



# System of Enterprise-Systems Integration Issues: an Engineering Perspective

Gérard Morel, Hervé Panetto, Frédérique Mayer, Jean-Philippe Auzelle

## ► To cite this version:

Gérard Morel, Hervé Panetto, Frédérique Mayer, Jean-Philippe Auzelle. System of Enterprise-Systems Integration Issues: an Engineering Perspective. IFAC Conference on Cost Effective Automation in Networked Product Development and Manufacturing, IFAC-CEA'07, Oct 2007, Monterrey, Mexico. pp.CDROM. hal-00168535

**HAL Id: hal-00168535**

**<https://hal.science/hal-00168535v1>**

Submitted on 21 Dec 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## SYSTEM OF ENTERPRISE-SYSTEMS INTEGRATION ISSUES AN ENGINEERING PERSPECTIVE.

Morel G.<sup>1</sup>, Panetto H.<sup>1</sup>, Mayer F.<sup>2</sup>, Auzelle J.P.<sup>1</sup>

<sup>1</sup> *Centre de Recherche en Automatique de Nancy (CRAN - UMR 7039), Nancy-University, CNRS, France  
{gerard.morel, jean-philippe.auzelle, herve.panetto}@cran.uhp-nancy.fr*

<sup>2</sup> *ERPI, ENSGSI, Nancy-University, France  
frederique.mayer@ensgsi.inpl-nancy.fr*

**Abstract** Today's needs for more capable enterprise systems in a short timeframe is leading more organizations toward the integration of existing component-systems into broader intra-organizational enterprise-systems and their integration into inter-organizational systems of enterprise-systems. Although important R&D efforts lead to a general consensus that Enterprise Modelling and IT Systems Interoperability are keys to Enterprise Integration to align the human-intensive infrastructure with the information-intensive architecture, these integration issues are not handled well in traditional systems engineering practices. Bridging the gap from an integrated system to a system of interoperable systems for architecting such broader enterprise-systems or complex systems of enterprise-systems requires 'multiple scale system thinking' to be fully comprehended beyond the 'system process' template of traditional systems engineering. This survey paper discusses some key problems, trends and accomplishments for Enterprise Integration with a systemic approach before advocating the System of Systems (SoS) paradigm to explore some rationales and forecasts in research and engineering. *Copyright © 2007 IFAC*

**Keywords:** *Enterprise Integration, Systems Interoperability, Systems Engineering, System of Systems, Enterprise Modelling.*

## 1. INTRODUCTION

Today's needs for more capable enterprise systems in a short timeframe are leading more organizations toward the integration of existing component-systems into broader intra-organizational or inter-organizational enterprise-systems. The remaining challenge of enterprise integration (EI) is to provide *the right information at the right place at the right time for decision-making* by integrating these heterogeneous information-intensive product-systems to achieve vertical business-to-manufacturing as well as horizontal business-to-business integration (Giachetti, 2004). Advances in information technologies (IT) applications interoperability facilitate implementation issues but are not efficient to support the single enterprise as well as the networked enterprise to move from tightly coupled systems based on enterprise application integration (EAI) to loosely coupled systems based on service-oriented architectures (SOA).

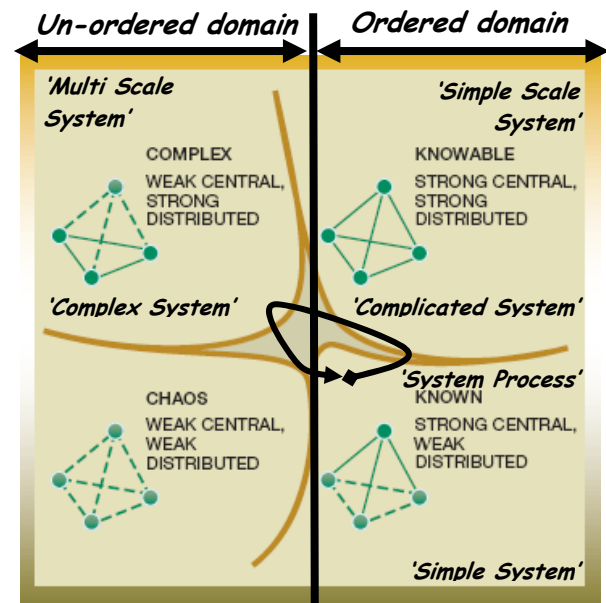
The *integration in manufacturing paradigm* (CIM concept) which underlies the global optimality of a monolithic enterprise-system fail to face this evolution, mainly because the related modelling frameworks are not appropriate to solve problems that continually change as they are being addressed. The *intelligence in manufacturing paradigm* (IMS concept) which is addressing the complexity to architect heterarchical enterprise-systems has difficulty to demonstrate its efficiency in real industrial environment (Marik et al., 2007), mainly because of the lack of a modelling framework to define, to develop, to deploy and to test self-organizing systems (Valckenaers 2003).

*Enterprise modelling* (EM) to the large meaning of the set of models necessary to manage the ontological issues related to control an organisation behaviour fails to penetrate the enterprise world despite important R&D efforts (Chapurlat, 2007). *Modelling in enterprise* based on *model-driven systems engineering* (MDSE) is not a current practice. A rough model, rather semantically poor, is built on demand for a specific project to deliver a specific solution, even if this solution is parameterized from a generic component-system. Indeed, a component-system is a combination of hardware and software but also knowledge packaged by a vendor (COTS) as a product-system to offer generic facilities. This product-system is then customized to perform specific stated purposes for a target enterprise-system to deliver its own (different) product-system. The modelling work is generally made by an external actor consulting in IT organization and the resulting enterprise-component tightly couples the parameterized application, its related engineering and the associated enterprise knowledge. Thus, enterprise-systems are supplier-intensive and each enterprise-component is not easily adaptable by the internal actors without an enterprise-system model.

These integration issues are not handled well in traditional systems engineering templates (SE) because of the increasing complexity to architect enterprise-systems as a whole for each opportunistic collaboration; this from the bottom set of heterogeneous component systems to the system of systems (SoS) that integrates them, while they continue to exist on their own missions (Maier, 1998).

We agree that *the essence of enterprise integration is the recursively interoperation of constituent systems to compose a system to achieve a specific purpose in a given context* (Carney et al., 2005).

The related interoperability relationship can be implemented in several ways to compose a fully, tightly or loosely integrated system or a SoS depending on the adaptability of the constituent systems and the assigned mission. Bridging the gap from an integrated system to a system of interoperable systems underlies knowledge-intensive organizational and cultural issues beyond technological ones requiring multi scale modelling frameworks to cope with the limitations of human abilities to face complexity (Bjerkemyr et Lindberg, 2007). Understanding the partition between the ordered (directed) domain of what is known or knowable and the un-ordered (emergent) domain of what can be patterned from interactions perceptions is of importance to engineer a system of enterprise-systems with a SoS-like perspective (Fig. 1).



**Fig. 1. Components connection strength of Cynefin domains (Kurtz and Snowden, 2003)**

While the *SoSE paradigm* is still in its infancy, some frameworks are being developed for decision-making and cost-effective management issues to face the emerging SoS problems (Gorod et al., 2007). Nevertheless, pragmatic issues cope with unclear bottom (bottom-up, top-down) approaches to point that the fundamental template of any SE project must progress from the top down to satisfy the intended mission of a

unique target-SoS, even if bottom-up engineering decisions are made to reuse existing systems (Cocks, 2006).

All these considerations remain research and development open issues pointing that the notion of system must be revisited in a *multi scale system thinking* to be fully comprehended (Kuras and White, 2005) beyond the *system process* and the *single scale system thinking* templates of traditional systems engineering.

