A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

INSIGHT

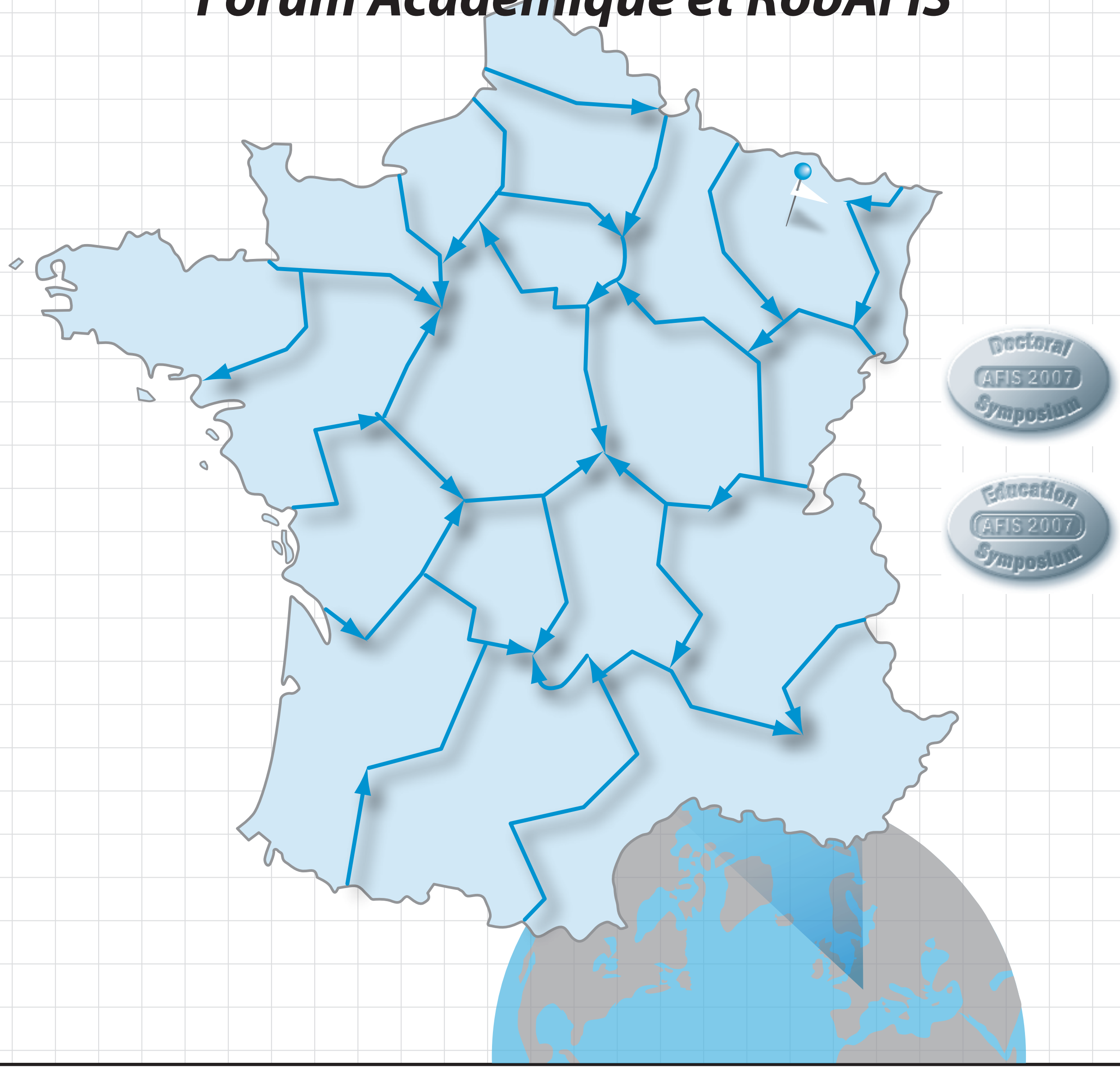
INCOSE
International Council on Systems Engineering

July 2008

Vol 11 Issue 3

SPECIAL FEATURE

The Best of France:
Forum Académique et RobAFIS



INSIGHT

Publication of the International Council on Systems Engineering

Chief Editor Bob Kenley
insight@incose.org (260) 460-0054
Assistant Editor Andrew Cashner
andrew.cashner@incose.org
Theme Editors Hervé Panetto and Gérard Morel
herve.panetto@incose.org and
gerard.morel@incose.org
Advertising Editor Alope Bennett
advertising@incose.org (800) 366-1164
Layout and Design Chuck Eng
chuck.eng@comcast.net (206) 364-8696
Member Services INCOSE Central Office
info@incose.org (800) 366-1164
On the Web http://www.incose.org
Article Submission INSIGHT@incose.org

Publication Schedule. ***INSIGHT*** is published four times per year. Issue and article/advertisement submission deadlines are as follows: *December 2008* issue–15 October; *April 2009* issue – 15 February; *July 2009* issue – 15 May; *October 2009* issue – 13 August. For further information on submissions and issue themes, visit the INCOSE Web site as listed above.

Advertising in *INSIGHT.* Please see http://www.incose.org/ProductsPubs/periodicals/insight.aspx – or e-mail advertising@incose.org.

Subscriptions to *INSIGHT* are available to INCOSE members as part of their membership. Complimentary copies are available on a limited basis. Back issues are available on the INCOSE Web site. To inquire about membership or to order a copy, contact Member Services.

Who are we? INCOSE is a 6000+ member organization of systems engineers and others interested in systems engineering. Its purpose is to foster the definition, understanding, and practice of world class systems engineering in industry, government, and academia. INCOSE is comprised of chapters located in cities worldwide and is sponsored by a corporate advisory board and led by elected officers, directors, and membership board.

President's Corner

Sustainability: A New and Critical Attribute

Pat Hale, patrick.hale@incose.org

As systems engineering professionals, we often mentally categorize items we are processing according to a number of attributes that provide context and definition to the qualities required for successful implementation. As our 2008 symposium theme, "Systems Engineering for the Planet," will have highlighted for all of us, sustainability is quickly becoming a key attribute for all new systems and products and processes, as well as for our economies and social institutions. Although many assume that sustainability refers only to environmental factors, it is now widely agreed to cover a far broader scope. The United Nations 2005 World Summit Outcome Document refers to the "interdependent and mutually reinforcing pillars of sustainable development" as economic development, social development, and environmental protection; sustainability is often illustrated as in figure 1, highlighting the confluence of these interdependencies.

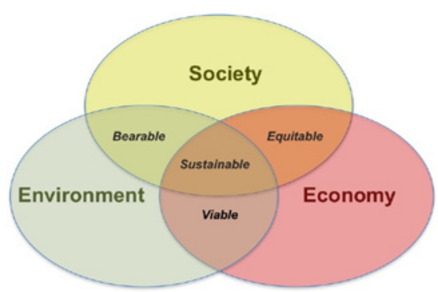


Figure 1. Sustainabilty as the meeting-place of economic development, social development, and environmental protection (Creative Commons license under GNU Free Documentation)[1]

What does this have to do with systems engineering? Our systems and products are becoming dramatically more expensive to develop, and sometimes equally expensive to retire and reclaim. Whether we work in defense or commercial industry, our development products now include environmental (explicit) or sustainability (implicit, in most cases) requirements through included specifications or regulatory constraints. Some areas of the world are progressing more rapidly than others, but with current emphases and regulations concerning carbon footprints, coupled with the present-day examples of energy prices and food shortages, we must balance all three sets of sustainability considerations if we want to be respected as true systems engineering professionals. Note that I am not advocating a political position here — I am simply suggesting that sustainability issues need to be part of our criteria for trade-offs, architecting, and measuring performance as we move forward. For those of you in the commercial sector, consumer preferences will combine with regulatory requirements to demand that sustainability become part of your lexicon of requirements. In the defense domain, some of us will see explicit specifications, but all of us will understand the implicit requirement to be balanced in sustainability attributes, so that we can create systems that will be funded and come to fruition, rather than be killed in the proposal or design stages.

In my "day job" in the university setting, I find that there are few topics that create more excitement in my mid-career technical professional students than learning and applying the policies, design, economics, and leadership associated with solving key societal challenges in a holistic and responsible fashion. The knowledge and skills required to mature as a systems engineering professional are

> *continues on page 5*

1. The image is derived from a French original by Johann Dréo, posted at http://commons.wikimedia.org/ in March 2006

From the Theme Editors

The Best of France: *Forum Académique et RobAFIS*

Gérard Morel, gerard.morel@incose.org; and Hervé Panetto, herve.panetto@incose.org

This issue is devoted to special coverage of the French Systems Engineering Academic Forum,[1] organized by AFIS (Association Française d'Ingénierie Système), the French chapter of INCOSE, with the support of Nancy University. The forum was held 28–29 November 2007 with the objective to develop strong relationships between industry and academia. It gathered 115 participants, of which one third were from industry.

Cross-fertilization between academia and industry occurred in twelve workshops addressing these topics:

- Learning systems engineering while doing research
- Learning systems engineering while doing projects
- Understanding systems engineering basics for teaching
- Training for systems engineering
- Deploying systems engineering education and research at the international level
- RobAFIS robotics competition for students
- Deploying systems engineering to others organizations
- Model-driven systems engineering and practice
- System of systems versus complex systems
- Resilient versus dependable systems
- Integration, validation, verification, and qualification
- Integrated operational systems

1. A full report of this event (in French) has been published in the twelfth issue of the AFIS newsletter at http://www.afis.fr/

President's Corner *continued from page 3*

becoming ever broader and more challenging, and are beginning to expand into dramatically different domains from traditional engineering. I find these changes to be energizing and exciting, and I hope you do, too.

Four invited lectures were presented:

- The national AIP-PRIMECA network presented on training using projects;
- The 3S R&D network, on the security and dependability of large-scale systems;
- PSA, on systems engineering practices for the functional design of motor-propulsion groups; and
- Jean Bezivin from University of Nantes, on model-driven engineering

AFIS organized a pre-forum for teachers, students, and employees of industry who were not members of AFIS to introduce the attendees to the practices, issues, and challenges of system engineering.

PhD students wrote the sixteen papers published in this issue based on the presentation of posters during a workshop on "Learning Systems Engineering While Doing Research." The objective of the workshop was to disseminate current doctoral research that is linked with industry needs.

Some of the papers describe systems engineering from an engineering perspective, in several different ways::

- **Mapping functional architecture decisions onto physical product architectures** with the aim of providing system architects with a method to simulate the mapping of the functional architecture onto the physical architecture by propagating functional choices and assessing alternatives.
- **Providing a system modeling and analysis framework for risk analysis in socio-technical systems** that integrates different concepts and tools coming from system theory, system engineering practices, and theoretical principles of risk management into framework to facilitate the system engineering process.
- **Using systems-of-systems engineering to improve the integration of enterprise-control systems** to help enterprises evolve their information system in response to changes in the internal and external business environment, and to address the formal mathematical definition of a product-driven interoperability relationship.
- **Proposing an ontology-based approach to knowledge management in system engineering processes**, arguing that knowledge about engineering processes constitutes one of the most valuable assets of a systems engineering organization. Normally, this knowledge is only implicit, and depends heavily on each systems engineer's own personal experience.

From a system perspective, **enterprise interoperability measurement** allows an enterprise to fully evaluate its own capacity to interoperate, and therefore to anticipate possible problems before a partnership. More specifically, **configuring product line requirements in the context of multiple models** facilitates configuring the products themselves.

Combining SysML and formal methods for safety requirements verification helps in the verification of safety requirements for the design of complex control systems involving software, mechanical, electrical, or pneumatic components for industrial safety-critical applications such as controlling power plants or embedded control systems, especially in the transport area.

Research issues are discussed on **implementing a quality reference and interoperability of processes in software collaborative projects** by integrating two standards of quality—*ISO 9001:2000* and Capability Maturity Integration (CMMI).

Other research papers define methodologies for systems engineering by

- **introducing multi-criteria decision making into software engineering** through a methodological support that guides software engineers through tactical choices with the application of multi-criteria methods;
- **defining a methodology for a probabilistic risk analysis of socio-technical systems** that supports a probabilistic risk analysis, and addresses analysis from multiple points of view in an integrated way by using one risk model to estimate the ocurrence

> *continues on next page*

From the Theme Editors *continued*

of risky scenarios and the impacts of safety barriers;

- **Proposing regeneration engineering for assessing weapon system availability** in dependability studies during the design phase;
- **managing product specifications' dependencies in product development systems** through a graph-oriented analysis of the dependency relationships between product specifications;
- **modeling strategic alignment with the INSTAL method** by considering strategic alignment as an engineering activity and
- designing a coherent and structured set of good practices or requirements, applied to a first study on **determining the overlap between quality repositories and regulatory requirements for the pharmaceutical industry.**

Systems engineering practices are also studied from an education perspective. Research in this domain deviates from education science, which optimizes the training process, and proposes an approach for learning systems optimization based on systems engineering processes used in operational processes design. It results in **optimizing learning systems using methods and tools for designing operational processes,** and **a control system for learning applied to higher education** that defines a performance measurement system of higher education institutions linked with action levers allowing a reactive control.

The next French Systems Engineering Academic Forum will be organized by École des Mines d'Alès (Nîmes, France) from 2 to 3 December 2008.

We would like to thank all authors for their contributions, and all reviewers for the tremendous work they have done to improve the papers.

INSIGHT SPECIAL FEATURE

Optimizing Learning Systems Using Methods and Tools for Designing Operational Processes

Olivier Bistorin, olivier.bistorin@incose.org; Thibaud Monteiro, thibaud.monteiro@incose.org; and Claude Pourcel, claude.pourcel@incose.org

Education is an important factor in a nation's development, because it is directly connected to the rate and the level of population employment, which generates growth (OECD 2006; Eurydice 2007). National states dedicate to education an average of 6.26% of their gross domestic product, and this contribution, in most cases, represents the foremost public budget item. For this reason, it is necessary for a nation to optimize its learning systems (Kogan and Hanney 2000).

