades, EM has proved to be a central activity in Enterprise Engineering and Enterprise Integration projects.

Enterprise Engineering (EE) deals with design or redesign of business entities (Kosanke & Nell, 1997). It concerns all activities, except enterprise operations, involved in the enterprise life cycle, i.e., mission identification, strategy definition, requirements definition, conceptual design, implementation description, installation, maintenance and continuous improvement as defined in GERAM (IFAC-IFIP Task Force, 1999). It mostly focuses on engineering and optimizing business processes of enterprises in terms of their related flows (namely, product/material flows, information/decision flows and control flows), resources (human agents, technical agents, components) as well as time and cost aspects. Hence, EM techniques for EE must cater for the representation and analysis of function, information, resource and organization aspects of business entities (AMICE, 1993). They can also cover cost/economic, performance or collaboration aspects.

Enterprise Integration (EI) is concerned with breaking down organizational barriers to create a synergistic whole to improve competitiveness and sustain growth of an enterprise or an networked enterprise. The goal is to make interoperable all elements of the enterprise (i.e., humans, machines as well as IT applications) to facilitate system-wide the 4C's (communication, co-operation, co-ordination and collaboration). An enterprise can be considered to be integrated when the right information is delivered at the right place at the right time. Enterprise interoperability is therefore an essential enabler of EI (Vernadat, 1996, 2007a).

The paper provides a state of the art review of enterprise modeling in the context of EE and EI and is organized as follows. First, definitions and fundamental principles of enterprise modeling are recalled. The CIMOSA modeling framework, which is based on an event-driven process-based modeling language suitable for enterprise system analysis and model enactment, will be used as a reference conceptual framework. Next, the focus is on new features that need to be covered by enterprise modeling languages including risk, value, capability and competency aspects as well as service orientation. Extensions to model collaborative aspects of networked organizations are also discussed as research outlook. A short panorama of tools is then given followed by the conclusion.

2. Enterprise Modeling Principles and CIMOSA

2.1. Enterprise modeling definition

Enterprise modeling (EM), not to be confused with process modelling (Curtis *et al.* 1992, Dalal *et al.* 2004), can be defined as the art of developing models, i.e., abstract representations of a definite part of an enterprise (in a more or less formal way), to accurately represent the structure, behavior and organization of a business entity. It is a generic term which covers the set of activities, methods and tools related to developing models for various aspects of an enterprise or a network of enterprises (Vernadat, 1996; Owen & Walker, 2013; Wikipedia, 2013).

Enterprise models can be used in practice to represent, visualize, understand, communicate, design, rengineer and improve enterprise operations with a focus on quality, cost or delays as well as system efficiency and effectiveness.

Especially, these models are useful (but should not be limited) to: understand and analyze the structure and behavior of an enterprise domain, reengineer a part of the enterprise, evaluate the behavior and performances of business processes before their implementation (either in terms of cycletime or cost), choose the best solution among various implementation alternatives ('what-if' scenarios), evaluate implementation risks and costs, optimize resource selection and management, support model-based integration or support continuous process improvement. However, the prime advantage of EM in industry is to provide a shared view or "picture" of the enterprise that can be communicated to the various actors, i.e., to build a consensus that enforces a common enterprise culture.

Enterprise modeling techniques equally apply to industrial firms, service companies, administrative organizations or even government agencies.

2.2. CIMOSA

Early EM methods, i.e., built before the 90's, were mostly activity centric and based on the functional decomposition principle (e.g., the IDEF and GRAI methods). At the turn of the 90's, business process-centric methods emerged advocating causal and precedence relationships among activities and object flows. The modeling framework of CIMOSA, an open system architecture for integrated manufacturing enterprises (AMICE, 1993), paved the way in this field by introducing an event-driven process-based modeling approach and formalizing the concept of business process (Jorysz & Vernadat, 1990). In addition to the usual functional and information modeling aspects covered by early EM methods, it soon became obvious that resource and organization aspects also had to be addressed to properly assess business processes and related concepts as found in manufacturing companies or in industrial supply chains.

Because CIMOSA has been the root for GERAM and several European and ISO standards for EM, e.g. ISO 19439 (2006) and ISO 19440 (2007), it is used as a reference in the paper.

2.3. Fundamental principles of EM

Due to the complexity and multi-faceted nature of enterprise organizations and especially industrial organizations, enterprise modeling frameworks should respect the following principles:

Principle #1: Plural nature of enterprise models.

This means that there is no such thing as an "enterprise model". There are enterprise models. Indeed, any business entity, be it a manufacturing plant, a R&D department, a branch of a company, a supply chain or a virtual enterprise, is so complex that it is impossible to represent it by one single model expressed in one language. Several models will be necessary and, indeed, the enterprise model is an assemblage of sub-models, each depicting some specific aspects.

Principle #2: Concept of modeling views.

The concept of modeling view or viewpoint is a mechanism that allows to focus on some aspects of a system while discarding others to manage structural complexity. A modeling framework will be powerful, complete and consistent if it provides a minimal set of non-overlaping views to cover all essential aspects of the system. In enterprise modeling, four basic views have been defined by CIMOSA and adopted in GERAM and ISO 19439, namely:

- The Function/Process View, which defines the enterprise functionality (i.e., what has to be done) and the enterprise behavior (i.e., in which order work has to be done).
- The Information/Object View, which defines what are the objects to be processed and to be used and what are their states over time.
- The Resource/Infrastructure View, which describes who/what does what, what is needed to execute operations, which roles, capabilities and skills are required or available.
- The Organization/Decision View, which describes organization units and decision levels of a business entity and their relationships, who is responsible for what or whom, who decides and who has authority on what.

Other views can be defined but ISO 19439:2006 and GERAM have retained the four CIMOSA views as the essential ones. Figure 1 provides a common illustration of Principles 1 and 2.

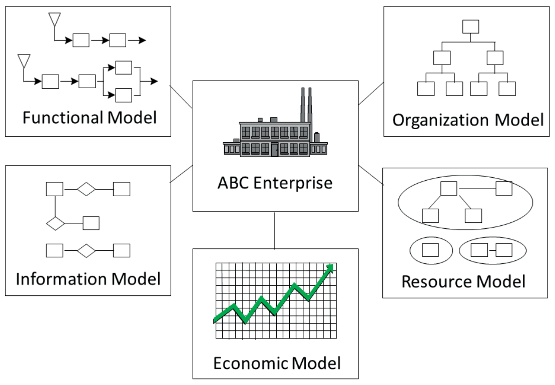

Figure 1. Illustration of the concept of Modeling View.

Principle #3: Three fundamental types of flows.

There are three fundamental types of flows circulating within or across any type of enterprises (excluding financial flows):

- Material flows (made of physical objects such as raw materials, semi-finished parts, products, components, tools…),
- Information flows (made of information and decision objects such as orders, documents, data, computer files, emails, phone calls…),
- Control flows (or workflow, i.e., logical execution sequences of tasks).

Principle #4: Processes versus agents.

At a macro-level, any enterprise can be viewed as a large collection of:

- Concurrent processes (called business processses) executed to achieve business goals and competing for resources on one hand, and
- Interacting agents, or functional entities (i.e., human or technical resources), executing processes on the other hand.

The art of management is to make sure that *in fine* the agents (i.e., the doers) execute the processes in an efficient, effective and economic way to achieve business objectives.

Principle #5: Business process synchronization.

There are three fundamental types of business process synchronization:

1. Event-based synchronization: Events in the form of messages, requests, orders or timers can be generated by one process and used to trigger other processes (e.g., EV2 in Figure 2).
2. Object-based synchronization: A step in a process control flow may need availability of objects produced by one or more steps of different processes.
3. Resource-based synchronization: Execution of a step in a process may need availability of some resource(s) that can also be needed by another step in another process (this is a resource conflict that needs to be solved by means of priority rules or a conflict resolution mechanism).