This paper overviews in section 2 some Enterprise integration issues (Vernadat, 2007) based on the knowledge gained in major European projects related to interoperability (Panetto, 2007) (Doumeights *et al.*, 2007). Section 3 deals with our main results and ongoing research works to explore some SoSE aspects of product-driven systems control by prototyping an IT architecture in which the product itself is active to interoperate with enterprise-systems to ensure a B2M2B service (Fig. 2). Industrial transfers of this approach exhibit complementary SoS properties and improve this partial understanding of why should be made for better SoSE practices in EI. Thus, conceptualization trends and open issues are discussed in section 4 before to conclude.

## 2. ENTERPRISE INTEGRATION ISSUES

Integration is generally considered to go beyond mere interoperability to involve some degree of functional dependence. While interoperable systems can function independently, an integrated system loses significant functionality if the flow of services is interrupted. An integrated family of systems must, of necessity, be interoperable, but interoperable systems need not be integrated. Integration also deals with organisational issues, in possibly a less formalised manner due to dealing with people, but integration is much more difficult to solve, while interoperability is more of a technical issue.

Compatibility is something less than interoperability. It means that systems/units do not interfere with each other's functioning. But it does not imply the ability to exchange services. Interoperable systems are by necessity compatible, but the converse is not necessarily true. To realize the power of networking through robust information exchange, one must go beyond compatibility.

In sum, interoperability lies in the middle of an "Integration Continuum" between compatibility and full integration. It is important to distinguish between these fundamentally different concepts of compatibility, interoperability, and integration, since failure to do so, sometimes confuses the debate over how to achieve them. While compatibility is clearly a minimum requirement, the degree of interoperability/integration desired in a joint family of systems or units is driven by the underlying operational level of those systems.

### 2.1. Enterprise-system architecting

The problems of enterprise applications interoperability can be defined according to various points of view and perspectives. These aspects correspond to modelling frameworks, with, as a common point, an implicit or explicit perspective of evolution according to a linear scale the more an application is interoperable with another and thus higher in a value scale, the more it relates to a high level of abstraction of the models and their semantics. For this reason, an interoperability development process is often classified in so-called "levels of interoperability" in the literature.

A widely recognized model for information systems interoperability is, 'Levels of Information Systems Interoperability' (LISI) (C4ISR, 98). LISI focuses on the increasing levels of sophistication of systems interoperability (Isolated systems, connected interoperability in a peer-to-peer environment, Functional interoperability in a distributed environment, Domain based interoperability in an integrated environment; Enterprise based interoperability in a universal environment. LISI focuses on technical interoperability and the complexity of interoperations between systems. The model does not address the environmental and organizational issues that contribute to the construction and maintenance of interoperable systems. Acknowledging this limitation, Clark and Jones (1999) proposed the Organizational Interoperability Maturity model (OIM), which extends the LISI model into the more abstract layers of command and control support (Independent, Ad hoc, Collaborative, Integrated, Unified). Beyond this organisational interoperability, the type of content of the exchange flows is also an issue. To cope with it, the NATO C3 Technical Architecture (NC3TA) Reference Model for Interoperability (NATO, 2003) focuses on technical interoperability and establishes interoperability degrees and sub-degrees (Unstructured Data Exchange, Structured Data Exchange, Seamless Sharing of Data, Seamless Sharing of Information). The degrees are intended to categorize how operational effectiveness could be enhanced by structuring and automating the exchange and interpretation of data.

Moreover, at a conceptual level, Tolk (2003) has developed the Levels of Conceptual Interoperability (LCIM) Model that addresses levels of conceptual interoperability that go beyond technical models like LISI. Systems interoperability is not only a technical problem (as stated by LISI or LCIM) but also deals with organisational issues (OIM). These aspects of interoperability are coherent with the definitions proposed by the European Interoperability Framework (EIF, 2004), which considers three aspects of interoperability (Organisational, Semantic, Technical).

While choosing a framework is a necessary condition for facilitating interoperability engineering because they are representing best practices in the domain of interoperable systems engineering.

## 2.2. Enterprise-system modelling

*Enterprise modelling* is the set of activities or processes used to develop the various parts of an enterprise-system model to address some desired modelling finality. It can also be defined as the art of “externalising” enterprise knowledge, i.e. representing the enterprise in terms of its organisation and operations (e.g. processes, behaviour, activities, information, object and material flows, resources and organisation units, and system infrastructure and architectures). The finality is to make explicit facts and knowledge that add value to the enterprises or can be shared by business applications and users. The prime goal of enterprise-system modelling is not only to be applied for better enterprise integration but also to support analysis of an enterprise, and more specifically, to represent and understand how the enterprise works, to capitalize acquired knowledge and know-how for later reuse, to design (or redesign) a part of the enterprise, to analyse some aspects of the enterprise (by e.g. economic analysis, organization analysis, qualitative or quantitative analysis,...), to simulate the behaviour of (some part of) the enterprise, to make better decisions about enterprise operations and organization, or to control, coordinate and monitor some parts of the enterprise.

The intensive production of tools, implementing various different enterprise-system modelling languages, has led to a *Tower of Babel* situation in which the many tools, while offering powerful and distinct functionalities, are unable to interoperate and can hardly or not at all communicate and exchange models. This is a serious drawback for awareness, acceptance and wide use of the EM technology since enterprises cannot capitalise from previous modelling efforts. This situation hinders true enterprise integration, interoperability, and sharing enterprise knowledge.

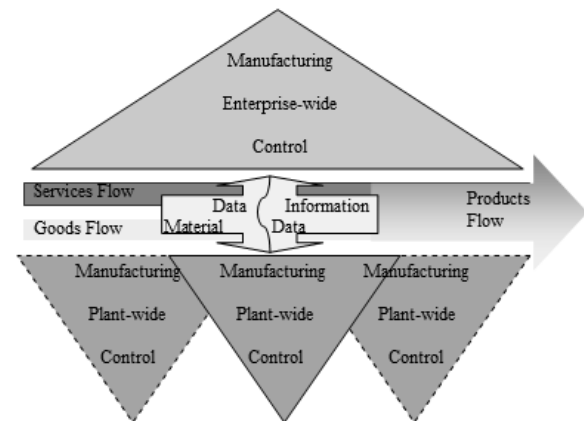
Enterprise-system Modelling is an *engineering discipline* closely related to computerised systems. As such, it requires the combined use of Enterprise Modelling Software Environments (EMSE), Enterprise Modelling Languages (EML), and Enterprise Engineering Methodologies (EEM). According to this point of view, there exist a lot of fragmented approaches to enterprise modelling (including Methodologies, Languages and Tools). Enterprise-system modelling approaches may also have very different objectives and needs.

Moreover Enterprise-system models are dependent of the systems engineering that produces them. This engineering includes best practices related to a specific domain that fail to take advantage to the enterprise modelling world-wide research. While standard initiatives and European R&D projects are trying to solve the problem of managing heterogeneous information coming from different systems by formalising the technical knowledge related to products

technical data and their processes, they are not yet dealing with reference enterprise-system modelling engineering, key drawback for loose integration. With the support of a product-driven IT architecture, we will revisit some definitions related to systems, system-of-systems, and systems engineering through the conceptualisation of these concepts in order to reach a common understanding on its semantics.

## 3. SYSTEM OF ENTERPRISE-SYSTEMS CASE-STUDY

Important research issue is that the complexity of SoS-like designs makes formal proof of their performance and capabilities impractical. Research developments must be tested in emulated enterprise-systems reflecting the size and complexity of the real world, beyond the toy test cases. Our focus is to prototype a product-driven IT architecture to conceptualize system integration issues for research purposes in order to improve SoSE practices for industrial transfer issues. The main modelling concept is to make the product interactive as the ‘controller’ of the manufacturing enterprise’s resources for enabling ‘on the fly’ interoperability relationships between existing product-systems and ensuring coherence between the physical and information flows all through the product life-cycle (Fig. 2).