Learning Systems Engineering

Deviating from the methodologies of education sciences (which optimize the training process), we present in this article an approach for optimizing the learning systems based on a systems engineering method used for designing operational processes. To this end we apply two strategies:

1. *Educational strategy*. Develop training processes adapted to the needs of our customers, while respecting the constraints fixed by supervisory bodies.
2. *Management strategy*. Manage our learning system by applying the principles, methods, and tools used for systems of goods and services production.

Our approach depends on the analogy between learning systems and systems for producing goods and services presented by Christian Clémentz (2000).

Our Approach

By using the modeling method MECI (*Modélisation d'Entreprises pour la Conception Intégrée*, "Companies Modeling for Integrated Conception") (Pourcel and Gourc 2005) and the modeling language UML (Unified Modeling Language) (Object Management Group 2007), we identify operational processes for learning systems and clarify what are the various actors, activities, tasks, and skills within these processes (Bistorin 2007). Our method then proceeds to three

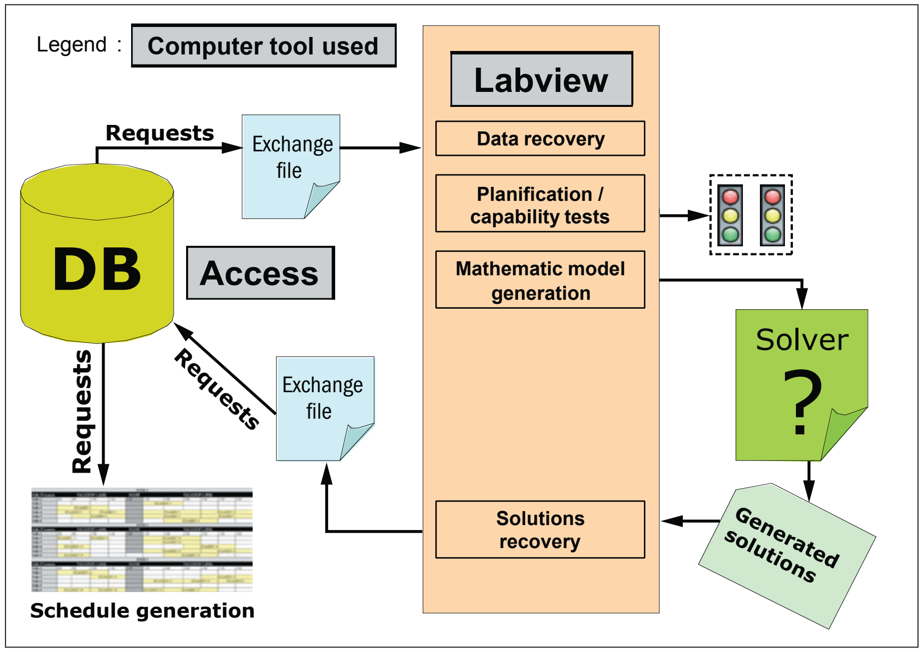


Figure 1. Our global approach

different stages: designing the learning route, planning the training programs, and, finally, proposing a mathematical model for scheduling. Every stage is associated with suited tools like Access for database management or Labview for creating learning routes and formalizing mathematical constraints. Figure 1 presents the global logic of these design processes.

By the use of Petri nets, we simulate several learning routes that respect the logic of learning, because they satisfy the constraints that arise due to the fact that learning some skills is prerequisite to learning other skills (Bistorin, Mon-

> continues on next page

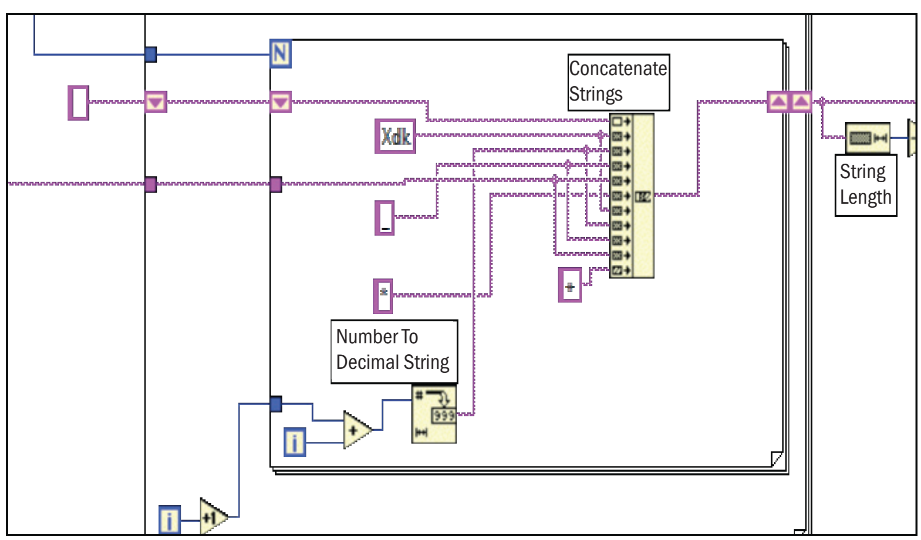


Figure 2. One stage of constraint formalization with Labview®

Bistorin *continued*

teiro, and Pourcel 2006). This simulation allows us to obtain one or several solutions that satisfy these constraints on prerequisites imposed by the educational aspect, but numerous other constraints on functioning and optimization exist within the learning system. These constraints aim at maximizing the performance and the triptych of costs, quality, and delays. In order to satisfy all the constraints on the learning system, we establish a mathematical model of the various parameters of the system. Our model consists of twenty-four families of constraints. By interfacing Microsoft Access with Labview®, we are able to extract automatically the information inherent to a particular learning system and finally to generate all the constraints of the system with the aim of solving the optimization problem as shown in figure 2. However, the exact solution methods that we used (linear programming with lp_solve or the CPLEX solver) do not allow us to propose a complete solution for scheduling. In the future, we wish to turn to methods for heuristics resolution in order to propose a coherent scheduling solution with the strategic objectives described above.

References

Bistorin, O., T. Monteiro, and C. Pourcel. 2006. Modélisation du parcours apprenant dans un système de production de compétences. Paper presented at the 6th French-Speaking Conference on Modeling and Simulation (MOSIM'06), Rabbat, Morocco.

Bistorin, O. 2007. Méthodes et outils d'aide à la conception des processus opérationnels des systèmes de formation. PhD diss., University of Metz.

Clementz, C. 2000. Modélisation des systèmes de production de compétences: Apports à l'ingénierie pédagogique. PhD diss., University of Metz.

Eurydice. 2007. *Structures of education, vocational training and adult education systems in Europe*. Brussels: Eurydice.

Kogan, M., and S. Hanney. 2000. *Reforming higher education.* Higher Education Policy Series. London: Jessica Kingsley Publishers.

OECD (Organisation for Economic Co-operation and Development). 2006. *Education at a glance.* Paris: OECD.

Object Management Group (OMG). 2007. *Unified Modeling Language, specifications v2.1.1.* Needham, MA, U.S.A.: OMG.

Pourcel, C., and D. Gourc. 2005. *Modélisation d'entreprise par les processus: Activités, organisation & applications.* Toulouse, France: Cépaduès-éditions.

Mapping Functional Architecture Decisions onto Physical Product Architectures

Eric Bonjour, eric.bonjour@incose.org; **Maryvonne Dulmet**, maryvonne.dulmet@incose.org; **and Ghassen Harmel**, ghassen.harmel@gmail.com

Defining product architectures is a critical activity during the preliminary system definition stage. Whereas design methods such as axiomatic design, quality function deployment, or "systems architecting" (Maier and Rechtin 2002) have described rules for mapping functions onto components, it is notable that no formal method exists either to jointly simulate the functional and physical architectures or to propagate the impact of changes to a mapping.

According to Ulrich (1995), product architecture consists of (1) the arrangement of functional elements (functional architecture), (2) the mapping from functional elements to physical components, or building blocks, and (3) the specification of the interfaces between interacting components (physical architecture). Satisfactory modular product architectures can be defined as the clustering of components such that the degree of interaction or dependency is maximised within building blocks (or modules) and minimized between them. Modules are commonly described as sets of functionally or structurally interdependent components. An ideal modular architecture corresponds to direct mapping between functional modules and component modules. This kind of architecture produces well decoupled modules (i.e., design sub-problems) that can be easily specified, and then dealt with by different teams concurrently and reused in future system design. Unfortunately, in real design situations, the modules are not completely decoupled, and moreover, there are integrative elements that link all the elements together. Consequently, it is difficult to group a set of functions and to attach this set to a physical module, for instance to facilitate the management of outsourcing. When couplings and mapping have not been formally addressed, the integration of the teams' contributions is more difficult and requires numerous design iterations.

Two kinds of matrices may support the representation of system architecture: (1) the traceability and allocation matrix (IEEE 1220-2005) or Domain Mapping Matrix (DMM) (Lindemann 2007), and (2) the Design Structure Matrix (DSM) (Steward and Donald 1981; Pimmler and Eppinger 1994). The matrices in the first category represent relationships between two domains, such as between product functions and components, or activities and teams. The matrix in the second category represents relationships between elements within the same domain. Few architecting methods have been developed to identify modular product architectures. The inputs of these methods usually are functional models (Stone et al. 2000) or component interactions (Pimmler and Eppinger 1994; Browning 2001). Occasionally, the inputs are a mapping of functions onto physical components (Liu et al. 1999), or more complex data intended to take into account key factors ("module drivers") of the whole system's lifecycle (Erixon 1998).

We aim to provide system architects with a method that helps them to simulate the mapping of the functional architecture onto the physical architecture by propagating functional choices and then assessing alternatives. In new product development situations or in re-engineering projects, system architects could use this method in the early design stages before the physical architecture is studied in detail. We have developed this method in collaboration with the powertrain design department of a French car manufacturer and applied it to the definition of new diesel engine architectures.

We summarize the propagation method in four steps (see figure 1). First, we capture the functional architecture decisions by identifying and evaluating couplings between functions (F DSM, shown in the leftmost column of figure 2). Second, we build the matrix for allocating functions to components

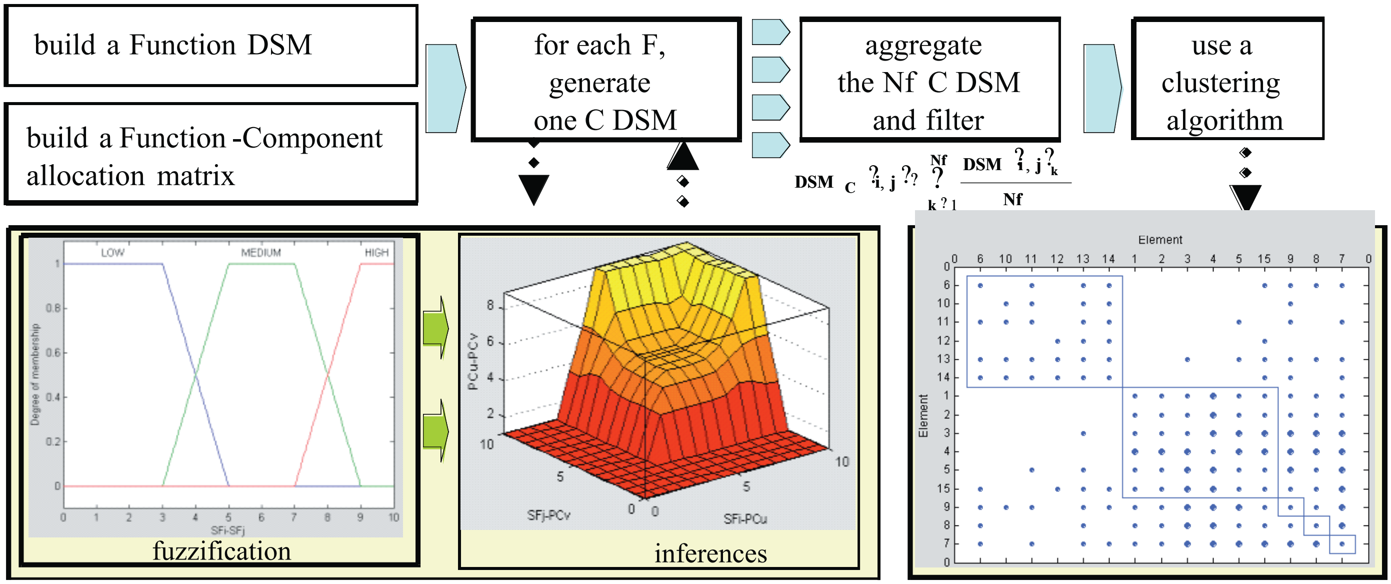


Figure 1. A matrix-based method with fuzzy processing

(F–C DMM, shown in the right half of figure 2, left column). A DMM value corresponds to the contribution intensity of a component to the fulfillment of a function. Third, we propagate functional architecture choices through the F–C DMM in order to generate component DSM (C DSM). Fourth, we identify a satisfactory physical architecture by applying a clustering algorithm. If the system architects are not satisfied, they could change the initial matrices and simulate new physical architectures.

The scientific challenge relies on the formalization of axioms and rules at the basis of the generation step. Moreover, since intensity values inside DMM and DSM are quite imprecise and subjective, we use a fuzzy inference system. We define fuzzy rules based on the following axiom: if (F_i, F_j) and (C_u, C_v) are coupled and if F_i and F_j interact, then C_u and C_v interact.