Figure 2 illustrates the three possible process synchronization mechanisms as well as the three types of flows (Principles 3 and 4).

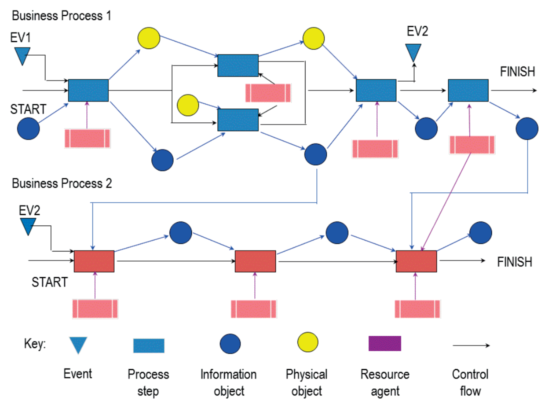

Figure 2. Process synchronization schemes.

Principle #6: The concept of modeling levels.

Enterprise models can be developed at three generic levels as proposed in CIMOSA, GERAM and ISO 14258 (1998). These are:

- Requirements definition: Used to represent "the voice of the users", i.e., what is needed, expressed in a detailed and unambiguous way in a user-orientated or descriptive language.
- Design specification: Used to formally specify one or more solutions satisfying the set of requirements, to analyze their properties and to select the "best" one. These models are expressed by means of a formal and computer executable language (prone to model validation, performance analysis or system simulation).
- Implementation description: Used to state in detail the implementation solution taking into account physical technical constraints. Operational models are defined at this level.

Due to the complexity of models handled in practice (both in terms of number of components and number of relationships existing among these components), a modular and incremental modeling approach is often recommended. To achieve this, most modeling languages use a *building block approach* (i.e., they are made of a limited set of constructs, defined as object classes or template structures (See Appendix) which can be assembled in a "lego" fashion) and support the definition of libraries of *partial models* (i.e., predefined, reusable modules which can be customized and put together) to develop the *particular model* of a specific business entity.

2.4. Simple illustrative example

The example used in the paper concerns a simple functional domain made of a business process dealing with customer order processing. Figure 3 depicts a process *Process_Customer_Orders* triggered by the *NewCustomerOrder* event and made of three process steps (*Enter_Order*, *Create_Customer* and *Process_Order*). Each step employs a number of sub-steps or functional operations as indicated. These operations are performed by two functional entities (*OrderEntryClerk* being a human agent and *OrderProcessing System* being a computer application).

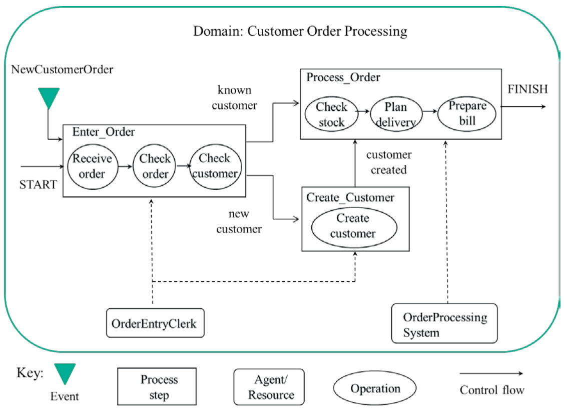

Figure 3. A simple business entity example.

3. Fundamental EM Constructs

To follow the CIMOSA and ISO 19440:1997 presentations, essential constructs of EM for enterprise engineering are presented according to the four basic modeling views introduced in section 2.3. The discussion is based on (Vernadat, 1998) and templates for the constructs are given in the Appendix of the paper.

3.1. Function/Process View

The goal is to describe the enterprise functionality and the enterprise behavior, i.e., what has to done and in which order. Essential constructs are Domains, Events, Business Processes, Enterprise Activities and Functional Operations interpreted as follows: a functional domain is a cluster of business processes which are triggered by occurrences of events and which control the execution of a sequence of enterprise activities to achieve enterprise business goals. Activities are made of functional operations.

A *Domain* is a functional area of the enterprise. It must not be confused with an organizational unit or department. It is used to break down the enterprise into manageable functional areas to cope with system complexity. In that sense, domains must not overlap. Examples of domains could be R&D functions, Marketing, Production Planning, Production, Change Management, Industrial Process Improvement, etc.

Domains are made of a set of consistent business processes. The golden rule is that when a process is assigned to a domain, it must be fully contained in this domain and must not span over several domains. Interactions among domains are called domain relationships and consist of exchanges of events or materialized objects.

An *Event* depicts a change in the state of the system. It is a fact, a happening, solicited or not, that happens at a given time and that implies an action to be taken (i.e., the start of a process or activity chain). Examples of expected events can be the arrival of an order, the sending of a request, the end of an activity or even a timer, i.e., a clock-time (e.g., 5:00 pm). Examples of unexpected events are a fire alarm, the breakage of a machine, an accident, etc.

Events can be endogeneous, i.e., generated within the context of the enterprise (e.g., by its resources or its activities) or exogeneous, i.e., generated by the environment of the enterprise (e.g., customers, suppliers, insurance companies, banks, revenue & taxation offices, etc.).

A *Business Process* can be defined as a partially ordered sequence of process steps executed to fulfil some business goals of the enterprise. Process steps are either elementary (enterprise activities) or composite (sub-processes). End-to-end processes fully contained in a domain are called *Domain Processes*. Business processes describe the *enterprise behavior*, i.e., in which sequence things are done.

A process must have a start point (START), defining the process trigeering condition, and an end point (FINISH), defining the results to be achieved by the process. All steps in between must be modeled as part of the process.

Occurrences of processes are triggered by occurrences of events. However, the triggering conditions of processes (or START conditions) can be much more complex. They can be made of a single event (occurrence of the event will trigger an occurrence of the process), they can be a combination of events using Boolean logical operators (e.g., arrival of a customer order AND 8:00 am) or they can be a logical combination of events and Boolean conditions (e.g., Start Process_Customer_Orders AND non-empty (CustomerOrder file)).

The enterprise behavior in a process is described by the sequencing of the process steps. For instance, in CIMOSA, this is done in the form of *WHEN (condition) DO action* rules, where the condition part of the rule describes the triggering condition defined above and the action part indicates the next step to be activated. The formalism used is based on the ECA (event[condition]/action) rules in which the [condition] clause is optional (Chakravarthy, 1989). With this formalism, it is possible to describe the following basic situations in a control flow: pure sequence, conditional branching, parallelization (synchronous or asynchronous), rendez-vous (synchronous or asynchronous) and the feedback loop (constructed with a Go-to).

For instance, using the Business Process template provided in the Appendix, the description of the process of the example of Figure 3 is:

BUSINESS PROCESS

Identifier: BP-031 *Name*: Process_Customer_Orders
Description: Used to process customer orders
Objectives: Process customer orders within 24 hours
Constraints: Process orders in arrival order
Declarative rules: Operates from 8:00 am until 5:00 pm
Triggering events: EV-001/NewCustomerOrder
Process Behavior:

WHEN (START WITH NewCustomerOrder AND past (8:00am) AND before (5:00pm)) DO Enter_Order
WHEN (ES(Enter_Order) = New customer) DO Create_Customer
WHEN (ES(Enter_Order) = Known customer) DO Process_Order
WHEN (ES(Create_Customer) = Customer Created) DO Process_Order
WHEN (ES(Process_Order) = Done) DO FINISH

Ending statuses: Done
Comprises: *Enter_Order, Create_Customer, Process_Order*
Where-Used: Nil

Convention: The name of business processes and enterprise activities must always start with an action verb (because they represent functionality, the nature of which is "to do").