**Fig. 2. Product-driven manufacturing enterprise-wide control (Morel et al., 2003)**

### 3.1. B2M2B case-study

This IT architecture is composed of a set of enterprise systems (PDM, ERP, MES, and SFC) distributed on two sites in Italy and France. This case-study emulates a networked product development and manufacturing virtual enterprise whose objective is to design, produce and sell products made from assembly of elementary components. The prototype will take advantage of new IT technologies and architecture such as SOA (Service Oriented Architecture), web services, and embedded systems. The active products (smart products) are, by definition (Wong et al., 2007), loosely coupled to maintain resilient relationships with the ERP-MES-PDM-SFSC systems by synchronizing transactions

*The studied enterprise-system is the ‘system of interest’ related to a new mission to ensure the product traceability for which an architectural solution must be prototyped from the existing enterprise-systems.*

Our approach is a particular interpretation of the *HMS paradigm* (Morel et al., 2003) aiming to implement the interoperability relationship at a MES level as a product-holon (a system) to ensure a recursive interoperation between the systems (ERP, PDM) which process the services (information) and the resources (SFC) which transform the goods.

The proposed model-driven approach for modelling product information, along its lifecycle, is to ensure the completeness of such model with regard to the various networked enterprises applications that will interoperate with it. The first assumption of our work is a necessary synchronous modelling of product material views and product informational views (Baïna, et al, 2008). Indeed, in a manufacturing enterprise environment, information and material flows are often considered separately, however information is always referring to a physical exchange and, conversely, material flows are always accompanied with some information. Throughout its lifecycle, the product evolves in a dual way, a physical product that is subject of physical processes, and an informational product that combines all information collected about product states during its lifecycle and product information needed for production control, quality assessment and traceability management. Moreover, our aim is more to simplify the modelling of different aspects of the product (information and material) by considering the aggregation of information about the product and the conceptual view all applications may have about it. 'Information contents' are perceived as holons (whole and part) at the 'scope row' of the Zachman framework (Zachman, 1987) (Sowa and Zachman, 1992) independently of their nature (data, event, material ...) to identify the core processes perform by the business and where they operate. Our adaptation of the Zachman framework is defined by a sequence of artefacts represented by a specific cell in matrix. The transition from one cell to one other is based on syntactic transformations or semantics mapping between models (Baina 2006).

```

classDiagram
    class PhysicalPart {
        <<empty>>
    }
    class Holon {
        <<empty>>
    }
    class InformationalPart {
        <<empty>>
    }
    class Attribute {
        <<empty>>
    }
    class ComplexHolon {
        <<empty>>
    }
    class ElementaryHolon {
        <<empty>>
    }
    class Property {
        <<empty>>
    }
    class State {
        <<empty>>
    }
    class AttributeValue {
        <<empty>>
    }
    class ProcessInstance {
        <<empty>>
    }
    class Process {
        <<empty>>
    }
    class Resource {
        <<empty>>
    }
    class Capability {
        <<empty>>
    }

    PhysicalPart "0..1" -- "1" Holon : is part of
    Holon "1" -- "0..1" InformationalPart : describes
    Holon "1..*" -- "1..*" Attribute : has attributes
    Holon "1..*" -- "0..1" ComplexHolon : is composed of
    ComplexHolon "0..1" -- "1..*" State : state
    Holon <|-- ComplexHolon : (Complete, Disjoint)
    Holon <|-- ElementaryHolon : (Complete, Disjoint)
    Holon <|-- State : (XOR)
    State "1..*" -- "0..1" State : state
    State "1..*" -- "1..*" AttributeValue : defined by
    State "1..*" -- "1..*" PropertyValue : defined by
    AttributeValue "1..*" -- "1..*" Attribute : concerns
    PropertyValue "1..*" -- "1..*" Property : concerns
    ProcessInstance "1..*" -- "1..*" Process : instance of
    ProcessInstance "1..*" -- "1..*" Resource : uses
    Process "1..*" -- "1..*" Capability : needs
    Resource "1..*" -- "1..*" Capability : provides
  
```

The diagram illustrates the relationships between various concepts in the OSW ontology. Key relationships include:
 

- Holon** is a base class for **Complex Holon**, **Elementary Holon**, and **State** (with a disjoint relationship).
- Physical Part** (0..1) is part of **Holon** (1).
- Holon** (1) describes **Informational Part** (0..1).
- Holon** (1..\*) has attributes **Attribute** (1..\*).
- Holon** (1..\*) is composed of **Complex Holon** (0..1).
- Complex Holon** (0..1) is in a state **State** (1..\*).
- State** (1..\*) is in a state **State** (0..1).
- State** (1..\*) is defined by **Attribute Value** (1..\*) and **Property Value** (1..\*).
- Attribute Value** (1..\*) concerns **Attribute** (1..\*).
- Property Value** (1..\*) concerns **Property** (1..\*).
- Process Instance** (1..\*) is an instance of **Process** (1..\*).
- Process Instance** (1..\*) uses **Resource** (1..\*).
- Process** (1..\*) needs **Capability** (1..\*).
- Resource** (1..\*) provides **Capability** (1..\*).

<sup>1</sup> MEGA Suite, MEGA International, [www.mega.com](http://www.mega.com)

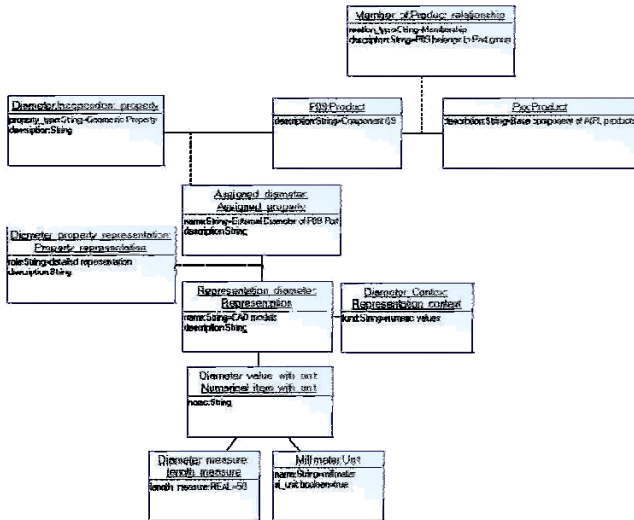


Fig. 4. PDM model instantiation

We are formalising this information model as a *Product Ontology*, thus including domain heuristics, able to express and share product knowledge among systems (Tursi, et al, 2007). This *explicit specification of our shared conceptualization* (Gruber, 1993) allows the formalization of the semantics of objects, formalizing and identifying the modelling concepts, their dynamic behaviour and discovering their inter-relative mappings (Fig. 5), through FOL rules.

The increasing performance of the infotonics technology (agent on RFID) is expected making a concrete sense in near future to our *product-driven control paradigm* as a contribution to the *paradigm the order is the product* for product oriented manufacturing systems for reconfigurable enterprises based on distributed automation<sup>2</sup>.

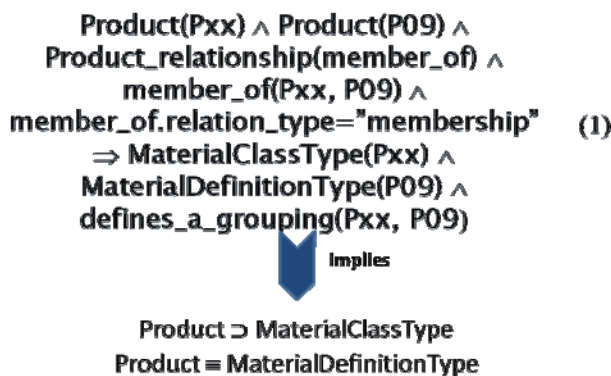


Fig. 5. Mapping discovery through DL Lite formalisation

#### 4. DISCUSSION ON SYSTEM-OF ENTERPRISE-SYSTEMS ISSUES

These above sections show that enterprise-systems are reaching a certain size and level of complexity, so that

traditional modelling and engineering approaches are not sufficient for addressing all relevant issues (ordered domain of figure 1).