In the engine design project, our method helped architects to question their decisions and consider alternatives concerning the F DSM and F–C DMM and to iterate the determination of these matrices in order to define satisfactory engine architecture. Obviously, the generated architecture has to be assessed by taking into account other requirements of the whole system's lifecycle. But the architects should be aware of the fact that the choice of other modules could increase coordination and teams' efforts in the stages of detail design and system integration.

We will continue to work on applying this method to propagate changes from one domain to another: for instance, function to component, product to activity, or activity to team.

References

Browning, T. R. 2001. Applying the design structure matrix to system decomposition and integration problems. *IEEE Transactions on Engineering Management* 48: 292–306.

Erixon, G. 1998. Modular function deployment: A method for product modularisation. PhD diss., Royal Institute of Technology, Stockholm.

Software Engineering Committee of the IEEE Computer Society. 2005. *IEEE 1220-2005: IEEE standard for application and management of the systems engineering process*. Revision of *IEEE 1220-1998*. Washington, DC: Institute of Electrical and Electronics Engineers Computer Society.

> continues on next page

Function DSM	F1 Air providing	F2 Fuel providing	F3 Gas cleaning-up	F4 Combustion	F5 Pressure-Torque Conversion	F6 Lubrication, ventilation	F7 Power transmission	F8 Secondary energy conversion	F9 Vibration	F10 Control	F11 Thermics	F12 Coupling	F13 Functional Volumes
F1	10	9	7	8			6	5	6	8	7		6
F2	9	10	9	9			7		7	8	5		7
F3	7	9	10	9		8	8		7	7	9		7
F4	8	9	9	10	8	4			8	9	8		7
F5				8	10	9	9	8	8		7	8	8
F6			8	4	9	10	7	8	5	6	8		6
F7	6	7	8		9	7	10	8	8		8	9	8
F8	5				8	8	8	10	3		5		6
F9	6	7	7	8	8	5	8	3	10	5		7	
F10	8	8	7	9		6			5	10	5		5
F11	7	5	9	8	7	8	8	5		5	10	8	
F12					8		9		7		8	10	8
F13	6	7	7	7	8	6	8	6		5		8	10

Functions - Components Allocation Matrix (DMM)		F1 Air providing	F2 Fuel providing	F3 Gas cleaning-up	F4 Combustion	F5 Pressure-Torque Conversion	F6 Lubrication, ventilation	F7 Power transmission	F8 Secondary Energy Conversion	F9 Vibration	F10 Control	F11 Thermics	F12 Coupling	F13 Functional Volumes
C1	EGR	8	0	9	0	0	0	0	0	6	8	7	0	6
C2	Fuel System	0	8	0	8	0	0	7	0	7	8	5	0	8
C3	Breech Block	5	5	6	5	0	8	7	5	7	0	8	0	8
C4	Air Intake	9	9	6	9	0	0	4	5	6	9	3	0	7
C5	Exhaust	8	0	9	0	0	6	5	0	6	6	8	0	6
C6	Camshaft/Valve Train	9	9	0	0	0	7	0	8	8	0	0	0	7
C7	Crankshaft	5	0	0	7	9	7	7	0	8	0	8	9	8
C8	Casing/Housing	0	0	0	8	6	8	9	0	8	0	8	9	9
C9	Lubrication, Blow-By	5	0	0	0	8	9	0	4	5	3	8	0	7
C10	Accesory Drive	0	0	0	0	6	0	8	9	5	0	6	0	8
C11	Synchronous Drive	0	0	0	0	8	9	0	9	8	0	6	0	7
C12	Vacuum Circuit	8	0	0	0	0	8	0	9	0	0	0	0	8
C13	Cooling Circuit	0	0	0	0	0	5	8	5	6	0	9	0	7
C14	Secondary Energy Gener.	0	0	0	0	0	4	5	9	5	0	5	0	7
C15	Sensors and control	7	9	7	8	0	6	0	0	0	9	9	8	4

Figure 2. Instances of function DSM and functions–components DMM

> continues on next page

A Control System for Learning Applied to Higher Education

Rémi Renauld, remi.renauld@incose.org; P. Padilla; D. Roy; and C. Pourcel, claude.pourcel@incose.org

Learning systems are a particular class of systems designed to produce services; their main purpose is to increase the competency of learners. This type of system is likely to face many transformations in the next few years. Indeed, learning systems must be able to respond to the changing needs of learners and of the industries that recruit the learners for employment. Moreover, they will operate in an increasingly competitive and more aggressive environment (Clementz 2000). The primary objective of this article is to define a performance measurement system of higher education institutions, which we will then link with action levers to allow a reactive control. Note that the main specificity of the learning system is that the "product" (learner) is also the actor in the production process and the client of the organization that provides the system. Thus we are confronted with a new type of product, "the active product"; and the control system must be able to take into account this characteristic of the learning system.

We propose an integrated and reactive system of control, using a two-level cooperating structure of control. We consider the system to be integrated, because it is a "control layer" that is superposed onto the process map of the learning system (figure 1). In fact, the global control inserts into each process a "root," linked to the "trunk," which acts as a supervisor.

To reinforce the integration of the system of measurement to the operational system, we propose to the users measurements that are derived from local objectives that trace to strategic objectives (Kaplan 2003). Each measurement is linked to one and only one process of the system. Moreover, each of them is also linked to a Success Key Factor to facilitate the building of efficient dashboards in which the measurements could be aggregated in performance indicators.

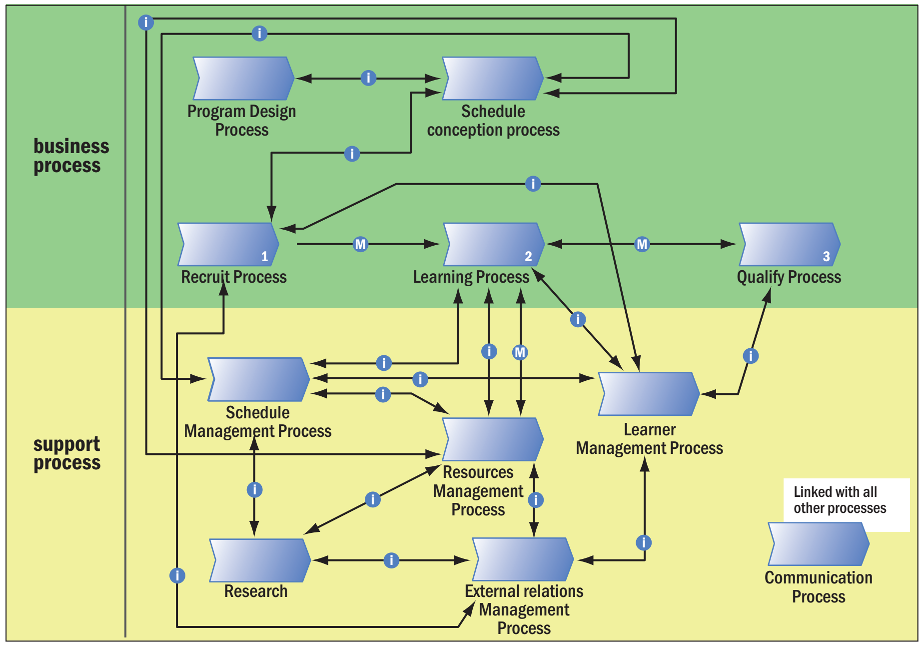


Figure 1. Restrained process map of a learning system

To be generic, we classify them in two groups, one considered as necessary and one facultative. Next, the final users can build their own control system in function of their needs. The creation of the dashboards constitutes the main part of the reactivity mechanism of our system and at different levels. Finally, we identify four loops (or levels) of reactivity, which are initiated following specific behaviors. Those behaviors are modeled using statecharts to show which events cause the system to react.

To validate our approach, we intend to deploy it on a small part of the French engineering education system at a school called Ecole Nationale d'Ingénieurs de Metz, choosing one option of the last semester of formation. Then, we will simulate the reactions of our control system following scenarios focusing on particular issues. As an extension of this work, we can consider applying the control system to any system in which the product is also an actor and/or a client, even if such situations are rather rare.

References

Clementz, C. 2000. Modélisation des systèmes de production de compétences: Apports à l'ingénierie pédagogique. PhD diss., University of Metz.

Kaplan, R. S., and D. P. Norton. 2003. *The balanced scorecard: Translating strategy into action.* Boston, MA: Harvard Business School Press.

Bonjour *continued*

Lindemann, U. 2007. A vision to overcome chaotic "design for X" processes in early phases. Paper presented at the International Conference on Engineering Design, 28-31 August (Paris).

Liu, Y.-C., A. Chakrabarti, and T. P. Bligh. 1999. Transforming functional solutions into physical solutions. Paper presented at the 11th International Conference on Design Theory and Methodology (ASME Design Engineering Technical Conferences), 12–16 September (Las Vegas, NV), Vol. 3.

Maier, M. W., and E. Rechtin. 2002. *The art of systems architecting.* 2nd ed. Boca Raton, FL: CRC Press.

Pimmler, T. U., and S. D. Eppinger. 1994. Integration analysis of product decompositions. In *Proceedings of the ASME Design Theory and Methodology Conference* (Minneapolis, MN), vol. 68: 343–351.

Steward, R. P., and V. Donald. 1981. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management* 28 (3): 71–74.

Stone, R., K. Wood, and R. Crawford. 2000. A heuristic method for identifying modules for product architectures. *Design Studies* 21: 5–31.

Ulrich, K. T. 1995. The role of product architecture in the manufacturing firm. *Research Policy* 24: 419–440.

A System Modeling and Analysis Framework for Risk Analysis in Socio-technical Systems

Saber Aloui, saber.aloui@gmail.com**; and Vincent Chapurlat,** vincent.chapurlat@incose.org

A complex system, which integrates multiple technologies into a socio-technical system, is composed of heterogeneous and interacting subsystems, components, or human actors evolving in a moving environment. For this reason a complex system will have to face unpredictable and unforeseen events throughout its lifecycle, which can induce unexpected behaviors and risky situations. These behaviors and situations can result in undesirable outcomes regarding the system's performance (in terms of delay, cost, and quality of service, for example), its stability, and its integrity. So how is it possible to design a more robust system that copes with risks?

In industry, risk management approaches have been successfully developed in parallel with systems engineering approaches (CAS Enterprise Risk Management Committee 2003). But they generally remain separated from each other or adapted to the study of given phenomena in some domain (such as a nuclear plant, a manufacturing plant, or the food industry). Our goal is to integrate different concepts and tools—from systems theory, systems engineering practices, and theoretical principles of risk management—into a framework for modeling and analyzing complex systems.

Requirements

Anyone involved with systems engineering must first gather and formalize as much knowledge as possible about a system. This may be achieved by simultaneously modeling the functions, behaviors, and structure of the system; the dynamics of the environment with which the system interacts; and the predictable risks that may occur and can impact the system. Second, the resulting model has to take into account and to describe the different points of view and known situations coming from all concerned with the system (e.g., the user, designer, or developer). Last, due to the inherent complexity of the system and therefore of its model, analysis mechanisms and tools are required to make the different points of view coherent, as well as to prove that the modeled system is robust in the face of different situations and operational scenarios.

Modeling

We have enlarged and formalized the systems engineering (Haskins 2006) framework called SAGACE (Penalva 1994) for guiding the modeling process. The result is a multi-view and multi-paradigm model. A view permits gathering and formalizing a given type of knowledge, focusing on the same aspect of the system. We propose four views:

1. *Functional*. What is the mission of this system? Why does it exist? What are its objectives—in other words, what level of performance does it intend to achieve? What are the different functions of the system?
2. *Structural*. What are the processes and activities that implement the functions of the system? What are the components and sub-systems (or even the resources and their interaction) in order to support these processes?
3. *Behavioral*. What are the possible operational scenarios and configurations of the system that authorize or limit the scenarios? What are the functioning modes? How does the system evolve to take into account the environments and events? How might it be adapted and controlled in order to avoid damage in case of emergency?
4. *Property*. This view allows users to enrich the model with complementary knowledge , in order to link the partial models formalized in the three previous views. This knowledge is represented by using the concept of property (Chapurlat, Kamsu-Foguem, and Prunet 2003). It expresses functional or non-functional requirements, such as coherence rules between views and between partial models, semantic rules, attribute evolution laws, expected behavior, constraints, and objectives. A property may also express potential causes and effects of risks. It is formally defined by a causal and typed relation that links two sets of events and data that come from partial models.

Each of these views is expressed by different actors (such as modelers, engineers, or other specialists in the field of study) to explain and describe their own point of view and express their own objectives. To model these views, both a common and unique ontology is defined. The ontology gathers terms that are commonly used and shared by all actors for describing the characteristics of the system. This ontology represents a unique, coherent, and sufficient set of concepts required for representing each view of the entire organization. In other words, while both respecting the paradigm of model-driven architecture and avoiding the compatibility problem between modeling languages, this ontology provides a unique and unified meta-model that allows us to adapt and unify some existing and pre-selected modeling languages from the domains of enterprise modeling and systems engineering that were suitable to each view. For example, the functional view uses the objective modeling language proposed by KAOS (Bertrand, Darimont, Delor, Massonet, and Van Lamsweerde 1998) and the IDEF-0 functional modeling language (Menzel and Maier 1998). The Unified Enterprise Modeling Language (UEML) (Panetto, Berio, Benali, Boudjlida, Petit 2004) allows one to describe the organizational view. Finally, enhanced Functional Flow Block Diagrams (eFFBD) (Oliver, Kelliher, and Keegan 2004) permit an engineer to describe operational scenarios in the behavioral view.