An *Enterprise Activity* is the locus of action. It transforms inputs x into outputs y using time t and a set of resources R. It can be modeled by a transfer function f_R such that $y = f_R(x, t)$, where f_R depicts the activity behavior. f_R can be an algorithm, a procedure, a recipe or a protocol. It models *enterprise functionality*.

Figure 4 provides generic representations of the activity. Figure 4a is the original representation proposed by Ross (1977) in SADT while Figure 4b is the CIMOSA representation which generalizes the SADT box.

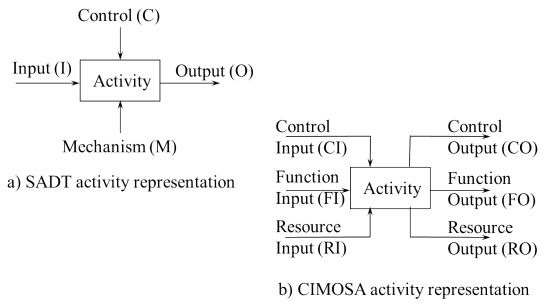

Figure 4. SADT and CIMOSA activity representation.

The vectors of inputs *x* and ouputs *y* are made of manifestations of enterprise objects called object views and representing objects in a certain state. Three types of inputs and three types of outputs of an activity can be distinguished as follows:

- *Function input (FI)*: These are all the objects processed or modified by the activity.
- *Control Input (CI)*: These are information objects constraining the execution of the activity but not modified by the activity.
- *Resource Input (RI)*: These are all the objects used in support to the execution of the activity.
- *Function Output (FO)*: These are all the objects created or transformed by the activity.
- *Control Output (CO)*: These are the events generated by the activity, if any.
- *Resource Output (RO)*: Returns the status of resource objects at the end of the activity.

For instance, the representation of the activity *Enter_Order* of Figure 3 using the Enterprise Activity template of the Appendix will be:

ENTERPRISE ACTVITY
Identifier: EA-001 *Name*: Process_Order
Description: Get order details and check data
Objectives: Process an order in less than 1 hour
Constraints: Process orders in arrival order
Declarative rules: Operates from 8:00 am until 5:00 pm
Required Capabilities: CS-001/OP_capabilities
INPUTS:
  Function: OV-002/Customer_Order, OV-022/Customer
  Control: OV-032/Product_File, OV-033/Customer_File
  Resource: FE-010/OrderEntryClerk
OUTPUTS:
  Function: OV-003/Checked_Customer_Order
  Control: Nil
  Resource: Nil
Activity Behavior: BEGIN FE-010.Review-order (OV-002), FE-010.Check-order (OV-002), FE-010.Check-customer (OV-022, OV-033) END
Ending Statuses: {known customer, new customer}
Where-Used: BP-031/ Process_Customer_Orders

The required capabilities indicate the set of capabilities expected from the resources needed to execute the activity. The ending statuses are all possible states in which the activity can finish its execution (also called output or intermediate events in other modeling languages).

Finally, enterprise activities can be further broken down into functional operations to explicit the activity behavior, i.e., how their transfer function is actually implemented. In the previous example, the activity *Process_Order* is made of three sequential operations.

A *Functional Operation* (FO) is an atom of functionality in the model in the sense that it corresponds to one instruction interpretable and executed by one resource object, called a functional entity (FE). It must be denoted by a verb.

Using a dotted notation, it can be specified as:

FE-name.FO-name (argument list)

FE-name is the name of the functional entity requested to execute the operation, *FO-name* is the label of the operation and *argument list* is list input and output parameters passed to or returned by the operation during execution.

3.2. Information View

The information view describes inputs and outputs of enterprise activities in the form of objects. More precisely, these inputs and outputs are objects in certain state at a certain time, hereby called object views.

The various states of the enterprise objects can be modeled in the form of state-transition diagrams, in which states are the object views and transitions are activities transforming an input state into an output state. These diagrams are also called object state transition networks (OSTN) in some methods.

A typical example concerns a machining process in which a part is sequentially machined by several machines. For instance, the blank part is first mounted on a milling machine for surface milling. Then, the milled part is mounted on a drilling machine to be drilled. Finally, the drilled part goes to a third machine for surface finishing to produce the finished part. The part is the enterprise object and "blank part", milled part", "drilled part" and "finished part" are four object views of the part object.

According to CIMOSA and ISO 19440, the two fundamental constructs of the Information View are: Enterprise Object and Object View. Templates proposed for these constructs are given in the Appendix.

An *Enterprise Object* is any entity of the enterprise having a life cycle of its own. It must be uniquely identifiable (i.e., it has an identity) and is described by a set of descriptive properties. Each property is defined by a name and a data type that can itself be an object or a set or list of objects.

Two abstraction mechanisms borrowed from the information system modeling theory can be used to represent semantic links among enterprise object classes: (1) the *generalization/specialization hierarchy* (or *Is-a* link) to relate a super-class to its more specialized classes, and (2) the *aggregation mechanism* (or *Part-of* link) to relate a compound class to its component classes. These are the same mechanisms as those found in object class diagrams of the Unified Modeling Language or UML (http://www.uml.org/).

An *Object View* is a manifestation or state of an enterprise object or a combination of enterprise objects at a certain time.

Usually, an object view is defined over one enterprise object as a projection over the set of properties of the object, i.e., whole or part of the properties of the object (some properties can be derived or calculated properties from the properties of the object) are used in the view. This would be the case for the example of the machined part just mentioned.

However, an object view can be a manifestation constructed over several enterprise objects. A typical example is the customer order. A customer order is not an entity of the enterprise in its own right because it depends on two other enterprise objects. Indeed, any customer order is the relationship between a customer (an enterprise object) with a product (another enterprise object) ordered in some quantity on a certain date.

The object view construct template therefore distinguishes between the so-called *Leading Object* (the main object on which the object view is defined) and *Related Objects* (the other enterprise objects providing properties of the object view).

Last but not least, two fundamental types of object views must be distinguished: *physical object views* and *information object views*. Physical object views represent the physical manifestations of the object, i.e., the ones that can be situated in shape, space and time. Information object views are purely descriptive and are usually computer or database representations of the objects. This distinction of nature is essential to separately represent the physical or material flows from the information flows as defined in section 2.3 (see Figure 2).

As a matter of example, the description of a customer order object class could be:

ENTERPRISE OBJECT
Identifier: EO-002 *Name*: Customer_Order
Description: Orders received from company customers
Is-a: EO-001/Order
Part-of: Nil
Properties:
Date: Date
    Customer: EO-100/Customer
    Billing address: Address
    Delivery address: Address
    List of items: list [1,30] of EO-101/Order_Item

3.3. Resource View

The resource view is intended to describe all enterprise objects used in support to the execution of enterprise activities (i.e., technical agents or human agents). It can also be used to describe the physical infrastructure of the enterprise.

Ontologically speaking, CIMOSA distinguishes active resources from passive resources. Active resources, called *Functional Entities*, are resources that can receive and send messages and can execute functional operations. Therefore, they have a controller device or function that can interpret the functional operation received as a command in the form of a message in a certain format transported by some protocol via a communication channel or media. Passive resources, called *Components*, are any kind of resources other than functional entities that cannot execute any functional operation (e.g., a tool, a cart, a truck, a telephone, etc.).

Functional entities can be further partitioned into three fundamental sets: *Humans*, *Machines* and *Applications*. Humans are all kinds of human agents. Machines are machines with a controller (e.g., robots, CNC machines, a truck with its driver equipped with a GSM or cellular phone, a printer connected to a computer network, etc.). Applications are IT systems or computer applications including database systems. Each category can be further decomposed.

To model resources and their charateristics, at least two essential constructs are needed: Resource and Capability Set. Templates of these constructs are given in the Appendix.