The component-systems that constitute a system-of enterprise-systems were not designed to work together; each component-system may be of different age and technology; the system of systems has no clear lead system that is a system that will oversee the integration of the systems.

The notion of system, central to all the above paradigms and to systems engineering for practical purposes, must be revised to accept the idea that exploring the un-ordered (emergent order) domains (Waldrop, 1992) is a way to support decision-making in the varied dynamical contexts of SoSE when moving from strong central connections to weak ones (Fig. 1, from right to left).

##### 4.1. Characterization issues

Many definitions are being amended with a number a required properties (Sage and Cuppan, 2001) (Fisher, 2006) to make SoS a candidate rationale artefact to distinguish a very large and complex socio-technical system of interoperable enterprise-systems from a monolithic non-SoS (Carlock and Fenton, 2001).

Although their socio-technical complexity seems smaller than for systems commonly exemplified as SoS, we explored this promising artefact on four industrial case studies for enterprise information-intensive systems modelling issues in order to improve our understanding of why these Enterprise-systems seems exhibit SoS properties and how their SE processes should evolve.

These observations fit more or less with the five Maier's criteria related to the basic qualitative traits of SoS and the three derivative implications proposed by (DeLaurentis *et al*, 2006). But the key question is that the limitations of human abilities when facing complexity could be the source of the observed SoS properties for this class of problems. (Bjelkemyr et Lindberg, 2007). On one hand, selected architectures for each solution result rather of an ad-hoc adjustment of the requirements to stay within constraints of the existing family of systems according to the *SoS process* vision of (Cocks, 2006). On another hand, some exhibited emergent behaviours underlie that several capabilities have been left unsatisfied and that the *SoS thinking* to share the interoperation between the existing systems is beyond the traditional template.

Another key question is the resilient attribute of the product-system designed, manufactured and delivered by a system-of enterprise-systems which should be by itself resilient to disruptions such as cost-effective pressures. Among many traits close to those of SoS, the system resilience is dependent of the largely human-intensive infrastructure to be able to function as a

<sup>2</sup> www.pabadis-promise.org



whole and of the information management system to be architected as a whole (Jackson, 2007). Emergence (emergent behaviour) is then perceived as an unexpected characteristic of a SoS and must be checked by preserving the interoperability relationships between the component-systems for adaptability issues. That could be the role of the active product in our case-study to re-synchronise on the fly the connectivity between the component-systems considered as resources for each product-system mission.

These considerations clearly underscore that these SoS-like are beyond just complicated systems and consequently their engineering beyond traditional templates. In advance of a practical SoSE framework, our first action of transfer to industry is the development of perennial partnerships based on the recruitment of doctors-engineers trained on MDSE to increase systemic expertise. This cultural issue is generally admitted as a basic to cope with complex-systems engineering as exhibited by SoS-like problems.

#### 4.2. Conceptualization issues

Previous works (Mayer, 1995) conceptualized on the basis of the General Systems Theory (Simon, 1990) that a system is the normal (positive) result of emergence within any SE process leading to an ad-hoc solution based on heuristics and normative guidelines such as ISO/IEC 15288<sup>3</sup>. The engineered-system as result of an engineering-system is a part

- emerging from a contextual whole which is the atomic relationship between the user-system (the environment) and the delivered product-system (the desired finality) ;
- Growing as a molecule by refining the successive relationships in guided directions with regards to the purpose to achieve.

The concept of design patterns was initially proposed by (Alexander, 1979). Later, this concept moved into the field of software engineering for the design of new applications reusing generic structures (Coad, 1992). In particular, the patterns were proposed after observing that a great number of software applications were based on the same principles. Their knowledge thus leads to significantly reduced design effort.

Nowadays, the main design patterns are gathered in the reference catalogue of (Gamma et al., 1995). The catalogue describes 23 fundamental models to build the architecture of complex systems. It capitalizes a major part of the know-how acquired from object-oriented software engineering by describing how to reuse structural and behavioural patterns. These patterns are then used to design systems that are more intelligible, more flexible and above all more generic.

Buschmann et al. (1996) present also a set of architectural patterns. This catalogue describes the

styles of organization and interaction at a higher level of abstraction—by presenting layered architectures, for example.

The description of a design pattern includes

- A name, to identify the pattern and to extend the vocabulary used to describe the architecture.
- An intention, to summarize what the design pattern does with the problem solved and the conditions of use.
- A context (or motivation), to illustrate how the components of the pattern respond to the design problem.
- A detailed description of the pattern, with a graphical representation of its structure, a specification of the different components, directions for use, particularities of the pattern.
- A range of applicability, to present the situations in which the design pattern can be applied.
- An example of use.

Today, the concept of design pattern moves from software engineering to other domains business systems, information systems, control systems (Cloutier and Verma, 2007)). Coplien and Schmidt (1995) propose a set of design patterns dedicated to these particular domains. The convergence between software engineering and control engineering is possible because the models and the processes considered are described at a high level of abstraction everything is modelled by objects (processes, controllers, interactions, etc.). (Shaw, 1995), (Fischwick, 1996) and (Sanz and Zalewski, 2000) have shown several examples which illustrate this convergence.

Our going works aim to approximate the content of this conceptualisation by intuitively selecting the *composite pattern* (Fig. 6)) and by deriving it to find similarities and to add contextual details (semantics) at each step of a MDSE process (Fig.7).

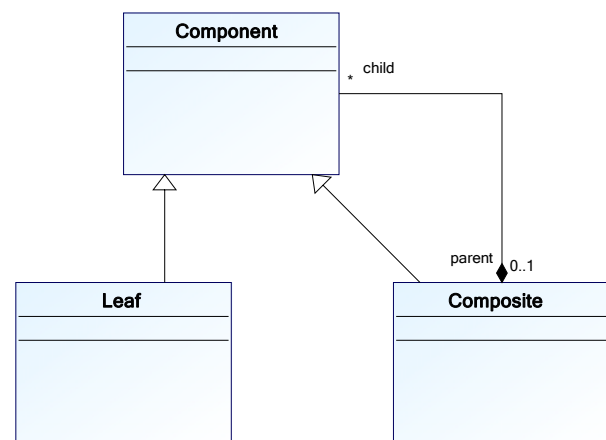
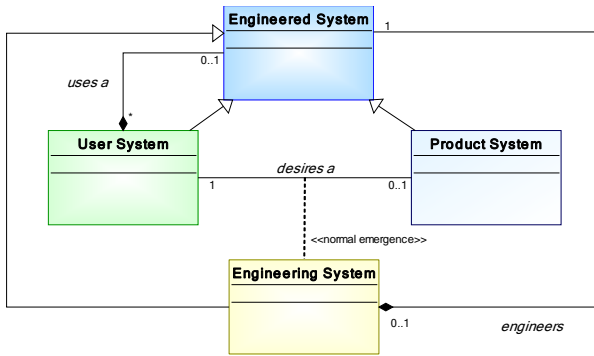


Fig. 6. The composite pattern

<sup>3</sup> www.incose.org





**Fig. 7. Normal emergence of a system**

This fundamental *composite pattern* reveals that, recursively, an element (the *composite* called here user-system and then engineering-system) may relate to other elements (the *components* called here engineered-system) in a hierarchical relationship. The recursive relationship will stop when reaching an elementary element (the *leaf*, called here the product-system). The common semantics of this kind of relationship is *composed of* but, when using this pattern, this relationship may be adapted to the context of the system to be modelled.