> *continues on next page*

Aloui *continued*

Analysis

The analysis process consists of checking the mode, that is, proving in a formal and automated manner that the specified properties are verified by the model. If this is not the case, the analysis process must provide a counterexample that indicates the reasons for which the property is unsatisfied. The modeler may then detect modeling errors, mistakes, or misunderstandings, thereby increasing the level of confidence on the model.

Verification aims to check the coherence of each view and between each view. Coherence of each view means the coherence of the data and knowledge collected into a view: this induces the checking of properties that describe coherence and construction rules, while taking into account the different levels of details expressed by using the same modeling language. Coherence between each view means the coherence of the data and knowledge collected or used in two separated views: this induces the checking of properties that describe coherence and construction rules, while taking into account the different modeling languages.

Validation aims to check the relevance of the model, that is, to evaluate the distance between the model and the real system. Proving some particular properties that describe system requirements accomplishes validation. Since this validation process must take into account the limitations of classical modeling hypothesis and modeling languages (for example, due to a semantic distance between concepts and relations handled by the modeling languages), the validation necessarily remains limited. When the model has been verified and (as much as possible) validated, the model is used for detecting causes of potential risks by proving that properties, which model the causes and effects of a potential risk, are not verified.

We propose formal rewriting mechanisms to translate the system model towards a formal model. Verification tools such as model checkers or theorem provers (ParaDiSe 2008) can then be used. However, the proposed checking technique is based on a formal knowledge representation and analysis language called Conceptual Graphs (Sowa 1984).

Results

We have applied this approach to risk management in health care organizations. Risks can result in bad outcomes for the patient or reduce the organization's performance. The modeling process provides a model of the organization that incorporates multiple points of view. We developed a properties repository by taking into account the concept of Cindynogenic Structural Deficiencies (Kervern 1994). This allowed us to characterize the different kind of risks, their causes, and their effects on the patient in a generic manner. The modeler can then parameterize some generic properties and apply them to the system in question. We applied the analysis process to detect some dysfunctional modes of the organization.

Perspectives

Our research has aimed to enlarge the set of mechanisms available for systems analysts by adding multi-agent systems methods. As several other researchers have proposed, each agent represents a human resource that is interacting with the system. Each agent can evolve independently from other agents, and it can communicate and share information with them. The main interest of the proposed extension of this work is to formalize and develop embedded checking mechanisms in each agent (Cardoso 2007). These allow engineers to verify local properties and then to modify the current behavior of the agent. Indeed, if a property cannot be verified—if a requirement is not assumed or a risky situation becomes possible—then the agent must change or adapt its own behavior for assuming its mission in the system. A new evolution scenario may be then detected and suggested to the designer.

References

Bertrand, P., R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde. 1998. GRAIL/KAOS: An environment for goal driven requirements engineering. Paper presented at the 20th International Conference on Software Engineering, IEEE-ACM, Kyoto.

Cardoso, H. L. 2007. Integrating JADE and Jess. http://jade.tilab.com/doc/tutorials/jade-jess/jade_jess.html.

CAS (Casualty Actuarial Society) Enterprise Risk Management Committee. 2003. *Overview of enterprise risk management*. [Arlington, VA]: Casualty Actuarial Society. www.casact.org/pubs/forum/03sforum/03sf099.pdf.

Chapurlat, V., B. Kamsu-Foguem, and F. Prunet. 2003. Enterprise model verification and validation: An approach. *Annual Review in Control* 27 (2): 185–197.

Haskins, C., ed. 2006. *Systems engineering handbook: A guide for system life cycle processes and activities.* Version 3. Seattle: INCOSE.

Kervern, G. Y. 1994. *Latest Advances in Cindynics.* Paris: Economica.

Menzel, C. P., and R. J. Mayer. 1998. The IDEF family of languages. In *Handbook on architectures of information systems,* ed. P. Bernus, K. Mertins, and G. Schmidt. Berlin: Springer.

Oliver, D. W., T. P. Kelliher, and J. G. Keegan, Jr. 1997. *Engineering complex systems with models and objects.* New York: McGraw-Hill.

Panetto, H., G. Berio, K. Benali, N. Boudjlida, and M. Petit 2004. A Unified Enterprise Modelling Language for Enhanced Interoperability of Enterprise Models. Paper presented at the 11th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2004), Brazil.

ParaDiSe (Parallel & Distributed Systems Laboratory). 2008. Yahoda verification tools database (Web site). Brno, Czech Republic: Masaryk University Faculty of Informatics. http://anna.fi.muni.cz/yahoda/.

Penalva, J.-M. 1994 SAGACE : La modélisation des systèmes dont la maîtrise est complexe. (SAGACE : The modelling of systems, the mastery of which is complex.) Paper presented at the 2nd International Conference on Integrated Logistics and Concurrent Engineering and Exposition (ILCE-94), Montpellier, France.

Sowa, J. F. 1984. Conceptual structures: Information processing in mind and machine. New York: Addison-Wesley.

Using Systems of Systems Engineering to Improve the Integration of Enterprise-Control Systems

Jean-Philippe Auzelle, jeanphillipe.auzelle@incose.org; **Gérard Morel**, gerard.morel@incose.org; **Hervé Panetto**, herve.panetto@incose.org; and **Frédérique Mayer**, frederique.mayer@incose.org

As the lifetime of many products (that is, goods or services) gets shorter, enterprises have to evolve their information system to account for changes in the internal and external business environment. Indeed, enterprises are using more and more software applications, such as APS, ERP, MES, and SCM[1] to deal with the increasing complexity of information flows. Moreover, these applications are often distributed among facilities with various owners and many suppliers (see figure 1). However, this supply-chain environment is quite unstable, because consumers or customers have an increasing need to customize their products in ways that respond to the changes in supplier networks. That has come about as a result of the new phenomenon of extended enterprises. These business constraints cause issues with short-lived relationships about the way the systems interoperate. In a more general way, this happens not only to enterprise software applications but also to any enterprise system that handles information.

The Research Centre for Automatic Control, Nancy, France, has been studying these interoperability issues for several years. The research team that focused on product-driven systems control has suggested a new paradigm: "product-driven interoperability" (Morel, Panetto, Zaremba, and Mayer 2003). In particular, in his PhD thesis, Jean-Philippe Auzelle is studying interoperability issues and formalization, where enterprise systems are dealing with information management in a manufacturing context.

In this context, a product embedding information about itself and being able to communicate with its environment may be qualified as an "active product." Among other information, such a product defines the set of capabilities needed to produce it. In order to be manufactured, those defined capabilities are provided by resources within a manufacturing system. These resources are controlled by a set of enterprise systems. When an "active product" joins this set of enterprise systems (figure 2), we are demonstrating that the new system (the existing enterprise system as well as the product) is characterised by some specific properties derived from autonomy, belonging, connectivity, diversity, or emergence. From this new system emerges a new mission devoted to processing this new coming "active product" (Morel, Panetto, Mayer, and Auzelle 2007). This new system may then be assimilated to a system-of-systems (Boardman and Sauser 2006) by gathering all enterprise systems, considered themselves as autonomous systems, together with the active product. All enterprise systems are also connected to the active product through an interoperating relationship that must be formalised. Intuitively, we were then inspired by the system-of-systems paradigm and all associated tools and models to study a system-of-systems-like perspective for integrating enterprise-control system.

1. These are the advanced planning and scheduling system, the enterprise resource planning system, the manufacturing enterprise system, and the supply chain management system.

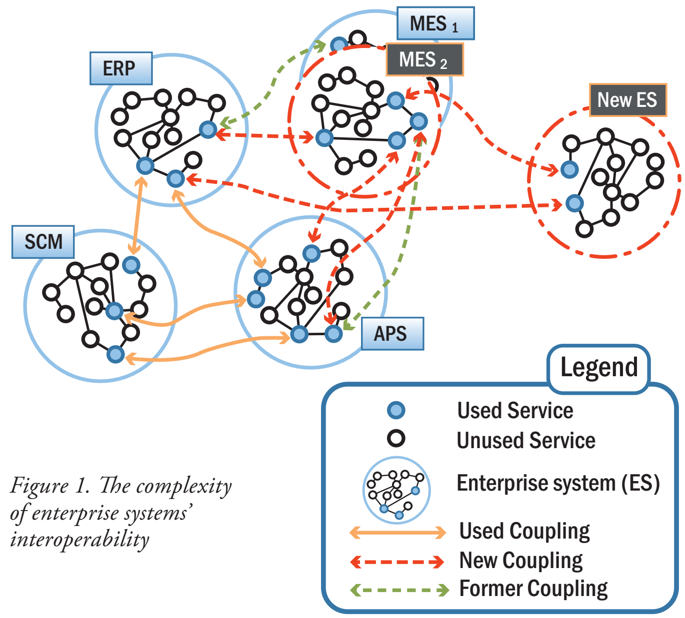


Figure 1. The complexity of enterprise systems' interoperability

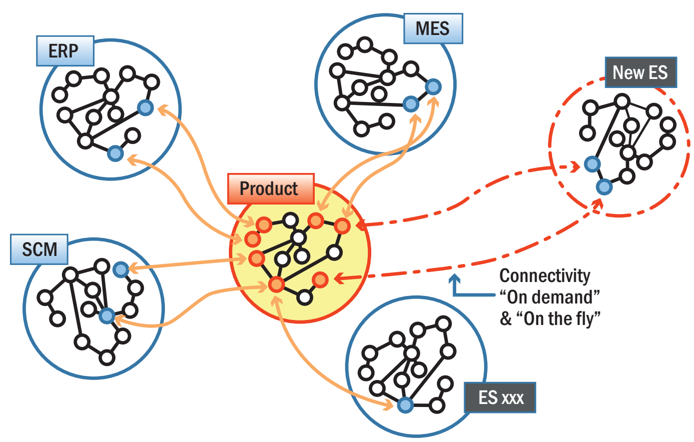


Figure 2. Product-driven interoperability

In this system-of-systems-like context, our current research addresses the formal mathematical definition of a product-driven interoperability relationship, taking into account a short-lived connectivity between an active product and a set of enterprise systems. This relationship implies "on the fly" information exchange for ensuring "on demand" processes.

At present, technologies based on enterprise application integration (EAI) or service-oriented architecture (SOA) offer languages and protocols to aid exchange of service-based information. Semantic requirements are not taken into account with these tools. Consequently, our research challenge is to explore the perspective of semantic relationships to give it more transparency and make interoperability seamless.

Our ongoing work aims to explore systems engineering

> continues on next page

An Ontology-based Approach to Knowledge Management in Systems Engineering Processes

Olfa Chourabi, olfa.chourabi@incose.org**; and Yann Pollet,** yann.pollet@incose.org

Systems engineering processes comprise highly creative and knowledge-intensive tasks that involve extensive problem-solving and decision-making activities among interdisciplinary teams (Meinadier 2002). Systems engineering projects involve the definition of multiple artifacts, which present multiple formalization degrees, such as requirements specification, system architecture, hardware and software components, and so on. All these elements along with "best practices" and engineers' "know-how" form implicit knowledge assets that are not valued enough in systems engineering practice. In such dynamic settings, traditional information management and product modeling systems fail to provide adequate support because their fine-grained process knowledge is not formalized. Process support should therefore alleviate this shortcoming by providing integrated techniques that allow management of experiences during process execution.

We argue that knowledge about engineering processes constitutes one of the most valuable assets of a systems engineering organization. Normally, this knowledge is only known implicitly, relying heavily on the personal experience of each systems engineer. To fully exploit this intellectual capital, it must be made explicit and shared among system engineers. Consistent and comprehensive knowledge management methods (Awad and Ghaziri 2004) need to be applied to capture and integrate the individual knowledge items emerging in the course of a system engineering project.

This paper is a part of an ongoing research project that aims to develop a semantic framework for capitalizing and reusing knowledge for the systems engineering community. The key idea behind our approach is a flexible ontology-based schema with formally defined semantics to enable the capture and reuse of system engineering experiences. In the engineering domain, an ontology is considered "a system (systematic, operational and prescriptive definitions) of fundamental concepts and relationships which shows how a model author views the target world and which is shared in a community as building blocks for models" (Mizoguchi and Kitamura 2000). Our environment is based on a set of consensual ontologies where entities such as models of processes, products, and information are interlinked to represent the knowledge essence of the engineering domain.

Ontologically-based Knowledge Modeling for Systems Engineering Processes

We focus here on the knowledge modeling issue, that is often considered as the first step in developing a knowledge management system. The aim of this process is to understand the types of data structures and relationships within which knowledge can be held and reasoned with. We use ontologies to describe the knowledge model by a formal representation language with expressive semantics. In order to determine the basic building blocks of the knowledge model, we introduce the notion of "systems engineering project asset" as the smallest granularity in the system experience knowledge. The systems engineering project assets represent an integrated structure that captures product and process knowledge in engineering situations as an instance of loosely connected ontology modules that are held together by a general ontology for systems engineering. This general ontology is developed in domain, product, and process modules. The three levels are required to provide a comprehensive semantic model for the systems engineering project asset through an integrated representation of its semantic content, its structural content, and its design rationale.

Three Levels

The domain ontology defines the specific domain concepts, attributes, constraints, and rules. It aims to capture formally a target system according to its different abstraction levels; in other words, for each engineering domain, the ontology defines a consensual semantic network to represent domain-specific requirements, functions, behavior, and physical components, as well as their structural relation-

Auzelle *continued*

processes and standards, such as STEP (ISO 10303:233), thus proposing a methodology to consider the formal semantics relationships between the involved enterprise system and the so-called active product. This new approach will open new avenues to manage the manufacturing process, considering each active product as a system per se that may interoperate with each enterprise system, enhancing the new planning perspectives in the manufacturing world.

References

Boardman, J., and B. Sauser. 2006. System of systems: The meaning of of. In *Proceedings 2006 IEEE/SMC International Conference on System of Systems Engineering* (Los Angeles), 6. [New York]: IEEE Systems, Man, and Cybernetics Society.