The *Resource* construct will be used to model either functional entities or components as well as aggregations of these in the form of resource sets or cells as found in manufacturing systems. The *Capability Set* construct is used to model additional specific aspects of the resource, usually in the form of technical constraints, and dealing with functional capabilities, object related capabilities, performance capabilities of operational capabilities of the resource.

From an ontological point of view, an active resource or functional entity is an enterprise object fundamentally defined by its non-empty list of functional operations and the list of capabilities that it can provide. An object view defined on the resource object can be used to describe additional intrinsic characteristics of the resource such as availability, capacity, location, reliability (MTTR, MTBF…), mono or multi-tasking capability, priority rules, etc.

As an example, let us consider the description of the *OrderEntryClerck*, a human functional entity in Figure 3, together with its capability set.

RESOURCE
Identifier: FE-010 *Name*: OrderEntryClerk
Description: Employee processing orders
Class: Functional Entity
Cardinality: 1 *Object View*: OV-010/OEClerk_characteristics
Capabilities: CS-123/OEClerk_capabilities
Operations: Review-order, Check-order, Check-customer, Create-customer, Create-daily-report
Made-of: Nil
Where-Used: Nil

CAPABILITY SET
Identifier: CS-123 *Name*: OEClerk_capabilities
Description: Capabilities for order entry & verification
Capabilities:
Functional: to receive customer orders, to check customer data, to check orders, to create customer records, to make daily reporting
Object-related: can deal with regular oders, special orders and sub-contracting orders
Performance: order processing time < 60 mn
Operational: must produce a daily report before leaving the office

3.4. Organization View

The organization view is used to describe the organizational structure (i.e., administrative and functional structures) of the enterprise as a collection of organizational entities (or decision centers) related to one another either by hierarchical (i.e., decisional) links or co-lateral (i.e., network) links. An organization entity may be a work-position, a work-station, a team, a department, a division, a plant or a branch, a direction or the entire enterprise. Organizational entities are usually assigned responsibilities and authorities on tasks or processes, systems, data or resources and people of the enterprise.

CIMOSA and ISO 19440 provide two basic constructs for the organization view (See Appendix): Organization Unit and Organization Cell.

An *Organization Unit* is a single decision centre reduced to one functional entity assigned to a specific job (defined by its job description) and provided with well-defined responsibilities and authorities. A responsibility is a duty for which a person is accountable (e.g., the order *O1* of customer *X* must be delivered by the end of the week) while an authority is a right (e.g., to hire someone, to sign a contract, to delete data in corporate databases, etc.).

An *Organization Cell* is an aggregation of organization units and/or cells to form a higher level decision center in the organization structure. It is placed under the management of one functional entity (a person, who is the manager) and has a set of well-defined responsibilities and authorities on components or people of the enterprise. This way, jobs can be aggregated in teams, teams in departments, departments in divisions, divisions in directions and directions in the entire enterprise to reflect the organization chart of the enterprise.

Using the "*Belongs to*", "*Comprises*" and "*Related to*" links of the two construct templates, it is possible to describe any kind of enterprise organizations, be it a single enterprise, a multi-branch enterprise or a networked enterprise.

4. EM Language Extensions

In the previous sections, the essential constructs for EM as understood all over the 90's have been recalled. Since then, a number of extensions to capture more information or to cover new aspects have been proposed. They are reviewed in this section.

4.1. Competency/skill modeling

Human aspects were poorly addressed in the CIMOSA, GERAM or ISO 19440 frameworks. Moreover, human agents, considered as a specific type of functional entites, are modeled somewhere in the resource view, aside to machines and applications in the enterprise architecture (see section 3.3).

PERA (Purdue Enterprise Reference Architecture), another Enterprise Architecture Framework for industrial engineering (Williams, 1992), enforced and positioned the human architecture at the heart of the framework.

To capture more human-related aspects in the model, Harzallah & Vernadat (2000) first proposed to add the *Job Profile* field in the Organization Unit construct. The job profile (or job description) is defined by a list of skills or competencies that the job holder must have. The job holder for this organization view is indicated in the *Functional entity* field (must be a person).

Next, the Capability Set construct has been expanded to also cope with competency or skill description of a human agent. Indeed, technical agents (i.e., machines and applications) are described in terms of their capabilities while human agents can be described by their capabilities and competencies. Thus, the Capability Set construct as been augmented with a *Competencies* section to become the *Capability/ Competency Set* construct presented in the Appendix. To generalize these concepts, a formal model of individual competencies has been proposed by Harzallah *et al.* (2006).

Using this new template, the capability/competency set of the *OrderEntryClerk* human resource of Figure 3 becomes:

CAPABILITY/COMPETENCY SET
Identifier: CS-123 *Name*: OEClerk_capabilities
Description: Capabilities for order entry & verification
Capabilities:
- *Functional*: to receive customer orders, to check customer data, to check orders, to create customer records, to make daily reporting
- *Object*-related: can deal with regular oders, special orders and sub.contracting orders
- *Performance*: order processing time < 60 mn
- *Operational*: must produce a daily report before leaving the office

Competencies:
- *Knowledge*: advanced order processing training, college accounting diploma
- *Know*-how: minimum 2-years experience in customer order processing, know how to create customer records, know how to check customer data, know how to produce daily reports
- Know-whom: to be organized, to be meticulous, to be service oriented, to meet deadlines, high integrity

4.2. Risk and value modeling

The aim of a business process is to achieve or fulfil a business goal. Should this goal have been rated as reasonably or highly valuable for the enterprise, its achievement must be monitored in terms of performance management. Industrial performance can be measured and assessed from four main dimensions: benefits, costs, value and risk. Therefore, risk management and value management are gaining ever increasing importance in industrial engineering.

Indeed, value reflects the expectations with regard to goal or objective satisfaction while risk reflects the concerns/fears of not achieving the goal or objective. Value is very much related to stakeholders' satisfaction (including usual quality, cost and delay performance criteria). In other words, the concept of value is based on the relationship between satisfying needs and expectations and the resources required to achieve them. The aim of Value Management is to reconcile all stakeholders' views and to achieve the best balance between satisfied needs and resources (IVM, 2014).

According to ISO 31000 (2009), risk is the effect of uncertainty on objectives, whether positive or negative. Risk Management is the identification, assessment, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities.

On the basis of this understanding of value and risk, Shah *et al.* (2014) have proposed a conceptual value-risk model based on enterprise modeling concepts that can support decision-making and performance management in manufacturing systems. Value and risk are analyzed at the enterprise activity level and can be aggregated to the business process level.

The proposed conceptual value-risk model is depicted by Figure 5. It can be interpreted as follows: An activity *i* has to achieve objectives that will be materialized by results (Output(s) of Activity *i*). The activity is subject to one or more risk factors that need to be analyzed and evaluated. *Risk judgements* will result on how well risks have been mitigated and *value judgements* will result on how good or bad objectives have been met.

The conceptual model can be complemented by a quantitative model to compute aggregated values of risk and value at the process level.

To quantify the process risks R_p, the risk assessment approach of Larson & Kusiak (1996) is adopted and modified. First, the likelihood of all risk events of event factors multiplied by their corresponding consequences (or severity) is modelled for each activity using Equation 1.

$$\text{Risk of activity } i = P_{ij}\left(C_{ij}^{q} + C_{ij}^{c} + C_{ij}^{t} \cdots\right) \qquad (1)$$

P_{ij} is the probability of a risk event j on activity i and $C_{ij}^{q}, C_{ij}^{c}, C_{ij}^{t}$ are consequences on quality, cost and time objectives, respectively.