For example, the architecture of the studied SoS-like is approximated to this *composite pattern* to which the added semantics to the initial derived pattern classifies it as a kind of “loosely coupled system”, which is an aggregation of other component systems. These components may be either, recursively, other loosely coupled systems or tight couple systems till one reach a leaf in this recursive relationship. This leaf is a fully integrated system that is considered as an elementary element where components, if they exist, are completely hidden. Coming back to some definitions, a tightly coupled system is a system composed of elementary elements, visible but so tightly linked together that the modification of one of them may cause trouble to the high order system that embeds it.

By definition, this relationship is recursive (even fractal) as any system is produced by another higher system, answering specific requirements. For a dedicated project, the target system is the final produced system, in this recursively loop.

Deriving the “composite” design pattern results then to this model, on the right side of the figure 8, where a project system produces a target system by putting in place systems engineering practices to transform initial requirements to technical ones, and thus defining what the target system is.

The challenge for the studied SoSE-like relies with *weak emergence* (Bedau, 1997) to perceive (in the unordered domain), to pattern and to check (in the ordered domain) added behaviours due to the interactions between the component-systems. This must be led by extensive SysML-driven requirements

analysis (System Modelling Language<sup>4</sup>) adding more details than current practices and experiments such as HMS/MAS simulation to track self-organizing patterns in order to improve components-systems adaptability.

Another analogy in search of the biology of systems has been recently proposed by (Sausser and Boardman, 2007); they visit the notion of *Holon* (Koestler, 1967), that which is both whole and part simultaneously, in order to make stark contrasts between systems (of parts) and SoS.

This dualism has been previously addressed by (Kuras, 2005) to make explicit the role of the environment for a multi scale definition of a system

$$S \equiv \{\text{pattern}_c\}; H$$

This expression means that a system,  $S$ , is a set  $\{\text{pattern}_c\}$  along with something called a *Holon* considered as a conceptualization operator to select, aggregate and distinguish the subset of relevant patterns of  $S$  available at this scale of conceptualisation.

$\text{Pattern}_c$  makes up the content our conceptualization approximated by the composite pattern to which detailed are added (Fig. 3) to make the active product possibly a system for *product-centric connectivity*. This composite pattern is further explored to define the SE process for MDSE purposes (Fig. 8). Our intent is to point out that theses SoSE processes meet those of the HMS community on the design of complex emergent systems (Valckenaers et al., 2003). Such model may be considered as a SoIS pattern (System of Interoperable Systems) that may be instantiated and derived for specific modelling purpose. The resulting specific model then provides elicitation of systems interfaces, classifying systems components through a generic upper ontology and defining their interoperation relationships.

Similarly, the SE process is modelled with the composite, transforms initial requirements, coming from a client, to technical requirements and constraints that define a target system to be produced.

#### 4.3. Architecting and modelling issues

The conceptualization of both the engineered and engineering systems needs their models to be fully coherent. Indeed, the recursive relationship as stated previously allows the specification of these systems with different refinements levels. Each of those refinement levels is then defined taking into account different points of view: from stakeholders to systems engineers ones. Each of these views specifies either a component of the engineered system, or a composite part of it. These many views must then be guided

<sup>4</sup> OMG SysML, System Modeling Language, [www.sysml.org](http://www.sysml.org)

though best practices, formalized into a modelling framework. The Zachman framework (ZFW) (Zachman, 1987) (Sowa and Zachman, 1992) meets our modelling requirements because of its multi-views and multi-abstraction levels definitions. Indeed, this framework has theoretical foundations in the domain of Model-Driven Engineering (MDE) (McGovern, et al. 2003), domain that mainly suggests best practices and practical tools for modelling a system through models transformation, mapping and linking. This engineering process, involving different actors, from systems specification to design and implementation stages is obviously recursive. This implies composite and structured definitions along with coherent models transformation.

As already presented in recent work (Panetto, et al. 2007), the Zachman framework may adapted and derived to take into account, recursively, many points of view. One single Zachman matrix is not sufficient to model a whole complex engineered system together with its engineering one. Better, when such system is composed of sub-systems and when, moreover, it may also be considered as a multi scale SoIS, the recursiveness of the framework (Fig. 9), based on the *composite pattern*, ensures, by design, the effective coherence between systems models in a SoS-like context.

The engineering system (project system) finality is to derive the composite pattern and to instantiate generic models (including generic frameworks), for a specific project. The resulting instantiated models are then recursively embedded into Zachman frameworks instances (Fig. 10). Our going work aims to explore the recursive composite loop. This loop is ending when from an engineered system view, the framework specifies the basic components (fully integrated systems) models (Fig. 9). To be coherent with SE practices, this last level of conceptualization instantiates the Zachman framework embedding models based on SysML, a UML profile based on UML2 (Fig. 11).

However, while the recursive composite loop allows, from an engineering system view, to simplify the human understanding of SoS-like hierarchical models, it also emphasizes the inherent complexity of such SoS-like, from an engineered system view. In order to border this complexity, one of the main research challenges concerns the quantitative measure of SoS-like complexity related to the connectivity property between enterprise-systems (Coll-Plexity European project<sup>5</sup>) (Schuh, et al., 2006) and the level of interoperability between enterprise-systems basic components (Ford, et al., 2007).

## 5. CONCLUSION

The focus of this paper is mainly the evolution of Enterprise Integration as the evolution of the interoperation complexity between existing enterprise and components systems architected as a SoS-like to achieve a specific purpose in a given context. Current approaches to interoperability for Enterprise Integration issues are examined as solutions to combine and implement enterprise-systems relationships. Nevertheless, these solutions are not sufficient when the number of relationships expand because of ad-hoc interfaces that do not take into account existing standards.

The enterprise system that we considered in the paper is the system of interest. In a system-of-systems, the number of possible combinations of interactions among the systems is theoretically infinite. System “unravelling” have an intelligence of their own as they expose hidden connections, neutralize redundancies, and exploit chance circumstances for which no system engineer might plan. In this paper, we propose a new paradigm for system-of-systems design.

Rather than decomposing each system within the system-of-systems in a functional fashion, we treat the system-of-systems as a single entity that is comprised of abstract classes.

We demonstrate how our paradigm can be used both to avoid the introduction of accidental complexity and to control essential complexity by applying object-oriented concepts of decentralized control flow, minimal messaging between classes, implicit case analysis, and information-hiding mechanisms. We argue that our paradigm can aid in the creation of sound designs for the system-of-systems in contrast to creating a federation of systems through a highly coupled communication medium.