ISO (International Organisation for Standardisation). 2008. *ISO/CD 10303:233: Industrial automation systems and integration; Product data representation and exchange; Part 233: Systems engineering data representation.* Geneva: ISO.

Morel, G., H. Panetto, F. Mayer, and J. P. Auzelle. 2007. System of enterprise-systems integration issues: An engineering perspective. Paper presented at the IFAC Conference on Cost Effective Automation in Networked Product Development and Manufacturing.

Morel, G., H. Panetto, M. B. Zaremba, and F. Mayer. 2003. Manufacturing enterprise control and management system engineering: Paradigms and open issues. *IFAC Annual Reviews in Control* 27 (2): 199–209.

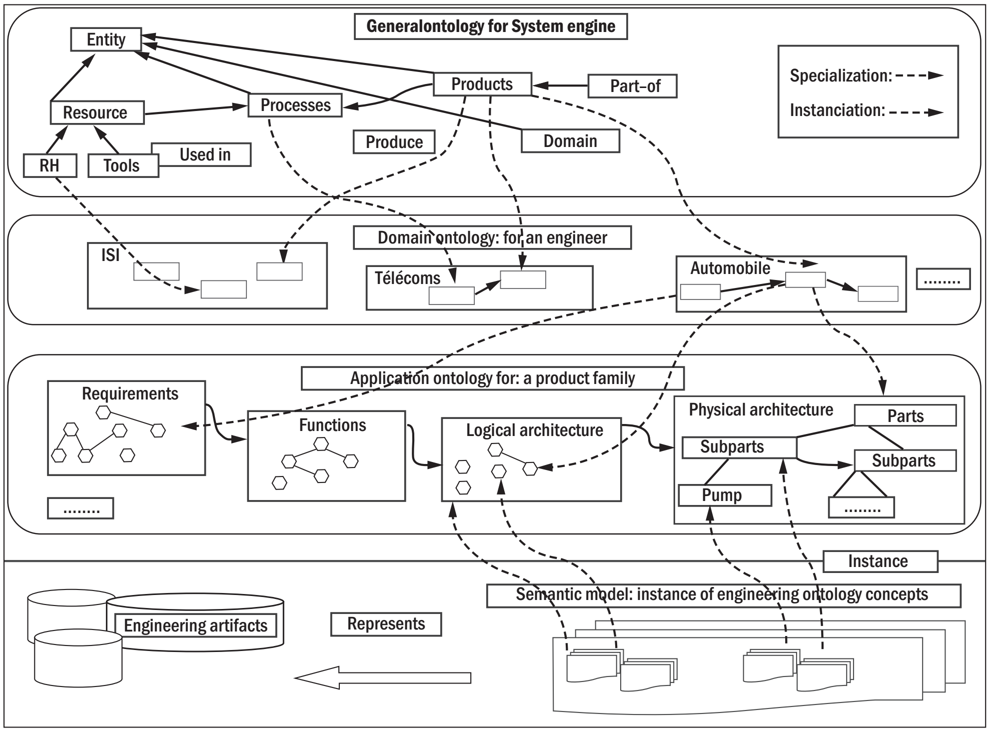


Figure 1. Layered ontologies for modeling systems engineering knowledge

ships (such as "is a part of") and their semantic relationships (such as "allocation"). For example, a domain ontology for electric circuits might define, among other things, generic types of electric components such as transistor, connection relation among components, physical laws among physical quantities, functions of components, and allocation relations between components and functions.

The product ontology contains concepts and relations that represent artifact types such as requirement documents, functional models, or conceptual schema. The product ontology provides a logical structure and basic modeling constructs to describe engineering artifacts. This means that data can be extracted from a domain ontology and packaged into an ontological, constructed, conceptual model or an engineering document. By formally relating modeling elements to domain concepts we could provide a systematic and semantic description of an engineering solution.

The process ontology contains concepts and relations that formally describe engineering activities, tasks, actors, and design rationales concepts (goals, alternatives, arguments, and justifications for engineering decisions).

Both the process and the product facets act as a formal logical structure for the systems engineering project asset. The domain facet provides semantic content for this structure.

While the ontological modules for domain, product, and process introduce general-level concepts that describe a systems engineering project asset, they need to be specialized and refined in order to provide an operational knowledge model for systems engineering projects. To this end, we introduce a layered organization of these ontological modules: a general ontology for system engineering, a specialized ontology for an engineering domain (such as automotive or information systems), and an application-specific ontology. Layers subdivide the ontology into several levels of abstraction, thus separating general knowledge from knowledge about particular domains, organizations, and projects. This allows all the engineering assets to be based on generic concepts while at the same time providing a mechanism to enable different stakeholders to define their own specific terminology and interpretation of the concepts. By instantiating the most specific ontological concepts, concrete information items can be stored in a centralized project repository. Ontological concepts act as a semantic index for engineering artifacts.

Each layer is defined along the two axes of abstraction and semantic links. Abstraction allows modeling a gradual specification of models that are more and more concrete, that is, from abstract system requirement to concrete system components. The semantic links define how the concepts within and between an ontology module are related to each other. Typical semantic links are subsumption relations, "part of" relations, and traceability relations. For example, in an ontological module for a domain, the "part of" relation could be defined on physical components assemblies and a traceability relation (allocation) could be defined to map system functions onto physical components.

References

Awad, E. M., and H. Ghaziri. 2004. *Knowledge management*. Upper Saddle River, NJ: Prentice Hall.

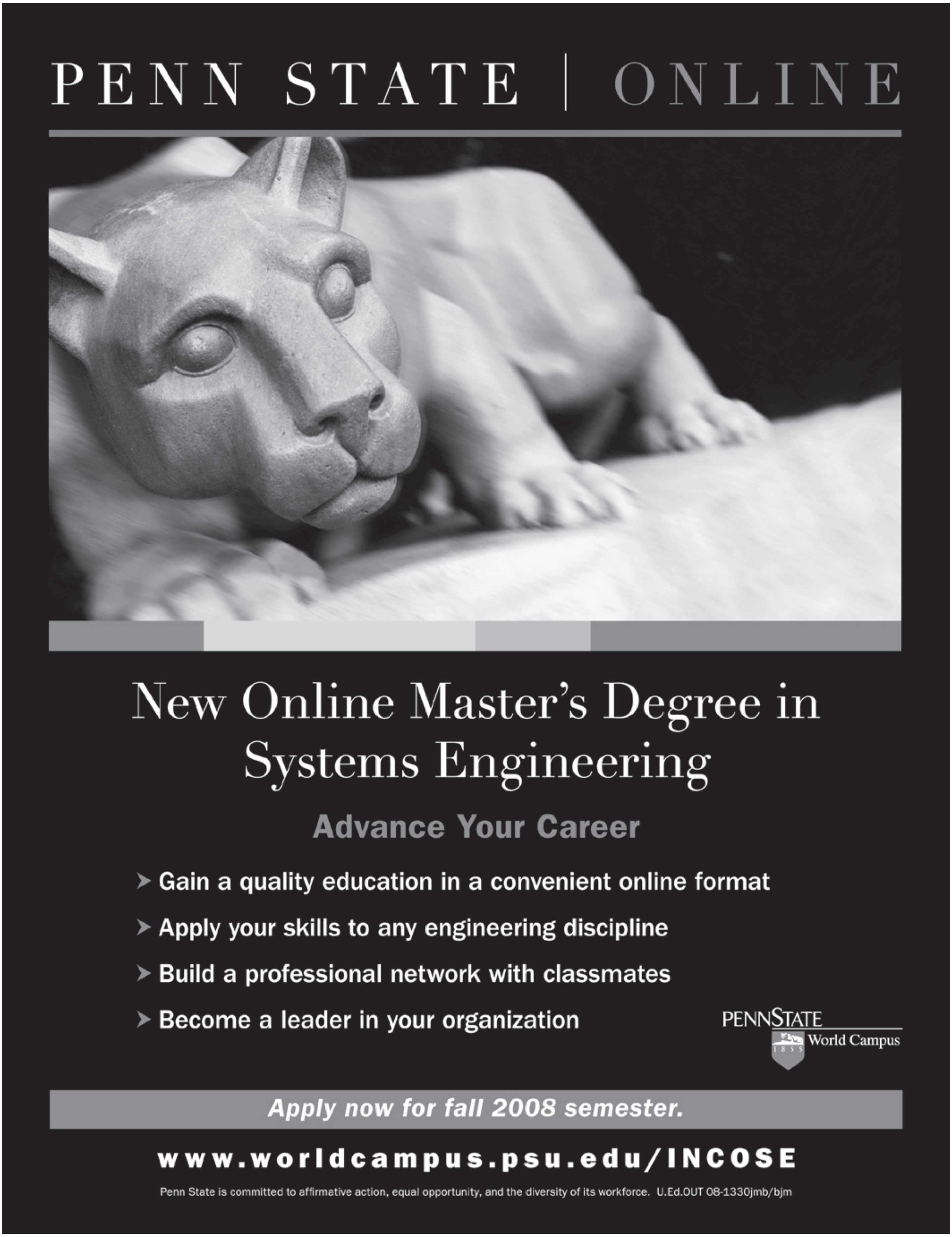
Kitamura, Y. 2006. Roles of ontologies of engineering artifacts for design knowledge modeling. In *Proceedings of the 5th International Seminar and Workshop Engineering Design in Integrated Product Development (EDIProD 2006)* (Gronów, Poland), 59–69. Also available at the Web site of the Mizoguchi Lab at Osaka University, Japan, http://www.ei.sanken.osaka-u.ac.jp/~kita/pub/documents/kita-ediprod06.pdf.

McMahon, C., A. Lowe, and S. J. Culley. 2004. Knowledge management in engineering design: Personalisation and codification. *Journal of Engineering Design* 15 (4): 307–325.

Meinadier, J.-P. 2002. *Le métier d'intégration de systèmes*. Paris: Hermès Lavoisier.

Mizoguchi, R., and Y. Kitamura. 2000. Foundation of knowledge systematization: Role of ontological engineering. In R. Roy, ed., *Industrial knowledge management: A micro-level approach*, 17–36. London: Springer.

INSIGHT


PENN STATE | ONLINE
New Online Master's Degree in Systems Engineering
Advance Your Career
Gain a quality education in a convenient online format
Apply your skills to any engineering discipline
Build a professional network with classmates
Become a leader in your organization
PENNSTATE World Campus
Apply now for fall 2008 semester.
www.worldcampus.psu.edu/INCOSE
Penn State is committed to affirmative action, equal opportunity, and the diversity of its workforce. U.Ed.OUT 08-1330jmb/bjm

Determining the Overlap between Quality Repositories and Regulatory Requirements for the Pharmaceutical Industry: Problem Formulation and Application

Romain Deguil, deguil@enstimac.fr; **Didier Gourc**, gourc@enstimac.fr; and **Hervé Pingaud**, herve.pingaud@incose.org

In the field of information systems, numerous maturity models, norms, and standards have been elaborated over the last decades (for example, Saiedian and Chennupati 1999). They are usually composed of requirements or good practices that provide guidance to professionals who develop products or services, by using totally controlled and mature processes. The regulatory requirements specific to a given business can be added to this short list. In this article, we will use the generic term *repository* to designate a coherent and structured set of good practices or requirements. It could be either a norm, a standard, a maturity model, or (of course) a set of regulatory requirements.

Considering that an organization can simultaneously maintain several repositories, it is legitimate to wonder about the overlap of these repositories. The goal of our study is to identify which practices or requirements of business repositories are necessary to satisfy the main repository required by regulatory authority. Therefore, we propose a specific formalism in order to represent relations between several repositories, and we give an exploitation methodology to determine which business practices to apply.

The Mapping of Repositories

We assumed that just two repositories have to be compared. The first one will be denoted *MAIN_REP* and the second one, *REP_2*.

We want to map *MAIN_REP* with *REP_2* (i.e., to establish all the functional equivalences between elements of the different repositories), so that we can set up relations in order to formally identify what is needed in *REP_2* to be equivalent to all the elements of *MAIN_REP* (or to a subset of it). Subsequently, we will use the word *cover* to designate this link. Moreover, we use E_i to denote the unit elements of *MAIN_REP* and P_j to indicate the unit elements of *REP_2*.

Our aim is to be able to establish links of the type 1: *N* ("one element of *MAIN_REP* and several elements of *REP_2* are linked together"). To do so, it is necessary to use some basic logical operators. We use the following:

– *AND*, which expresses the fact that several elements of *REP_2* are simultaneously needed to cover an element of *MAIN_REP*,

– *OR* inclusive, which expresses the possibility of making a choice between several elements of *REP_2* in order to respect the element of *MAIN_REP* concerned. The simultaneous realization of all the elements is allowed.

– *OR* exclusive, which expresses the need to choose between several elements of *REP_2* and whose simultaneous realization is impossible.

By using this formalism, all elements of *MAIN_REP* and *REP_2* are linked by expert analysis of repositories. If possible, each E_i is defined by a logic expression composed of P_j.

Identification of the Elements Needed

The aim is to identify what is needed in *REP_2* to cover *MAIN_REP*. We note that if E_i is covered, its value is 1; otherwise, it is 0. Therefore, the requirement that *MAIN_REP* must be covered can be explained by the following equation:

$$E_1 * E_2 * \ldots * E_{n-1} * E_n = 1$$

By substituting each E_i by the combination of P_i established previously, each product of the new expression describes a solution for the equation (e.g., a value of the *n*-tuplet $(P_1,\ldots P_j,\ldots P_m)$, with $P_j = 1$ if P_j is present in the product; otherwise $P_j = 0$. The best *n*-tuplet $(P_1,\ldots P_j,\ldots P_m)$ has to be chosen. At first, we want to cover all the elements of *MAIN_REP* and to have a minimal number of elements of *REP_2* to deploy. We just have to choose the *n*-tuplet where the number of P_j is minimal (for example, with a branch and bound method for example [Cohen 1995]). Indeed, this *n*-tuplet is the product with the minimal "dimension" (the product whose elements are the least different).