The global risk for activity i subject to J risk events is given by Equation 2:

$$\mathrm{R_i} = \sum\nolimits_{j=1}^{J}\left(\mathrm{P_{ij}} \times \mathrm{C_{ij}^{obj}}\right) \qquad (2)$$

Where, $\mathrm{R_i}$ is the risk magnitude of activity i.

However, all risk events are not equally important in any given scenario. Some risks are more important than others depending on what objective(s) they are influencing. Therefore, a weighting factor or importance index $\mathrm{d_{ij}}$ is introduced to model the importance of the risk events in the scenario S. So, Equation 2 can be rewritten as Equation 3:

$$\mathrm{R_i} = \sum\nolimits_{j=1}^{J} d_{ij}\left(\mathrm{P_{ij}} \times \mathrm{C}_{ij}^{obj}\right) \qquad (3)$$

Therefore, the global risk of the process path $\mathrm{p_k}$ is given by Equation 4:

$$\mathrm{R(p_k)} = \sum\nolimits_{\forall i \in p_k} \mathrm{R_i} = \sum\nolimits_{\forall i \in p_k} \sum\nolimits_{j=1}^{J} d_{ij}\left(\mathrm{P_{ij}} \times \mathrm{C_{ij}}\right) \qquad (4)$$

The probability P_r $(p_k)$of the path set (or scenario) must verify Equation 5:

$$\sum\nolimits_{k=1}^{K} P_r(p_k) = 1 \qquad (5)$$

So, the expected risk of the process P made of K path sets is given by Equation 6 which gives the expected risk of the whole process:

$$E(Rp) = \sum_{k=1}^{K} P_r(p_k)\left(\sum_{\forall i \in p_k} \sum_{j=1}^{J} d_{ij}\left(\mathrm{P_{ij}} \times \mathrm{C_{ij}}\right)\right) \qquad (6)$$

To compute an interim value (value at an activity level) using the conceptual model, the output of the activity i is judged with regard to the objective which is appraised by a performance measure, using utility theory principles.

As a result, value functions *v(c)* are developed for each performance measure *c* at the activity level. The overall value of a process is then obtained by aggregating the interim values of all its activities using an aggregation operator (e.g., weighted sum or Choquet integral) (Clivillé *et al.*, 2007). Similarly, the outputs of activity *i* are judged using risk measures to develop the interim risk function *v(r)* and aggregated to obtain the global risk indicator.

4.3. Business service orientation

The business process approach as practiced throughout the 90's has introduced a natural and horizontal way in organizing business systems as opposed to previous vertical and activity centric approaches. However, rapidly changing market conditions and business requirements tend to increase the gap between what the business requires and what IT can deliver. To build agile, i.e., more flexible, extensible and evolvable environments in which IT can be more quickly and easily aligned with the business, many organizations are adopting Service-Oriented Architecture (SOA) principles to close the gap (Herzum, 2001, Vernadat, 2007a).

SOA is emerging as a new wave for building agile and interoperable enterprise systems. It is an IT strategy consisting in exposing as encapsulated services the business functions of an application. Broadly speaking, an SOA is essentially a collection of services.

In technical terms, SOA is about designing and building IT systems using heterogeneous network addressable software components (preferably communicating over Internet). These interoperable standards-based components or *services* (i.e., callable and reusable functions accessible by their interface) can be directly invoked by business users or executed as steps of business processes. They can be combined, modified, or reused quickly to meet business needs. They can be implemented as Web Services (WS) or functions of Web applications and, therefore, be located anywhere on the Web (Herzum, 2001; Khalaf *et al.*, 2005).

From an IT perspective, a service is a piece of business functionality that can be invoked by its locator and its interface(s) as publicly published in a standard format. Details of its implementation are hidden. The service can be implemented using any IT technology (C program, C++, PL/SQL, Java, .Net/C#…).

From a business user perspective, a service is an invokable piece of functionality that will return a result (i.e., provide a service to a client) under the conditions defined in its service level agreement (SLA). It has higher grain than a WS.

Practically, *Business Services* are functional or informational business components designed to be accessed as such by service clients or consumers. They represent stand-alone and reusable business functions of the real world (e.g., Declare birth of a child, Change personal address data or Create a customer).

Compared to a business process, a business service can be one step in a business process, a business process can be a choregraphy of execution of a set of business services and a business service can be implemented by a business process. SOA is an IT technology that can be used to implement business services.

A *Business Service* is a discrete piece of functionality that appears to be platform-independent, logically addressable and self-contained from the point of view of the service caller. It must be uniquely identified within the enterprise and has a service owner. Logically addressable means that it can be dynamically invoked simply by calling its logical address or universal resource identifier (URI), thus without having to know where it is physically located. Self-contained means that the service exists as a whole and that it maintains its own state. In practice, it is recommended to develop only stateless services (i.e., each service call is independent of previous calls), especially if they are implemented as Web services. However, stateful services may also exist (they require that both the consumer and the provider share the same consumer-specific context – passed in the message) (Vernadat, 2007b; OMG, 2012).

Physically, a business service can either be performed by a human agent or a technical agent.

For instance, the *Create_Customer* activity could be a stateless business service offered by the *Customer Order Processing* domain. In this case, the service can either be invoked within the course of the *Process_Customer_Orders* process or be called as a separate function by raising the *Create_Customer_Request* event as illustrated by Figure 6.

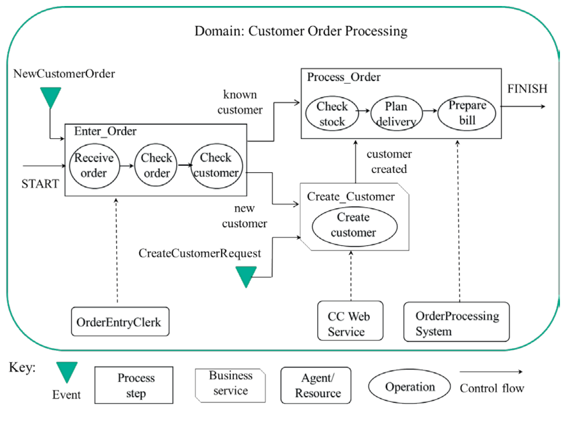

Figure 6. Business process with business service.

5. Research Outlook: Collaborative Network Organizations

Enterprise networks and Collaborative Networked Organizations (CNOs) are made of a finite set of collaborating entities from different partner companies working together. This implies among other things business process synchronization, sending events, exchanging material or information objects and even, in some cases, sharing resources.

A Collaboration View has recently been proposed as an extension of EM frameworks to describe the context and characteristics of the collaboration from the point of view of each partner (Kosanke *et al.*, 2014). Indeed, the enterprise architecture of each partner belongs to this partner while the architecture of the whole network does not belong to any specific partner.

Following the model view concept of CIMOSA/ ISO 14258, this view identifies three new language constructs with their respective elements to globally describe collaboration as commonly depicted in the literature (Camarinha-Matos & Afsarmanesh, 2007; Camarinha.Matos *et al.*, 2009; Jagdev & Thoben, 2010). These are (see Appendix for construct details):

- Collaboration Domain
- Collaborating Partner
- Collaboration Point

The *Collaboration Domain* construct is used to describe a given collaboration area between the enterprise at hand (*Us*) and its partner companies identified as its Collaborating Partners. It indicates the collaboration entities (i.e., processes, activities, resources or organization units described in the other modeling views of the enterprise architecture), the exchanged objects in terms of events and object views as well as the list of Collaboration Points, i.e., gateways supporting the various exchange flows with the different partners.

Collaborating Partners of the enterprise at hand are business entities involved in the collaborative exchanges with this enterprise (*Us*). They are defined in terms of their role in the collaboration (e.g., supplier, provider, consumer or retailer). A description of their ICT environment can be made.