---

<sup>5</sup> Coll-Plexity FP6 STREP N° 12781 (Collaborations as Complex Systems)

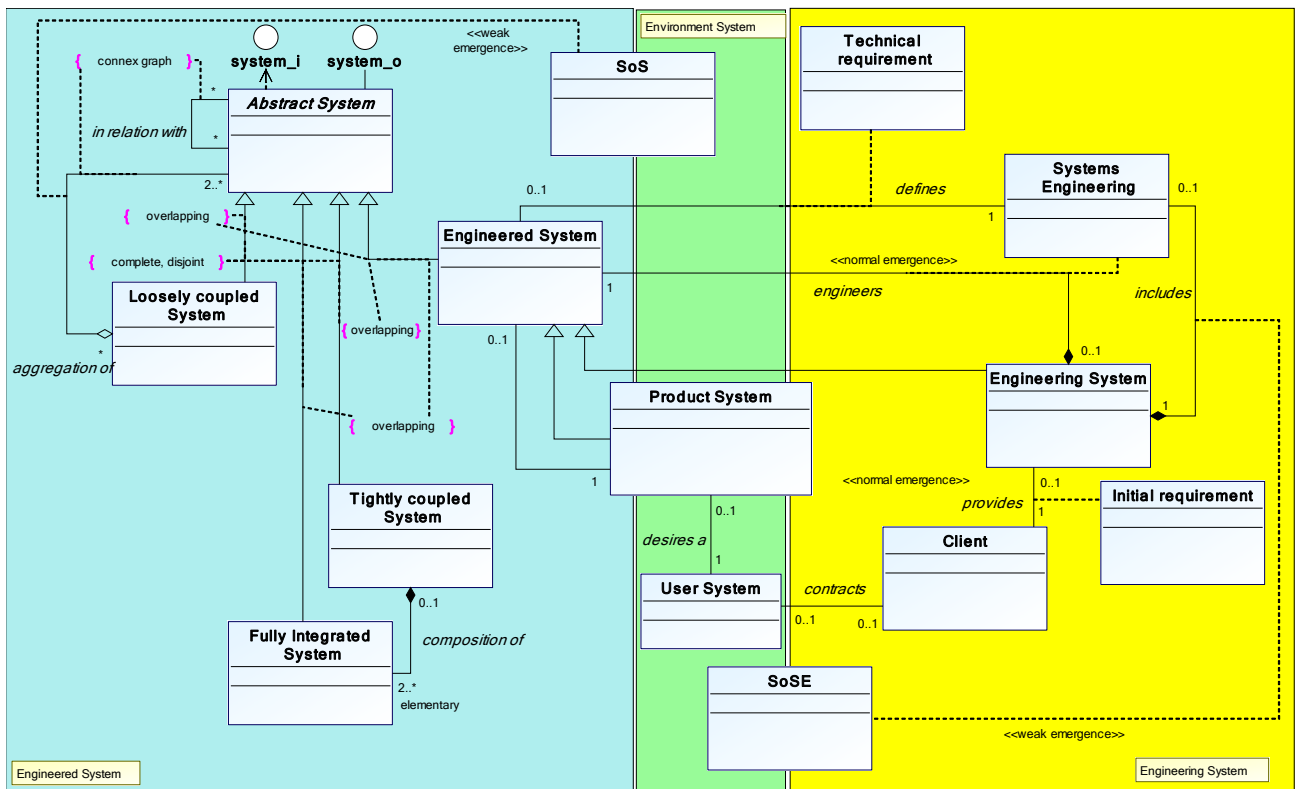


Fig. 8. Systems Engineering model-driven pattern

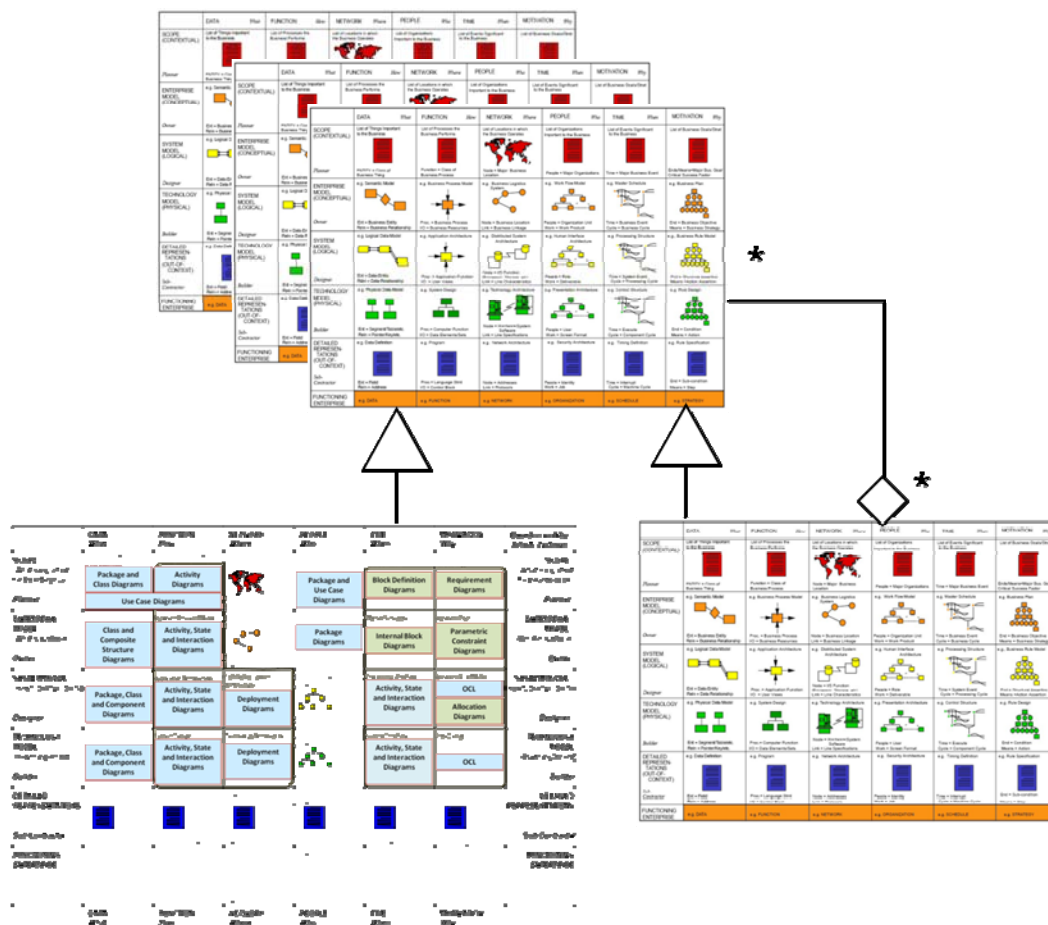


Fig. 9. Composite pattern-based recursive frameworks

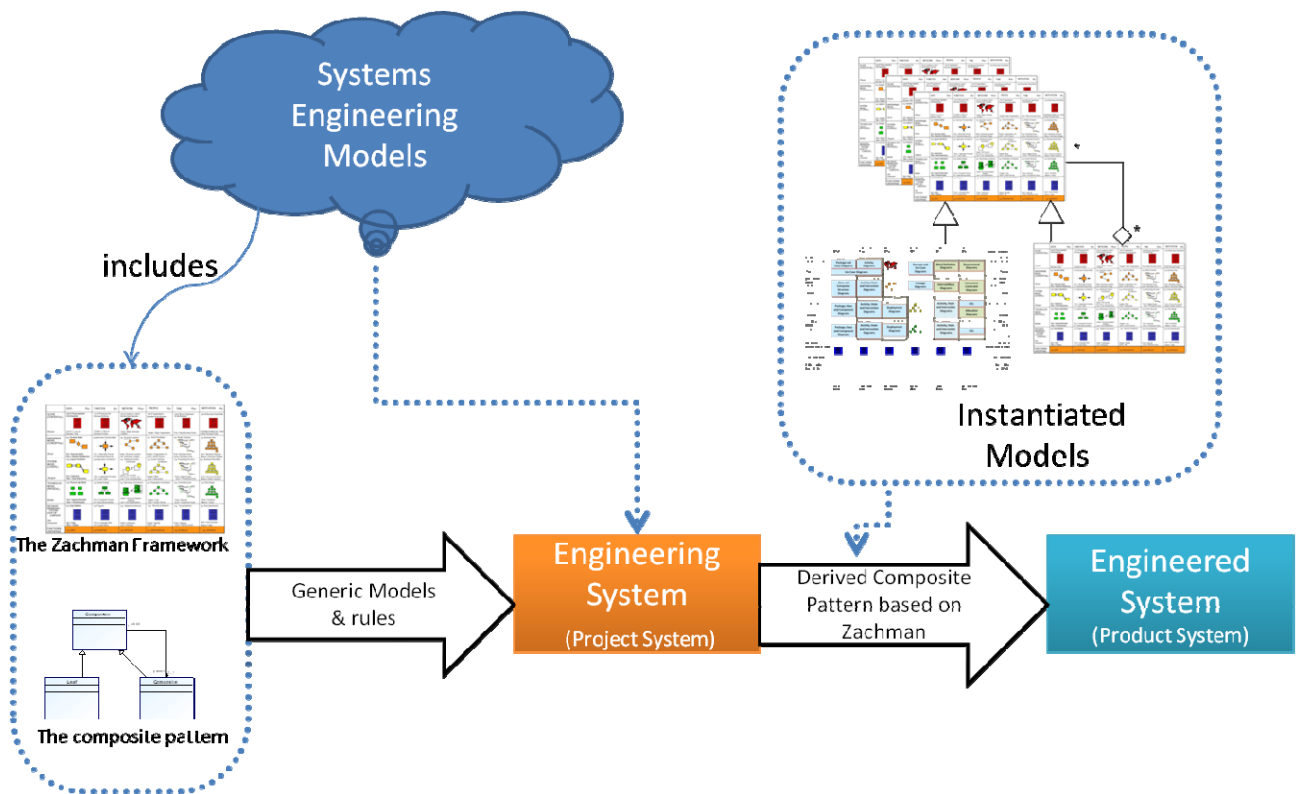






Fig. 10. From generic models and frameworks to the recursive modelling hierarchy

|   | DATA<br><i>What</i>                             | FUNCTION<br><i>How</i>                          | NETWORK<br><i>Where</i>  | PEOPLE<br><i>Who</i>   | TIME<br><i>When</i>                            | MOTIVATION<br><i>Why</i>             | Based on work by<br>John A. Zachman                       |
|---|---|---|--|--|--|--------------------------------------|---|
| <b>SCOPE</b><br>(What is important<br>for the enterprise) | Things Important<br>to the Business             | Processes Performed                             | Business Locations   | Important Organizations  | Events Significant<br>to the Business          | Business Goals<br>and Strategy       | <b>SCOPE</b><br>(What is important<br>for the enterprise) |
| <i>Planner</i>  | Package and<br>Class Diagrams                   | Activity<br>Diagrams                            | <br>Node = Major<br>Business Locations                  | Package and<br>Use Case<br>Diagrams  | Block Definition<br>Diagrams                   | Requirement<br>Diagrams              | <i>Planner</i>  |
| <b>ENTERPRISE<br/>MODEL</b><br>(What is available)        | Semantic Model                                  | Business Process Model                          | Business Logistics<br>System   | Work Flow Model  | Master Schedule                                | Business Plan                        | <b>ENTERPRISE<br/>MODEL</b><br>(What is available)        |
| <i>Owner</i>  | Class and<br>Composite<br>Structure<br>Diagrams | Activity, State<br>and Interaction<br>Diagrams  | <br>Node = Business Location<br>Link = Business Linkage | Package<br>Diagrams  | Internal Block<br>Diagrams                     | Parametric<br>Constraint<br>Diagrams | <i>Owner</i>  |
| <b>SYSTEM MODEL</b><br>(How to build products)            | Logical Data Model                              | Application Architecture                        | Distributed System<br>Architecture   | Human Interface<br>Architecture  | Processing Structure                           | Business Rule Model                  | <b>SYSTEM MODEL</b><br>(How to build products)            |