To develop this method, we have chosen to use the Perl language. A prototype supporting this method of resolution has been developed and applied to some real cases. Next, we develop an example of application to illustrate our approach and the results.

Case Study

Suppliers of computerized systems for the pharmaceutical industry represent a pertinent study case. They must comply with some regulatory requirements (called "GxP") (International Society for Pharmaceutical Engineering 2001) and may simultaneously deploy a software process improvement approach, based on the Capability Maturity Model Integration (CMMI), for example. The CMMI is composed of five levels of maturity, and lists the most important "rules on how to work" (good practices), shared by a wide range of professionals of the information technology sector (Software Engineering Institute 2002).

We applied our approach to this field (*MAIN_REP* is the GxP, E_i is a requirement or *MAIN_REP*, *REP_2* is the CMMI repository, and P_j is a good practice) in order to bring an answer to the question, "If a supplier of computerized systems of the pharmaceutical industry achieved a CMMI maturity level, to what degree does it complies with the main business regulatory requirements?" Moreover, other repositories have been integrated in our study: *ISO 20000* (based on the Information Technology Infrastructure Library) and *ISO 27000* (in order to cover all the lifecycle of a computerized system). The conclusion is that suppliers have to deploy about 90% of the practices of the level 2 of the CMMI, 50% of the practices of the level 3 and 90% of the *ISO 20000* and *ISO 27000*.

> *continues on next page*

Enterprise Interoperability Measurement: The Potentiality

Nicolas Daclin, nicolas.daclin@incose.org; Bruno Vallespir, vallespir@incose.org; and David Chen

From Interoperability Maturity Models…

Numerous models evaluate the interoperability of systems in terms of maturity. They all distinguish three categories of barriers to interoperability: a set of systems can encounter conceptual, organizational, or technological problems (C4ISR 1998; Clark and Jones 1999; Tolk and Muguira 2003). An enterprise can be considered as a system and defined by a set of elements that interact with its environment. The main objective of interoperability is to improve these interactions. Thus, it is appealing to behold how the concept of maturity can contribute to this enhancement. Furthermore, since these interoperability maturity models are focused on a system of systems (maturity between several systems) and strictly on one category, it is relevant to consider an approach that takes all categories into account, and also consider a system independently from other ones. This approach allows an enterprise to fully evaluate its own capacity to interoperate, and therefore to anticipate possible problems before a partnership. As a consequence, we introduce the concept of *potentiality.*

…Toward an Enterprise Interoperability Potentiality Model

An enterprise that has potentiality possesses intrinsic attributes related to the three categories of interoperability that allow it to interoperate more easily with another enterprise in a partnership. In other words, potentiality is an intra-enterprise evaluation without the need to know the interoperating partner. The main goal is to increase the capacity to implement interoperability and to decrease the risk of encountering problems during a partnership. Our enterprise interoperability potentiality model defines the evaluation of the potentiality of an enterprise according to the three categories that impact the development of interoperability, and the levels where interoperability takes place: business, processes, services, and data (figure 1). For each category and each level of interoperability, there are five levels that characterize potentiality: (1) the **isolated** level, which represents a total incapacity to interoperate; (2) the **initial** level, where interoperability requires strong efforts that affect the partnership; (3) the **executable** level, where interoperability is possible even if the risk of encountering problems is high; (4) the **connectable** level, where interoperability is easy even if problems can appear for distant partnership; and (5) the **interoperable** level, which considers the evolution of levels of interoperability in the enterprise, and where the risk of meeting problems is low.

The goal is to evolve throughout the levels of potentiality to reach the top one. Although this evolution is compulsory in order to decrease the risk of meeting problems and to facilitate the implementation of interoperability, maximum potentiality does not imply full interoperability. Indeed, the use of standard tools by an enterprise does not ensure that a partner will use the same ones. Hence, problems of interoperability can still appear.

Potentiality / Level	Conceptual potentiality	Organizational potentiality	Technological potentiality
Business	Isolated, Initial, Executable, Connectable, Interoperable	Isolated, Initial, Executable, Connectable, Interoperable	Isolated, Initial, Executable, Connectable, Interoperable

Figure 1. The enterprise interoperability potentiality model for the business level

References

[C4ISR.] U.S. Department of Defense C4ISR Architecture Working Group. 1998. *Levels of Information Systems Interoperability (LISI).* Washington, DC: Department of Defense.

Clark, T., and R. Jones. 1999. Organisational interoperability maturity model for C2. In *Proceedings of the 1999 Command and Control Research and Technology Symposium* (Newport, RI) (Web site: http://www.dodccrp.org/events/1999_CCRTS/toc_track.html), http://www.dodccrp.org/events/1999_CCRTS/pdf_files/track_5/049clark.pdf.

Tolk, A., and J. A. Muguira. 2003. The levels of conceptual interoperability model. Paper presented at the Fall Simulation Interoperability Workshop of the Simulation Interoperability Standards Organization, Orlando, FL. Available at the Web site of Carnegie Mellon's Software Engineering Institute, http://www.sei.cmu.edu/isis/pdfs/tolk.pdf.

Deguil *continued*

References

Cohen, V. 1995. *La recherche opérationnelle*. Paris: Presses Universitaires de France.

International Society for Pharmaceutical Engineering (ISPE). 2001. *Good automated manufacturing practice, version 4: Guide for validation of automated systems. Brussels: ISPE.*

Saiedian, H., and K. Chennupati. 1999. Towards an evaluative framework for software improvement models. *Journal of Systems and Software* 47 (Issues 2–3): 139–148.

Software Engineering Institute. 2002. *Capability Maturity Model Integration (CMMI SM), version 1.1, continuous representation.*

INSIGHT

Product Line Requirements Configuration in the Context of Multiple Models

Olfa Djebbi, olfa.djebbi@incose.org; Camille Salinesi, camille.salinesi@incose.org; and Colette Rolland, colette.rolland@univ-paris1.fr

Product-line engineering is emerging as a viable and important development paradigm that allows companies to realize order-of-magnitude improvements in time-to-market, cost, productivity, quality, and flexibility (SEI 2008). The basic concept of product-line engineering is that capitalizing on the commonality and managing the variations among products in the product line generate these incomes. As a result, the main effort to design a product from the product line relates to the analysis of the variations and of the impact of choices made for the required product.

In this context, the processes of requirements engineering have two goals: (1) to define and manage requirements within the product line, and (2) to coordinate requirements for the single products. Researchers in both academia and industry have proposed numerous techniques to model product line requirements at different levels (Djebbi and Salinesi 2006; Czarnecki and Eisenecker 2000; Halmans and Pohl 2003). On the other hand, product configuration methodologies are rather scarce and are mostly focused on the technical level (Czarnecki and Eisenecker 2000; Sinnema, Dielstra, and Hoekstra 2006). Regarding requirements engineering methods, the existing methods only allow selecting requirements from a single product line requirements model.

However, our experiences in industry show us that the configuration process is more complicated for many reasons:

- Different stakeholders (including final users) should be involved in the requirements elicitation process. We strongly believe that systematic guidance is needed to check the consistency of product line requirements with stakeholders' requirements.
- The complexity of current software development justifies the simultaneous use of several models to specify and communicate various views and aspects of a software system with regard to the involved stakeholders (e.g., executives, developers, marketing). What is needed is a systematic way to capture all the information given by the various viewpoints and to organize it so that missing information is more easily identified, the full impact of change is more easily understood, and dependencies are explicitly discerned so that configuration is facilitated.
- The perception of variability often depends on the organization in question and the area of expertise of the involved stakeholders. It is not unusual that different models are expressed by heterogeneous variability notations within a single project development (e.g., analysts/use cases or architects/features). There is clearly a need for an independent notation to bring together variability expression from various areas in a common frame of reference.

The RED-PL Approach (Requirements Elicitation and Derivation for Product Lines)

Based on these observations, we have undertaken answer the research question: "How do we configure a product in a requirements-driven way and in the presence of several product line models?"

Our approach (Djebbi and Salinesi 2007) is original in two ways: (1) a product specification is not just a choice made on alternatives offered by the product capabilities, but also a response to the several stakeholders' requirements and constraints; and (2) the requirements configuration is guided as a complex and stepwise decision-making activity.

Our variability frame of reference balances the obtained product between (1) initial requirements that can be satisfied by the product line, (2) initial requirements that, on the contrary, cannot be satisfied by the product line as it is defined; nevertheless, they should be included in the final product configuration, and (3) product line assets that should be implemented in the final configuration even though they have not been expressed originally as requirements. To enable matching and mitigation, we propose to consider the product line requirements as constraints, stakeholders' requirements as predicates, and the configuration process as a constraint resolution problem rather than a series of choices driven by variability points (figure 1). Constraints seem to be an adequate language to express several stakeholders' requirements, as well as product line capabilities. The constraint language can also be mapped into the programming languages for constraint satisfaction problems. Besides, the constraint paradigm is flexible enough to be able to deal with new needs such as the introduction of new types of requirements dependencies or the expression of richer constrains.

As shown in figure 1, we define a constraint-based language that allows one to express any product line requirements and to configure products (Djebbi and Salinesi 2007). Stakeholders' requirements, constraints, and preferences are taken into account within an interactive decision process under the form of requests. A resolution engine, implemented using techniques for constraint satisfaction problems, allows one to capture stakeholders'

> continues on next page

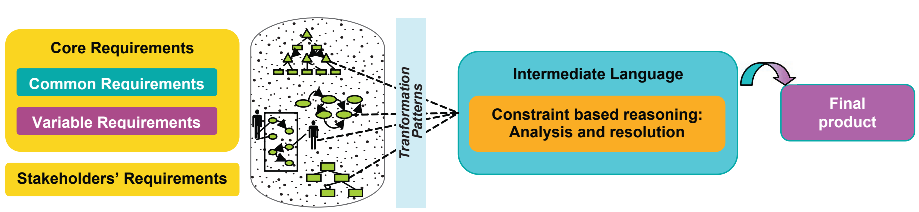


Figure 1. Constraint-based configuration overview

Djebbi *continued*

requirements, to analyze them to identify inconsistencies, and to generate propositions. When requirements are incomplete, users receive propositions showing explicitly the impact of some choices before making firm decisions. To achieve this, we propose patterns that allow systematic translation of variability models into our constraint language, independently from the language in which they were originally specified.

Our approach was implemented and validated in an industrial setting (Stago: http://www.stago.fr) by considering the requirements engineering phase of a product line of blood analyzers (Djebbi and Salinesi 2008).

References

Czarnecki, K., and U. Eisenecker. 2000. *Generative programming: Methods, tools, and applications.* Boston, MA: Addison Wesley.

Djebbi, O., and C. Salinesi. 2006. Criteria for comparing requirements variability modeling notations for product lines. Paper presented at the 4th International Workshop on Comparative Evaluation in Requirements Engineering, at the 14th IEEE International Requirements Engineering Conference, Minneapolis/St. Paul, MN.

———. 2007. RED-PL: A method for deriving product requirements from a product line requirements model. Paper presented at the 19th International Conference on Advanced Information Systems Engineering, Trondheim, Norway.

———. 2008. Towards an automatic PL requirements configuration through constraints reasoning. Paper presented at the 2nd International Workshop on Variability Modelling of Software-intensive Systems, Essen, Germany.

Djebbi, O., C. Salinesi, and D. Diaz. 2007. Deriving product line requirements: The RED-PL guidance approach. Paper presented at the 14th Asia–Pacific Software Engineering Conference, Nagoya, Japan.

Halmans, G., and K. Pohl. 2003. Communicating the variability of a software-product family to customers. *Software and Systems Modeling* 2: 15–36.

SEI (Software Engineering Institute). 2008. Software product lines. Web site of the Software Engineering Institute of Carnegie Mellon University, Pittsburgh, PA. http://www.sei.cmu.edu/productlines/.

Sinnema, M., S. Deelstra, and P. Hoekstra. 2006. The COVAMOF derivation process. Paper presented at the 9th International Conference on Software Reuse, Turin, Italy.

INSIGHT

Combining SysML and Formal Methods for Safety Requirements Verification

Dominique Evrot, dominique.evrot@incose.org; Jean-François Pétin, jeanfrancois.petin@incose.org; and Gérard Morel, gerard.morel@incose.org

This work, developed at Nancy Research Centre of Automatic Control (CRAN from Nancy University) jointly with INRS (Institut National de Recherche et de Sécurité) deals with the verification of safety requirements for the design of complex control systems involving software, mechanical, or electrical components for industrial safety-critical applications such as power plant control or embedded control systems.

The dynamic of such control systems results from an interaction network between all the system components that can introduce undesired behaviour with critical impact on safety. Consequently, safety properties of the control system cannot be verified by only proving local properties of each component, but need to be studied through the emerging behaviour that issues from the interaction network. The main difficulties of this last verification are caused by the heterogeneity of the control components' different technology domains that use proper design formalisms and tools.

During the last years, tools allowing multiple-skill approaches and a high abstraction level have been developed. SysML is a UML2.0 profile that has met the expectations of the systems engineering community about requirements modeling, behavioural and architectural designs, and the abstract representation of systems. However, this toolbox does not provide verification supports, so there is still a need for verification tools that fit with a multiple-skill approach. On the other hand, dedicated tools for modeling and verifying real-time reactive system are available using the discrete event system theory and model-checking technology.