Collaboration Points, as defined by Li *et al.* (2013), represent the collaboration interfaces between collaborating entities of an enterprise and those of one of its Collaborating Partners. The type of collaboration can be unidirectional or bidirectional, synchronous or asynchronous or based on mutual adjustment. The exchange media or transportation means supporting the exchange flows must be specified.

A partner enterprise, or one of its branches, can be involved in several Collaboration Domains. Each Collaboration Domain may comprise several Collaboration Points.

Note: The use of these three modeling constructs enables the representation of a collaboration model from the point of view of a given company with reference to the models of its individual partner enterprises by describing its collaboration domains and interacting partners.

Figure 7 illustrates a collaboration domain between two partner enterprises (ABC and XYZ) containing two collaboration points (CP1 and CP2) for the exchange of orders and invoices. The model belongs to the architecture of the XYZ company. The "+" in functional boxes indicates that they represent business processes.

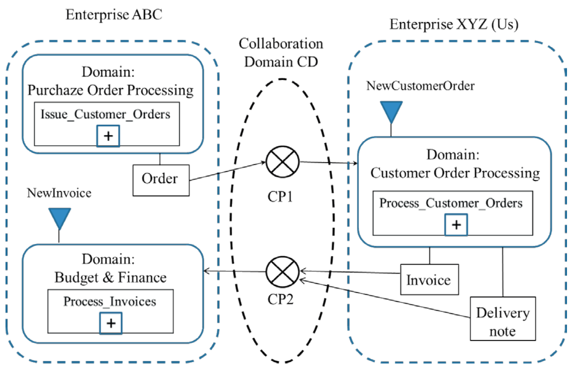

Figure 7. Collaboration example.

6. Panorama of EM Methods

Enterprise modeling techniques have their roots in the Entity-Relationship (ER) model of Chen (1976) and in SADT (Structured Analysis and Design Technique) of Ross (1977), both developed for software engineering. SADT has later been renamed IDEF0 in the 80's and has provided a generic definition of the activity in the form of the ICOM (Input – Control – Output – Mechanism) box (see Figure 4) to the IDEF Method (www.idef.com). SADT/IDEF0 has formalized the functional decomposition process to break down complex systems into sub-systems (Golden rule: a system can be broken down into at least three sub-systems and no more than six) while the ER model has formalized the concept of entity in information systems, which has later been extended to the concept of object class in the object-oriented approach.

The first real enterprise modeling techniques and languages appeared in the 80's with the IDEF method (ICAM, 1981) and the GRAI method (Doumeingts, 1984) which then became GRAI-GIM (Roboam *et al.*, 1992).

- IDEF Method: The IDEF (ICAM Definition) Method was developed within the framework of the Integrated Computer-Aided Manufacturing (ICAM) Program of the US Air Force and sponsored by the US Department of Defence (DoD) to model large manufacturing systems. It is made of a series of modeling methods: IDEF0 to model enterprise activities with SADT, IDEF1 to model data structures in the form of extended entity-relationship models and IDEF2 to model system behavior using the queueing formalism of the simulation language SLAM (ICAM, 1981). Unfortunately, it poorly addresses resource aspects, does not cover organizational aspects and cannot model business processes. Furthermore, IDEF2 has been abandoned.
- IDEF3: Because IDEF0 can only model enterprise activities and not business processes, the IDEF3 method has been added to IDEF in 1992 (Mayer *et al.*, 1992). IDEF3 is a popular method in North America to represent process control flows in manufacturing systems. Process steps in a control flow are called units of behavior (UOBs) and are connected by junction boxes (AND, OR or XOR boxes). The major strength of IDEF3 is that it can represent any kind of business processes (well-structured, semi-structured or ill-structured) although its syntax and semantic are not precisely defined.
- GRAI-GIM: It is a methodology and enterprise architecture framework containing enterprise modeling tools, including IDEF0 for activity modeling. It covers data (i.e., information), process (i.e., function) and operational aspects (mixing resource and organization aspects). The major advantage of GRAI-GIM is its ability to analyze the decision system of an enterprise for which it provides the GRAI grid to identify decision centers and the GRAI nets to model operational and decision activities. In addition, GRAI-GIM has been provided with a performance evaluation framework for economic evaluation (Doumeingts & Vallespir, 1995; Doumeingts & Ducq, 2001).

- ARIS ToolSet: ARIS (Architecture for Information Systems) was originally, as its name suggests, an architecture framework for building IT systems (programs and databases). It has then been proposed as an enterprise architecture framework (Scheer, 1992, 1999). Although made of four views (data, control, function and on top organization), it mostly covers function and information aspects and very partially organization and resource aspects. Each view comes with its modeling constructs. Especially, business processes are modeled as sequences of events and activities called Event-Process Chains (EPC). EPC has been a popular business process language because it has been promoted by ARIS ToolSet, the modeling tool of ARIS which is number one sales in the world, and recommended by SAP for implementing the SAP ERP. Recently, the EPC notation has been replaced by BPMN and its use is declining in industry.
- UML: Although clearly announced as a modeling method for software-based systems by OMG (Object Management Group), UML (Unified Modeling Language) can be useful in enterprise modeling, especially for building models of enterprise objects and object views in the information view but also to specify computer and IT systems to be implemented in the particular architecture of an enterprise. UML provides object class diagrams, activity diagrams, sequence diagrams and collaboration diagrams that can be useful to analyze, design and specify IT components of the enterprise. Starting with version 1.3 (OMG, 2000), the current version of UML is 2.4 (OMG, 2011a). Version 2.5 is now available in beta version.
- BPMN: Because there were just too many similar but different and non interoperable formalisms, languages and specific tools to model business processes, there was a need for a *de facto* standard language or notation. BPMN (Business Process Model and Notation), coming from IT, has become the Esperanto in the field. Most commercial systems for modeling business processes provide a BPMN interface, including ARIS ToolSet. As the name says, it is a business process description notation to be used to capture process structure and behavior. This is a graphical notation. Models are boxes connected by arrows and documented by text. It is however a simple and sufficiently precise notation to build models of process control flows that can easily and quickly be communicated to many stakeholders of the enterprise. BPMN is not an enterprise modeling language. It is a process modeling language which covers only enterprise behavior description (i.e., process control flow and some related elements). Originally, proposed by BPMI (Business Process Management Initiative), BPMN is now supported by OMG. It is widely used in industry. The current version is BPMN 2.0 (OMG, 2011b).
- ArchiMate: ArchiMate (The Open Group, 2012) is an enterprise architecture modeling language to support the description, analysis and visualization of particular enterprise architectures within and across business domains. It is a *de facto* standard based on concepts of the IEEE 1471 standard intended for describing the architecture of a “software-intensive system”. As such, it has a strong IT flavor and can only partially cover the four essential views of enterprise modeling as defined in ISO 19439. The aim of ArchiMate is to become an open standard in enterprise architecture to generalize previous modeling frameworks. It is currently gaining popularity. Mapping ArchiMate with TOGAF (www.opengroup.org/togaf/), one of the currently prominent reference architecture frameworks, has been addressed. Both TOGAF and ArchiMate are supported by the Open Group (http://www.opengroup.org) and find their way in industrial or administrative applications.
- SoaML: Originally intended as an open source specification for describing a UML profile and metamodel for the modeling and design of services in a Service Oriented Architecture, SoaML (Service oriented architecture Modeling Language) provides a consistent framework for (1) modeling business services and business processes and (2) for implementing these services and processes with Web services or SOA technology (OMG, 2012). In that sense, it is the perfect framework for implementing the business service orientation presented in section 4.3.

7. Conclusion

Enterprise Modeling emerged in the late 80’s as a technique for describing various aspects of an enterprise, especially for the purpose of analysis, design, reengineering, optimization, performance evaluation and even control of a given business entity. At that time, CIMOSA demonstrated that this can be achieved in an integrated and uniform way by developing modeling language constructs and templates. These templates can then easily be implemented in the form of object classes to be stored in relational or object databases and used by advanced modeling and simulation tools. This what most EA and EM commercial tools now do.