| <i>Designer</i>   | Package, Class<br>and Component<br>Diagrams     | Activity, State<br>and Interaction<br>Diagrams  | Deployment<br>Diagrams   | <br>People = Role<br>Work = Deliverable   | Activity, State<br>and Interaction<br>Diagrams | OCL<br>Allocation<br>Diagrams        | <i>Designer</i>   |
| <b>TECHNOLOGY<br/>MODEL</b><br>(How to Implement)         | Physical Data Model                             | System Design                                   | Technology Architecture  | Presentation<br>Architecture   | Control Structure                              | Rule Design                          | <b>TECHNOLOGY<br/>MODEL</b><br>(How to Implement)         |
| <i>Builder</i>  | Package, Class<br>and Component<br>Diagrams     | Activity, State<br>and Interaction<br>Diagrams  | Deployment<br>Diagrams   | <br>People = User<br>Work = Screen Format | Activity, State<br>and Interaction<br>Diagrams | OCL                                  | <i>Builder</i>  |
| <b>DETAILED<br/>REPRESENTATIONS</b>                       | Data<br>Definition                              | Program   | Network<br>Architecture  | Security<br>Architecture   | Timing<br>Definition                           | Rule<br>Design                       | <b>DETAILED<br/>REPRESENTATIONS</b>                       |
| <i>Sub-Contractor</i>                                     | Ent = Field<br>Rel = Address                    | Proc = Language Statement<br>IO = Control Block | Node = Addresses<br>Link = Protocols   | People = Identity<br>Work = Job  | Time = Interrupt<br>Cycle = Machine Cycle      | End = Sub-Condition<br>Means = Step  | <i>Sub-Contractor</i>                                     |
| <b>FUNCTIONING<br/>ENTERPRISE</b>                         | Data  | Function  | Network  | Organization   | Schedule                                       | Strategy                             | <b>FUNCTIONING<br/>ENTERPRISE</b>                         |
|   | Ent =<br>Rel =                                  | Proc =<br>IO =                                  | Node =<br>Link =   | People =<br>Work =   | Time =<br>Cycle =                              | End =<br>Means =                     |   |
|   | <b>DATA</b><br><i>What</i>                      | <b>FUNCTION</b><br><i>How</i>                   | <b>NETWORK</b><br><i>Where</i>   | <b>PEOPLE</b><br><i>Who</i>  | <b>TIME</b><br><i>When</i>                     | <b>MOTIVATION</b><br><i>Why</i>      |   |