There is a growing interest, though, in methods and tools that facilitate the validation of software-intensive automation systems. This interest becomes a legal requirement when dealing with safety-critical systems: the safety-related standard *IEC 61508* strongly recommends that formal verification methods be applied in the certification process by the suppliers, integrators, or independent external authorities, but does not define how they can be applied. The main idea is to develop an integrated methodology that combines both approaches: a system-oriented approach that identifies and formalizes safety requirements using SysML, and a skill-oriented approach for components design and verification using the formal methods of the discrete event system theory.

An engineer's central problem is to ensure that the system under control to be developed (denoted S) is compliant with the end-user requirements (denoted R). This assertion can be noted as S:=R. We must first remember that (1) requirements R are usually captured in multiple levels of abstraction and broken down into sub-requirements at different levels of abstraction, denoted $R = \{R_1, R_2, \ldots R_n\}$; and that (2) the system under control is often composed of heterogeneous subsystems (such as mechanical, software, or electronic) that cooperate to achieve the systems goal, denoted $S = \{S_1, S_2, \ldots, S_m\}$. These two facts complicate the initial problem by introducing a non-bijective relationship between the set of requirements and the set of system components (or subsystems). In other words, it means that a given component S_i may satisfy a subset of requirements, such that $S_i := \{E_k\}_{k\in[1,n]}$; and it means that a given requirement R_j may be satisfied by a subset of system components, such that $\{S_k\}_{k\in[1,n]} := R_i$. It is our objective to identify and prove this relationship is preserved, especially for the safety requirements, through the whole systems engineering process.

Two ways of thinking about this topic can be found in the scientific literature. One way is based on the properties assertion method that uses an *a posteriori* verification process that aims to prove that a given system specification, design, or implementation fulfills a given model of safety requirements. An important technique in this type of work is model checking (Clarke, Grumberg, and Peled 2000). Limits are linked to the size of the models due to explosion phenomena and to the identification and formalization of the properties to be proved. The other way of thinking is based on a refinement method that uses an *a priori* verification process that aims to progressively enrich an initial set of abstract requirements to produce a concrete model of the system that satisfies, by construction, the set of requirements (Pétin, Morel, and Panetto 2006). Formal refinement mechanisms such as those supported by the "B method" (Abrial 1996) belong to this kind of approach.

In this way, we have proposed an integrated method that aims to formally identify and refine the safety requirements of a complex control system by using the SysML requirements diagram, to allocate the requirements on a given set of system components using a meta-model of requirements traceability concepts, to define the impact of this allocation in terms of safety properties to be satisfied by each of the system components, and then to provide the existing model-checkers with this correct model of component properties.

Requirements Refinement, Allocation, and Projection with SysML

To design a component's architecture that fulfills the system's safety requirements, we need a set of more detailed specifications related to the safety control functions or the material and immaterial barriers to be used. We based our approach on refinement rationale in the same idea than the "B" formal refinement mechanism. This mechanism substitutes a set of more concrete requirements for an abstract requirement. In the SysML Requirements diagram, this operation is represented by implication

> *continues on next page*

Evrot *continued*

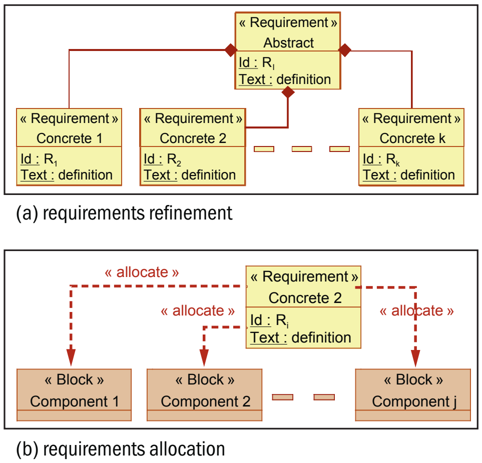


Figure 1. Requirements modeling using SysML

or composition links, meaning that if the set of concrete requirements is fulfilled, then the abstract requirement is also fulfilled (figure 1a). A requirement can refine two upper requirements. Requirements are modelled with a class stereotype that includes an open list of attributes like the requirement text definition, source, identification, and so on. This set of requirements must then be allocated onto system functions or components. Functions are modelled using the Activity SysML diagram while system architecture and components are modeled using the Block and Block Definition SysML diagrams. These diagrams allow the engineer to trace the requirements that system functions and components are expected to fulfill (figure 1b).

In order to provide proofs about requirements refinement and allocation, we need a more formal description of the different safety properties, associated to each abstract and concrete requirements, denoted respectively P_{aj} and P_{ci}. In this way, we have modified the SysML meta-model to introduce logic properties that formalize the textual description of requirements (see figure 2). This class stereotype allows the engineer to describe the properties in terms of logic predicates or numerical parameters. This extension enables an engineer to make the link between system modeling and trade-oriented design with the objective to prove that behaviour of a given component is compliant with local expected properties; in this way, concrete properties P_i of each components can be reused as logic or temporal predicates by model checking tools, such as UPPAAL[1], or can be reused as post-conditions by simulation and testing tools. The extension also allows the engineer to formally demonstrate that the combination of all of the components' concrete properties P_{ci} establishes the system's abstract properties P_{aj}, using a theorem prover, such as COQ (Huet, Khan, and Paulin-Mohring 1995).

This approach has been implemented (figure 2b) through the definition of an XML format that enables the exchange of data and models between different commercial tools, such as those designed for system specification with SysML (Magic Draw), for control system design and simulation using trade-oriented languages (ControlBuild), and for proving safety properties (UPPAAL model checker) using XSLT transformations.

1. Magic Draw is a product of No Magic, Inc.; ControlBuild is a product of Geensys; UPPAAL is developed by the Information Technology Department of Uppsala University (Sweden) and the Computer Science Department of Aalborg University (Denmark).

Ongoing Work

We have proposed a system modeling approach that combines non-formal methods, which are necessary to capture and formalize system requirements, and formal methods, which are required by safety-related standard such as *IEC 61508*, to prove that the local behavior of each system component contributes to satisfying system properties. Even if the end result is clearly related to software control, the approach does not focus on this point of view but tries to deal with all system technical components in a systems engineering context. This point is very helpful for the development of COTS-based (commercial off-the-shelf) control, where subcontractors' requirements must be clearly identified. Although these interdisciplinary exchanges between computer science and system engineering approaches demonstrate that they contribute to verify the highest Safety Integrity Levels (*IEC 61508*), common experiments on laboratory-scale and industrial-scale case-studies emphasize that effort must still be employed to make the proposed engineering framework effective in practice.

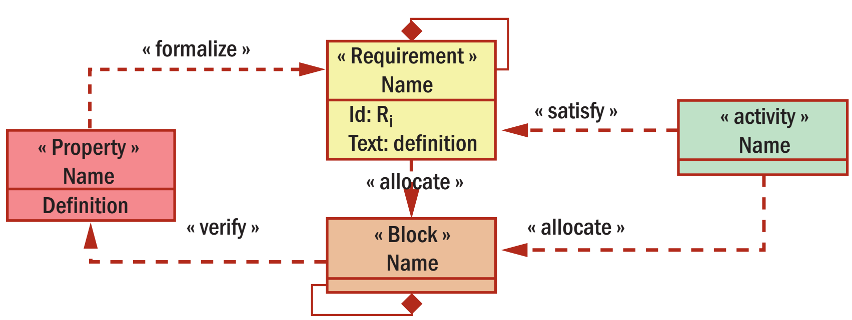


Figure 2. SysML Requirements traceability

References

Abrial, J.R. 1996. *The B-book: Assigning programs to meanings.* Cambridge, U.K.: Cambridge University Press.

Clarke, E. M., O. Grumberg, and D. A. Peled. 2000. *Model checking.* Cambridge, MA: MIT Press.

Huet, G., G. Khan, and C. Paulin-Mohring. 1995. *The Coq proof assistant: A tutorial.* INRIA (French National Institute for Research in Computer Science and Control) Technical Report 178. Rocquencourt, France: INRIA.

Pétin, J.-F., G. Morel, and H. Panetto. 2006. Formal specification method for production systems automation. *European Journal of Control* 12 (2): 115–130

Implementing a Quality Reference and Interoperability of Processes in Software Collaborative Projects

Anis Ferchichi, anis.ferchichi@incose.org; Jean-Pierre Bourey, jeanpierre.bourey@incose.org; Michel Bigand; and Hervé Lefebvre.

In this paper, we present how to integrate several processes using a common reference frame offering various viewpoints. This step is applied to the integration of two standards of quality—*ISO 9001:2000* and Capability Maturity Model Integration (CMMI)—in order to generate a quality reference frame with multiple views which allows certification relative to the two standards. This reference frame takes into account the chapters of the ISO standard and the recommendations of CMMI.

Quality Reference Standards Integration

Figure 1 shows the cartography of the enterprise processes, and particularly the level of integration of CMMI process areas (that is, a cluster of related practices in an area). The process areas for decision analysis and resolution (DAR), integrated teaming (IT), organizational environment for integration (OEI), integrated supplier management (ISM), and risk management (RSKM) are not treated by the ISO standard. All the rest of the process areas are localized in our ISO procedures.

Interoperability of Processes

Basing on our integrated quality reference, we are implementing interoperability between our company's processes and those of its clients and suppliers. We have created a process, in compliance with the recommendations of both standards, named "PROCES-SUPP-007: Manage the agreement with clients and suppliers." This process is the interface between the internal processes of the company and those of its clients and suppliers. It permits us to select potential contractors, to control the quality of their deliverables, and to define precisely which information must be exchanged.

References

Basque, R. 2004. *CMMI: Un itinéraire fléché vers le Capability Maturity Model.* Paris: Dunod.

Mutafelija, B., and H. Stromberg. 2003. Systematic process improvement using *ISO 9001:2000* and CMMI. Boston, MA: Artech House.

Bourey, J.-P, R. Grangel, G. Doumeingts, and A. Berre. 2007. Deliverable DTG2.3: Report on model driven interoperability. INTEROP.

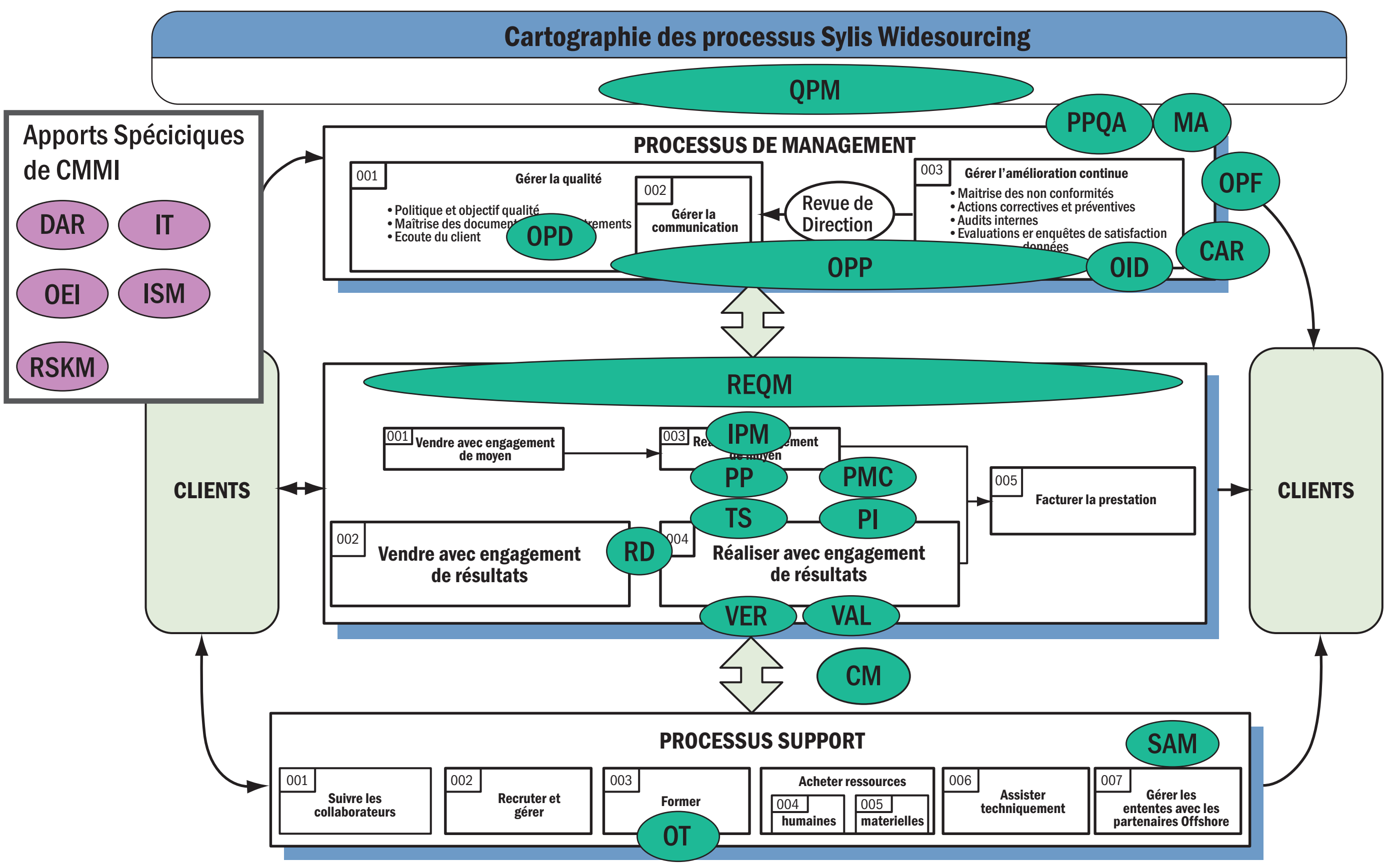


Figure 1. Enterprise's processes cartography integrating CMMI's process areas and ISO 9001:2000

Introducing Multicriteria Decision-Making into Software Engineering

Elena Kornyshova, elena.kornyshova@incose.org; and Camille Salinesi, camille.salinesi@incose.org

All software methodologies include steps where choices must be made. In practice, these choices are made in an intuitive way, based on an engineer's experience and expertise. Some methodologies offer a way to guide engineers through these choices. However, in most cases, the guideline is a simple description. On the other hand, organizations are expressing a need for rational decision-making to help them carry out choices in software engineering activities, to have detailed guidelines for applying multicriteria methods, to have a tool that is accessible to "non-specialists," to consider the specificity of a given situation, to support decisions based on multiple criteria. Our research asks the question, is "Which systematic approach should be used to support decision-making in software engineering?" Our aim is to propose a methodological support that guides software engineers through tactical choices with the application of multicriteria methods.