Since then, EM has evolved and has been constantly enriched with new constructs to capture more details or cover additional aspects. In the paper, fundamental principles and constructs of EM have been first reminded before presenting recent extensions and a short panorama of well-known modeling techniques to make a state of the art.

Further developments can still be proposed to cover additional aspects. We can mention soft issues (for instance, trust or human behavior), security and legal aspects in collaborative environments or value in networked enterprises, just to name a few.

References

AMICE. (1993). *CIMOSA: Open System Architecture for CIM*, 2nd revised and extended edition. Berlin: Springer-Verlag. 234 pages.

Camarinha-Matos, L.M., Afsarmanesh, H. (2007). A comprehensive modeling framework for collaborative networked organizations. *Journal of Intelligent Manufacturing*, 18(5): 529-542. doi:10.1007/s10845-007-0063-3

Camarinha-Matos, L., Afsarmanesh, H., Galeano, N., Molina, A. (2009). Collaborative networked organizations - concepts and practice in manufacturing enterprises. *Computers & Industrial Engineering*, 57(1): 46-60. doi:10.1016/j.cie.2008.11.024

Chakravarthy, U.S. (1989). Rule management and evaluation: An active DBMS perspective. *ACM SIGMOD Record*, 18(3): 20-28. doi:10.1145/71031.71034

Chen, P.P.S. (1976). The entity-relationship model: toward a unified view of data. *ACM Transactions on Database Systems*, 18(1): 9-36. doi:10.1145/320434.320440

Clivillé, V., Berrah, L., Mauris, G. (2007). Quantitative expression and aggregation of performance measurements based on the MACBETH multi-criteria method. *International Journal of Production Economics*, 105(1): 171-189. doi:10.1016/j.ijpe.2006.03.002

Curtis, B., Kellner, M.I., Over, J. (1992). Process modeling. *Communications of the ACM*, 35(9): 75-90. doi:10.1145/130994.130998

Dalal, N.P., Kamath, M., Kolarik, W.J., Sivaraman, E. (2004). Toward an Integrated Framework for modeling enterprise processes. Communications of the ACM, 47(3): 83-87. doi:10.1145/971617.971620

Doumeingts, G. (1984). *Méthode GRAI : Méthode de conception des systèmes en productique*. PhD Thesis, University of Bordeaux I, France. 519 pages (In French).

Doumeingts, G., Vallespir, B. (1995). A methodology supporting design and implementation of CIM systems including economic evaluation. In P. Brandimarte & A. Villa, Eds. *Optimization Models and Concepts in Production Management* (pp. 307-331). New-York, NY: Gordon and Breach Science Publishers.

Doumeingts, G., Ducq, Y. (2001). Enterprise modelling techniques to improve efficiency of enterprises. *Production Planning & Control*, 12(2): 146-163. doi:10.1080/09537280150501257

Harzallah, M., Berio, G., Vernadat, F.B. (2006). Analysis and modeling of individual competencies: toward better management of human resources. *IEEE Trans. on Systems, Man, and Cybernetics*, 36(1): 187-207. doi:10.1109/TSMCA.2005.859093

Harzallah, M., Vernadat, F. (2000). Validation of modelling constructs for the organisational aspects in CIMOSA. *Proc. of IFAC/IEEE/INRIA Int. Conf. on Manufacturing Control and Production Logistics* (MCPL'2000), Grenoble, France, 4-7 July 2000. CD-ROM.

Herzum, P. (2001). Web Services and Service-Oriented Architecture. *Cutter Consortium, Executive Report 4*, No. 10.

ICAM. (1981). *Integrated Computer-Aided Manufacturing (ICAM) Architecture*, Part II, Vol. IV (IDEF0), Vol. V (IDEF1), Vol. VI (IDEF2). Waltham, MA: SofTech Inc.

IFAC-IFIP Task Force. (1999). *GERAM: Generalised Enterprise Reference Architecture and Methodology*, Version 1.6.3. http://www.ict.griffith.edu.au/~bernus/taskforce/geram/versions/geram1-6-3/v1.6.3.html.

ISO 14258. (1998). *Industrial automation systems – Concepts and rules for enterprise models*. Geneva, Switzerland: International Standards Organization.

ISO 19439. (2006). *Enterprise integration – Framework for enterprise modelling*. Geneva, Switzerland: International Standards Organization.

ISO 19440. (2007). *Enterprise integration – Constructs for enterprise modelling*. Geneva, Switzerland: International Standards Organization.

ISO 31000. (2009). *Risk management: Guidelines on principles and implementation of risk management*. Geneva, Switzerland: International Standards Organization.

IVM. (2014). *The Institute of Value Management*. http://ivm.org.uk.

Jagdev, H.S., Thoben K.D. (2010). Anatomy of enterprise collaborations. *Production Planning & Control*, 12(5): 437-451. doi:10.1080/09537280110042675

Jorysz, H.R., Vernadat, F. (1990). CIM-OSA Part I: Total enterprise modelling and function view. *Int. J. of Computer Integrated Manufacturing*, 3(3): 144-167. doi:10.1080/09511929008944444

Khalaf, R., Curbera, F., Nagy, W.A., Mukhi, N., Tai, S., Duftler, M. (2005). Understanding Web Services. In M. Singh, Ed. *Practical Handbook of Internet Computing* (Chap. 27). Boca Raton, FL: Chapman & Hall/CRC Press.

Kosanke, K., Nell, J.G. (Eds.). (1997). *Enterprise Engineering and Integration: Building International Consensus*. Berlin: Springer-Verlag. doi:10.1007/978-3-642-60889-6

Kosanke, K., Vernadat, F.B., Zelm, M. (2014). Means to enable Enterprise Interoperation: CIMOSA Object Capability Profiles and CIMOSA Collaboration View, *Proc. of the 19th World Congress of the IFAC*, Cape Town, South Africa, 24-19 August 2014.

Larson, N., Kusiak, A. (1996). Managing design processes: A risk assessment approach. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 26(6): 749-759. doi:10.1109/3468.541335

Li, Q., Wang, Z., Li, W., Li, J., Wang C., Du, R. (2013). Application integration in a hybrid cloud computing environment: modelling and platform. *Enterprise Information Systems*, 7(3): 237-271. doi:10.1080/17517575.2012.677479

Mayer, R.J., Cullinane, T.P., deWitte, P.S., Knappenberger, W.B., Perakath, B., Wells, M.S. (1992). Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report. *Armstrong Laboratory*, Wright-Patterson AFB, Ohio 45433, AL-TR-1992-0057.

OMG. (2000). *OMG Unified Modeling Language Specification, Version 1.3*. Object Management Group. http://www.omg.org/spec/uml.

OMG. (2011a). *OMG Unified Modeling Language Specification (OMG UML), Version 2.4*. Object Management Group. http://www.omg.org/spec/uml.

OMG (2011b). *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, http://www.omg.org/spec/bpmn.

OMG. (2012). *Service oriented architecture Modeling Language (SoaML) Specification, Version 1.0.1*. Object Management Group. http://www.omg.org.

The Open Group. (2012). *Archimate 2.1 Specification*. The Open Group, http://pubs.opengroup.org/architecture/archimate2-doc/toc.html.

Owen, S., Walker, Z. (2013). *Enterprise Modelling and Architecture*. New Dehli, India: Ocean Media Pvt. Ltd.