All SysML  
Diagrams

UML2  
Diagrams

SysML  
extended  
Diagrams

Fig. 11. UML2 and SysML mapped onto the Zachman framework

## REFERENCES

- Alexander C. (1979). *The Timeless Way of Building*, Oxford University Press, New York
- Baïna, S. (2006). Model-driven interoperability product oriented approach for enterprise-systems interoperability. PhD Thesis, Henri Poincaré, Nancy University (in French).
- Baina S., Panetto H., Benali K. (2008). Product oriented modelling and Interoperability issues. In Manolopoulos, Y.; Filipe, J.; Constantopoulos, P.; Cordeiro, J. (Eds.) *Enterprise Information Systems VIII, Lecture Notes in Business Information Processing*, Vol. 3, February, Springer, Berlin, ISBN 978-3-540-77580-5
- Bedau M. (1997) Weak emergence, philosophical perspectives mind, causation and world. Vol. 11, Blackwell Publishers.
- Bjelkemyr et al. (2007) An engineering systems perspective on system of systems methodology, 1st Annual 2007 IEEE Systems Conference, Hawaii
- Buschmann R., Meunier H., Rohnert P., Sommerland M. (1996). *Pattern-oriented Software Architecture – A System of Patterns*, Wiley, Chichester.
- C4ISR Architectures Working Group report (1998). Levels of Information Systems Interoperability (LISI), DoD, February 1998, Washington, DC.
- Carlock, P.G., and Fenton, R.E. (2001). Systems of Systems (SoS) Enterprise Systems Engineering for Information- Intensive Organizations. *Systems Engineering*, 4/4, pp. 242–261.
- Carney, D., Fischer D., & Place P. (2005). Topics in Interoperability System-of-Systems Evolution, Report CMU/SEI-2005-TN-002.
- Chapurlat, V. (2007). Complex systems verification and validation application to enterprise modelling. Accreditation to supervise research, Montpellier 2 university (in French).
- Clark, T. and R. Jones (1999). Organisational Interoperability Maturity Model for C2, In *Proceedings of the Command And Control Research And Technology Symposium (CCRTS)*, June 29, July 1, Newport, RI, USA.
- Cloutier R.J. and Verma D. (2007). Applying the concept of patterns to systems architecture. *Systems Engineering Journal*, 10/2, 138-154, Wiley
- Coad P. (1992). Object-oriented patterns, *Communications of the ACM* 35 (9), 152–159.
- Cocks, D. (2006). How should we use the term "system of systems" and why should we care? 16th Annual International Symposium Proceedings, System Engineering Shining Light on the Tough Issues, Orlando.
- Coplien J.O., Schmidt D.C. (1995). *Pattern Language of Program Design*, Addison-Wesley, Reading
- Csaji B.C., Monostori L. (2008). A Complexity Model for Networks of Collaborating Enterprises. 17<sup>th</sup> IFAC World Congress, July 6-11, Seoul, South Korea
- DeLaurentis Daniel, Fry Donald, et al. (2006). "Modeling Framework and Lexicon for System-of-Systems Problems." *IEEE Transactions on Systems, Man, and Cybernetics-Part A Systems and Humans*.
- Doumeings G., Mueller J., Morel G., Vallespir B. (2007) Guest Editors, Book on Enterprise Interoperability new Challenges and Approaches, proceedings of I-ESA'06, Springer.
- EIF (2004), European Interoperability Framework for pan-European eGovernment Services, Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens (IDABC), November, Luxembourg.
- Fischwick P.A. (1996). Toward a convergence of systems and software engineering, Technical Report 005, Department of Computer and Information Science and Engineering, University of Florida.
- Fisher, D. A. (2006). An Emergent Perspective on Interoperation in Systems of System, Carnegie Mellon University.
- Ford, T., Colombi, J., Graham, S., Jacques, D. (2007). The Interoperability Score. Proceeding of the 5th Annual Conference on Systems Engineering Research. March 14-16. Stevens Institute of Technology Campus, Hoboken, New Jersey, USA
- Gamma E., Helm R., Johnson R., Vlissides J. (1995). *Design Patterns Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading
- Giachetti, R. E. (2004). A framework to review the information integration of the enterprise. *International Journal of Production Research*, 42(6), 1147–1166.
- Giret A. and Botti V. (2004); Holons and Agents. *Journal of Intelligent manufacturing*, 15, 645-659.
- Gorod Alex, Ryan Gove, et al. (2007). System of Systems Management A Network Management Approach. IEEE International Conference on System of Systems Engineering., San Antonio
- Gouyon, D. (2004). Product Driven Control of Manufacturing Execution Systems synthesis techniques contribution. PhD Thesis, Henri Poincaré, Nancy University (In French).
- Gruber T.R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation* (N. Guarino and R. Poli, eds.), Kluwer Academic Publishers.
- ISO 14528 (1999). *Industrial Automation Systems – Concepts and rules for Enterprise Models*, TC 184/SC5/WG1, Geneva, Switzerland.
- Jackson, S. (2007). System Resilience Capabilities, Culture and Infrastructure. System Engineering Key to Intelligence Enterprises. INCOSE 2007 - 17th Annual International Symposium Proceedings, San Diego.
- Koestler, A. (1967) *The Ghost in the Machine*. London, UK: Arkana Books.
- Kuras and White. (2005). A Multi-Scale Definition of a System, MITRE Technical Report, www.mitre.org.
- Kurtz and Snowden. (2003) The new dynamics of strategy sense-making in a complex and

- complicated world. *IBM Systems Journal*, **42/3**, 462-482.
- McGovern J., Ambler S.W., Stevens M., Linn J., Sharan V., and Jo E. (2003). *The Practical Guide to Enterprise Architecture*. Prentice Hall
- Maier, M. W. (1998). "Architecting principles for systems-of-systems." *Systems Engineering* Vol. 1(4) 267-284.
- Marik, V., Lazansky, J. (2007) Industrial applications of agent technologies. *Control Engineering Practice*.
- Mayer F. (1995). Contribution to manufacturing engineering: application to pedagogical engineering within a CIME centre. PhD Thesis, Henri Poincaré, Nancy University (in French).
- Morel G., Panetto H., Zaremba M.B., and Mayer F. (2003). Manufacturing Enterprise Control and Management System Engineering paradigms and open issues. In *IFAC Annual Reviews in Control*. 27/2, 199-209, Elsevier.
- NATO Allied Data Publication 34 (ADatP-34) (2003). NATO C3 Technical Architecture (NC3TA), Version 4.0.
- Panetto H. (2007). Towards a Classification Framework for Interoperability of Enterprise Applications. *International Journal of CIM*, 20/8, 727-740, Taylor & Francis, December, ISSN 0951-192X.
- Panetto H., Baïna, S., Morel G. (2007). Mapping the IEC 62264 models onto the Zachman framework for analysing products information traceability a case study. *Journal of Intelligent Manufacturing*, 18/5, 679-698, Springer Verlag, December, ISSN 0956-5515
- Pannequin, R. (2007). Proposition of a benchmarking environment of product-driven control architectures. PhD Thesis. Henri Poincaré, Nancy University (in French).
- Sage, A. P. and C. D. Cuppan (2001). "On the Systems Engineering and Management of Systems of Systems and Federations of Systems." *Information, Knowledge, Systems Management* Vol. 2 (No. 4) 325-345.
- Sanz R., Zalewski J. (2000). Pattern-based control systems engineering using design patterns to document, transfer, and exploit design knowledge, *IEEE Control Systems Magazine*, 1-15.
- Schuh, G., Sauer A., Döring S. (2006). Modelling collaborations as complex systems. In: *Proceedings of the 4th International Industrial Simulation Conference (ISC'2006)*, June 5-7, The University of Palermo. Palermo, Italy, 168-174.
- Shaw M. (1995). Beyond objects a software design paradigm based on process control, *ACM Software Engineering Notes* 20 (1), 27-38.
- Simao, J. (2005). A contribution to the development of a HMS simulation tool and proposition of a meta-model for holonic control. PhD Thesis, Henri Poincaré, Nancy University
- Simon. (1990) Simon, H., 1996 *Sciences of the Artificial*. 3rd edition. MIT Press.
- Sowa J. F. and Zachman J. A. (1992). Extending and Formalizing the Framework for Information Systems Architecture, *IBM Systems Journal*, 31/3, 590-616
- Tolk, Andreas. (2003) "Beyond Technical Interoperability – Introducing a Reference Model for Measures of Merit for Coalition Interoperability." In *Proceedings of the 8th International Command and Control Research and Technology Symposium (ICCRTS)*, Washington, DC, June 17-19, 2003.
- Tursi A., Panetto H., Morel G., Dassisti M. (2007). Ontology-based products information interoperability in networked manufacturing enterprises. *IFAC Cost Effective Automation in Networked in Product Development and Manufacturing (CEA'2007)*, Monterrey NL, México, October 2-5
- Valckenaers, P., H. Van Brussel, et al. (2003). On the design of emergent systems an investigation of integration and interoperability issues. *Engineering Applications of Artificial Intelligence*, 16, 377-393, Elsevier
- Vernadat F.B. (2007). Interoperable enterprise systems Principles, concepts, and methods. In *IFAC Annual Reviews in Control*. 31/1, 137-145, Elsevier.
- Wong, C., D. McFarlane, et al. (2002). The intelligent product driven supply chain. *IEEE International Conference on Systems, Man and Cybernetics*.
- Zachman J. A. (1987). A Framework for Information Systems Architecture, *IBM Systems Journal*, 26/3, 276-295