Roboam, M., Zanettin, M., Pun, L. (1989). GRAI-IDEF0-Merise (GIM): integrated methology to analyse and design manufacturing systems. *Int. J. Computer-Integrated Manufacturing Systems*, 2(2): 82-98. doi:10.1016/0951-5240(89)90021-9

Ross, D.T. (1977). Structured Analysis (SA): A language for communicating ideas. *IEEE Trans. on Software Engineering*, SE-3(1): 6-15. doi:10.1109/TSE.1977.229900

Shah, L.A., Etienne, A., Siadat, A., Vernadat, F. (2014). Decision-making in the manufacturing environment using a value-risk graph. *Journal of Intelligent Manufacturing*, 25, 2. doi:10.1007/s10845-014-0895-6

Scheer, A.-W. (1992). *Architecture of Integrated Information Systems: Foundations of Enterprise Modelling*. Berlin: Springer-Verlag. doi:10.1007/978-3-642-97389-5

Scheer, A.-W. (1999). ARIS – Business Process Modeling, 2nd edition- Berlin: Springer-Verlag. 218 pages. doi:10.1007/978-3-642-97998-9

Vernadat, F.B. (1996). *Enterprise Modeling and Integration: Principles and Applications*. London: Chapman & Hall. 528 pages.

Vernadat, F.B. (1998). The CIMOSA Languages. In P. Bernus, K. Mertins and G. Schmidt, Eds. *Handbook on Architectures of Information Systems* (pp. 243-263). Berlin: Springer-Verlag. Reedited in 2006.

Vernadat, F.B. (2007a). Interoperable Enterprise Systems: Principles, Concepts, and Methods. *Annual Reviews in Control*, 31(1): 137-145. doi:10.1016/j.arcontrol.2007.03.004

Vernadat, F.B. (2007b). Reengineering the organization with a service orientation. In C. Hsu, Ed. *Service Enterprise Integration – An Enterprise Engineering Perspective* (pp. 77-101). New York, NY: Springer Science+Business Media, LLC.

Williams, T.J. (1992). *The Purdue Enterprise Reference Architecture*. Charlotte, NC: Instrument Society of America.

Wikipedia. (2013). *Enterprise Modeling*. http://en.wikipedia.org/wiki/Enterprise_modelling.

Appendix: EM construct templates

Note: The following templates have been derived from the Formal Reference Base of CIMOSA and ISO 19440:2007. "xxx" denotes a reference number that uniquely identifies each construct class.

Convention: Reference to another construct in a construct is made by <identifier/name> of the referenced construct.

Function/Process View

DOMAIN
Identifier: DM-xxx
Name: name of the domain
Description: text
Domain Objectives: Business objectives to be fulfilled by this domain
Domain Constraints: Applicable constraints
Business Processes: list [1,n] of business processes belonging to this domain
Boundary: list [1,n] of domain relationships between this domain and other functional domains (defined as binary relationships)
Object Views: list [1,n] of object views used or generated within this domain
Events: list [1,n] of events used or raised within this domain

EVENT
Identifier: EV-xxx
Name: name of the event
Description: text
Source: "External" or identifier/name of a resource or an enterprise activity
Triggers: list [1,n] of business processes
Object View: one-or-zero object view attached to the event
Predicate: Boolean expression defining the happening condition of the event
Timestamp: time at which the event occurrence will happen

BUSINESS PROCESS
Identifier: BP-xxx
Name: name of the business process
Description: text
Objectives: to be fulfilled by the process
Constraints: to be met by the process
Declarative rules: list [0,n] of business rules
Triggering events: list [0,n] of events
Process Behavior: non-empty set of
WHEN (condition) DO action rules
Ending statuses: set [1,n] of ending statuses
Comprises: list[1,n] of sub-processes or activities used in this process
Where-Used: list [0,n] of processes employing this process

ENTERPRISE ACTVITY
Identifier: EA-xxx
Name: name of the enterprise activity
Description: text
Objectives: to be fulfilled by the activity
Constraints: to be met by the activity
Declarative rules: list [0,n] of business rules
Required Capabilities: list [1,n] of capability sets defining required capabilities
INPUTS:
Function: list [0,n] of object views
Control: list [0,n] of information object views
Resource: list [1,n] of resources
OUTPUTS:
Function: list [0,n] of object views
Control: list [0,n] of generated events
Resource: information on resource status
Activity Behavior: {pre-conditions, sequence of functional opeerations, post-conditions}
Ending Statuses: set [0,n] of all possible ending statuses for this activity
Where-Used: list [1,n] od processes employing this activity

Information View

OBJECT VIEW
Identifier: OV-xxx
Name: name of the object view
Description: text
Leading Object: one enterprise object
Related Objects: list [0,n] of enterprise objects
Properties: list [1,n] of object properties in the form: property*_name: data type*

ENTERPRISE OBJECT
Identifier: EO-xxx
Name: name of the enterprise object
Description: text
Is-a: zero-or-one enterprise object
Part-of: list [0,n] of enterprise objects
Properties: list [1,n] of object properties in the form: property*_name: data type*

Resource/Infrastructure View

RESOURCE
Identifier: FE-xxx or CP-xxx
Name: name of the resource
Description: text
Class: "Functional Entity" or "Component" or "Resource Set" or "Resource Cell"
Cardinality: number of occurrences of this class
Object View: object view containing descriptive information about the resource (e.g. availability, capacity, location, RTBF, MTTR…)
Capabilities: list [1,n] of capability sets
Operations: list [1,n] of functional operations defined as: *FO-name (IN parameters, OUT parameters)*
Made-of: list [0,n] of sub-resources
Where-Used: list [0,n] of parent resources

CAPABILITY/COMPETENCY SET
Identifier: CS-xxx
Name: name of the capability set
Description: text
Capabilities:
Functional: functional capabilities
Object-related: object related capabilities
Performance: performance capabilities
Operational: operational capabilities

Competencies:
Knowledge: theoretical knowledge
Know-how: acquired skills
Know-whom: personal traits and behavior

Organization View

ORGANIZATION UNIT
Identifier: OU-xxx
Name: name of the organization unit
Description: text
Functional entity: assigned functional entity
Task description: text
Job Profile: list [1,n] of required skills/competencies
Responsibilities: list [1,n] of responsibilities
Authorities: list [0,n] of authorities
Belongs to: organization cell

ORGANIZATION CELL
Identifier: OV-xxx
Name: name of the organization cell
Description: text
Level: “Team”, “Workstation”, “Service”, “Department”, “Division”, “Direction”, Entity”
Manager: human functional entity managing the organization cell
Responsible for: list [1,n] of model components under the responsibility of the cell manager
Authority on: list [1,n] of activities, resources or enterprise objects on which the cell manager has authority with indication of the authority type
Belongs to: upper organization cell
Comprises: list [1,n] of organization units or cells belonging to this cell
Related to: list [0,n] of organization cells in relation with this cell

Collaboration View

COLLABORATION DOMAIN
Identifier: CD-xxx
Name: name of the collaboration domain
Description: text
Collaborating Partners: list [1,n] of partners
Collaborating Entities: list [1,n] of entities
Exchanged Object Views: list [1,n] of object views
Collaboration Points: list [1,n] of collaboration points

COLLABORATION PARTNER
Identifier: PC-xxx
Name: name of the collaboration partner
Description: text
Partner Identification: Legal name, Legal status, Location
Parent Entity: name of parent entity, if any
Partner Role: “Supplier”, “Service Provider”, “Retailer”, “Costumer” or “Other”
Partner ICT Environment: Description text

COLLABORATION POINT
Identifier: CP-xxx
Name: name of the collaboration point
Description: text
Collaboration Partner: collaboration partner
Collaboration Type: Asynchronous, synchronous, mutual adjustment
Exchange Flows: list [1,n] of (Sender, Receiver, Flow: list [1,n] of events and/or object views)
Exchange Media: Type of media supporting the exchange (e.g. file transfer, e-mail, HTTPS, virtual private network, courier, transportation, etc.)