These are several well known multicriteria methods: multiattribute utility theory (MAUT), analytical hierarchy process (AHP), outranking methods, and weighting "methinks," among others. The goal of these methods is to define priorities between alternatives (actions, scenarios, projects) according to multiple criteria. In these methods, the decision-making steps are defined as follows: (1) diagnosing the problem, (2) identifying the problem's parameters (alternatives, criteria), (3) partial estimations of the alternatives, and (4) definition of priorities.

In our view, decision-making situations can be classified according to the number of criteria and of decision-makers in the following manner (figure 1): a monocriterion situation with one decision maker; a monocriterion situation with multiple decision-makers; a multicriteria situation with one decision maker; a multicriteria situation corresponding to multiple decision makers having each his own criterion; and a multicriteria situation corresponding to multiple decision makers having each multiple criteria.

The first situation presents a monocriterion problem and can be resolved as an optimization task. The other types are more widespread. For these kinds of problems, one can use multicriteria methods. These problems are presented in different disciplines of the software engineering and of connected domains: in requirements engineering, software design, tool selection, commercial off-the-shelf technology selection, enterprise architecture, business process management, portfolio management, and so on.

We propose an approach that helps selecting multicriteria methods and integrating them into software engineering methods to support decision-making (Kornyshova, Deneckère, and Salinesi 2008). Our approach is based on method engineering principles. This approach guides the following steps: specifying the decision-making situation, selecting an appropriate multicriteria method, and integrating it into existing methodologies. The decision-making situation is specified using pre-defined attributes such as the nature and number of alternatives, criteria data type, measure scale, and so on. The selection of a multicriteria method can be achieved by multicriteria search or by weighting and the multicriteria methods integration by assembly or extension.

We are developing a tool called Referee as proof of concept of our approach. Referee is a repository of multicriteria methods and it has the following features (figure 2):

- *Selection*. It offers guidance for choosing a multicriteria method or its fragment that better matches the project needs.
- *Adaptation*. It helps the engineer to adapt the selected method or fragment to the project specificities.
- *Guidance*. It offers a description of multicriteria methods and their fragments in order to guide the engineer to use the available methods.

Our method was evaluated using several case studies: within the software development domain for prioritizing use cases, tools, and risks; within the method engineering domain for selecting method fragments (Kornyshova, Deneckère, and Salinesi 2007); and within the business process management domain for prioritizing business processes to improve information systems security (Salinesi and Kornyshova 2006) and business process re-engineering (Kornyshova and Salinesi 2007). Our research perspectives include defining the underlying approach, developing a tool, and validating and evaluating the approach.

> *continues on page 26*

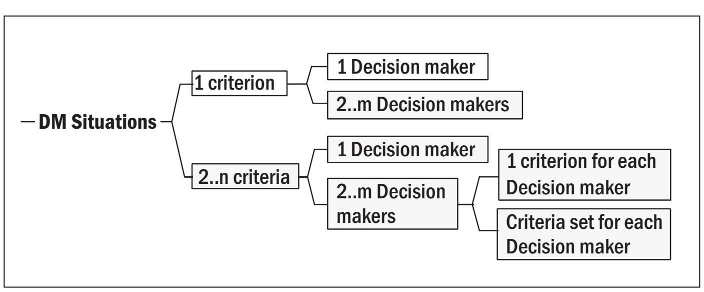


Figure 1. Typology of decision-making situations

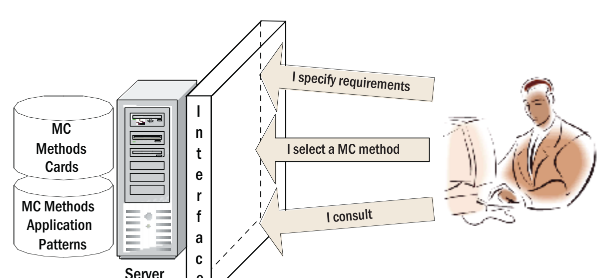


Figure 2. Tool for supporting the suggested approach

Methodology for a Probabilistic Risk Analysis of Socio-Technical Systems

Aurélie Léger, aurelie.leger@incose.org; Eric Levrat, eric.levrat@incose.org; Philippe Weber; Benoît Iung, benoit.iung@incose.org; Carole Duval; and Régis Farret

All industrial system operation implies different kinds of actors that have different kinds of behaviors and interactions. Indeed, the technical system evolves because of regular human interventions, themselves conditioned by organizational decisions. The risk analysis of such complex systems, called socio-technical systems, should therefore consider all of these dimensions in order to reveal their possible influences on the occurrence of risky situations. This issue becomes increasingly important for critical systems, since recent studies have revealed deep causes (beyond immediate technological failures) coming from human and organizational dimensions (Nivolianitou, Konstandinidou, and Michalis, 2006). We propose to develop a methodology supporting a probabilistic risk analysis for such a system. The goal of this methodology is to address this kind of multi-viewpoint analysis in an integrated way (i.e., in a same-risk model) in order to estimate the occurrence of risky scenarios and the impacts of safety barriers. Indeed, safety barriers are considered as key elements in the risk prevention field because of their critical position in the system operation. Hence, human and organizational influences are studied through these system safety elements. For these reasons, we propose to accomplish this kind of analysis by stages (figure 1).

This research results from PhD dissertation research, achieved in collaboration with the Research Center for Automatic Control, a research and development center of the French Electricity Board's nuclear branch and the French National Institute for Industrial Environment and Risks.

The preliminary stage consists of defining system limits by specifying the system being considered and its contextual dimensions. Then it becomes possible to identify convenient methods that can be used to collect information in each system dimension (knowledge extraction) and to propose a shared representation of these different kinds of information that then enables their aggregation in a risk model (knowledge unification). These first three stages can be facilitated through the definition of a "conceptual framework" (figure 2, based on Paté-Cornell and Murphy 1996).

In our approach, the system is broken down into three representative layers that interact through horizontal and vertical exchanges: the technical layer, the human layer, and the organizational layer. This system is then influenced, through transactional exchanges, by external constraints: the organizational and the natural environment contexts.

For the technical layer, the objective is to represent the occurrence of accident scenarios. Thus we use the "bow-tie" method, developed in the European project ARAMIS (www.aramis.jrc.it), which describes this kind of scenarios from initiators to final consequences and accounts for safety barriers.

For the human layer, the aim is to represent human actions, such as maintenance and control actions, that have an impact on the technical system, and, more specifically, on safety barrier components. We use human reliability principles (Hollnagel 1993), which regard human actions not only as a source of errors but also as a source of performances, to estimate the effectiveness of these actions.

For the organizational layer, the objective is to represent organizational factors that can occur in accidents as defined by Pierlot, Dien, and Llory (2007): "In fact, for determining if an organization is in good health, it is far simpler… to define a set of pathogenic organizational factors than to exhaustively list the organizational factors required and sufficient to ensure a good safety level within an organization."

These different kinds of knowledge (probabilistic knowledge for technical and human dimensions and deterministic knowledge for the organizational dimension) are then merged into a same-risk model (risk model construction) by using Bayesian networks, through a modeling of safety barriers operation (generic barrier models, figure 3) as developed by Léger, Duval and others

> continues on next page

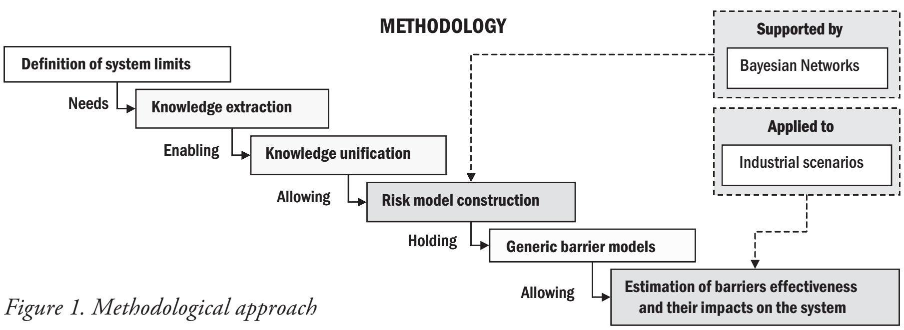


Figure 1. Methodological approach

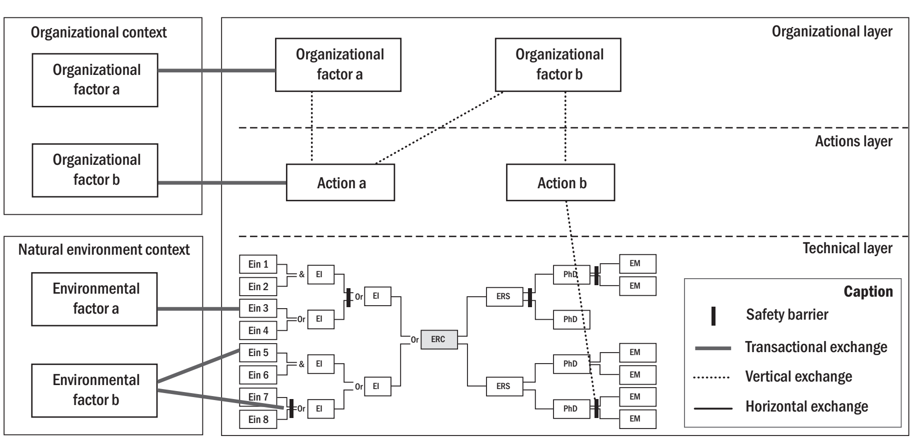


Figure 2. Conceptual framework

Leger *continued from page 25*

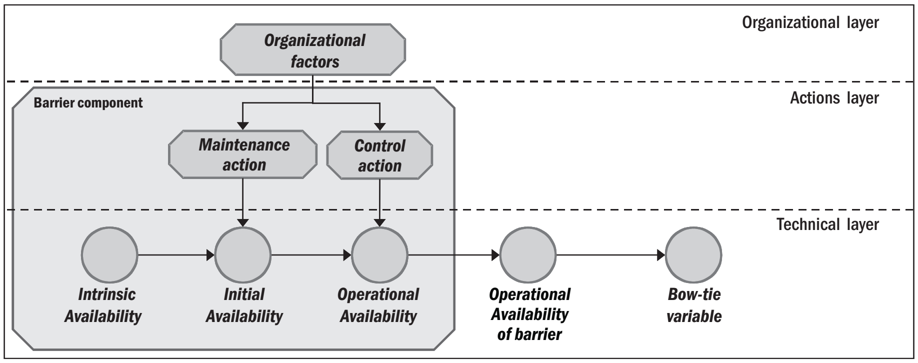


Figure 3. Barrier component model

(2008) and by Léger, Farret and others (2008).

Finally we apply this methodology to industrial cases collectively with the French Electricity Board's nuclear branch and the French National Institute for Industrial Environment and Risks in order to validate it and to propose, if necessary, improvements.

References

Hollnagel, E. 1993. Human reliability analysis: Context and control. London: Academic Press, 1993.

Léger, A., C. Duval, R. Farret, P. Weber, E. Levrat, and B. Iung. 2008. Modeling of human and organizational impacts for system risk analyses. Paper presented at the 9th International Probabilistic Safety Assessment and Management Conference, Hong Kong, People's Republic of China.

Léger, A., R. Farret, C. Duval, E. Levrat, P. Weber and B. Iung. 2008. A safety barriers-based approach for the risk analysis of socio-technical systems. Paper presented at the 17th IFAC (International Federation for Automatic Control) World Congress.

Nivolianitou, Z., M. Konstandinidou, and C. Michalis. 2006. Statistical analysis of major accidents in petrochemical industry notified to the major accident reporting system. *Journal of Hazardous Materials* 137 (1): 1–7.

Paté-Cornell, M. E., and D. M. Murphy. 1996. Human and management factors in probabilistic risk analysis: The SAM approach and observations from recent applications. *Reliability Engineering and System Safety* 53: 115–126.

Pierlot, S., Y. Dien, and M. Llory. 2007. From organizational factors to an organizational diagnosis of the safety. Paper presented at the ESREL 2007 Safety and Reliability Conference, Stavanger, Norway.

Kornyshova *continued from page 26*

References

Kornyshova, E., R. Deneckère, and C. Salinesi. 2008. Using multicriteria decision-making to take into account the situation in system engineering. Paper presented at the International Conference on Advanced Information Systems Engineering, Montpellier, France.

_______. 2007. Method chunks selection by multicriteria techniques: An extension of the assembly-based approach. Paper presented at the International Conference on Situational Method Engineering, Geneva, Switzerland.

Kornyshova, E., and C. Salinesi. 2007. Business process priorisation with multicriteria methods: Case of business process reengineering. Paper presented at the International Conference on Enterprise Information Systems, Funchal, Portugal.

Salinesi, C., and E. Kornyshova. 2006. Choosing a prioritization method: Case of IS security improvement. Paper presented at the International Conference on Advanced information Systems Engineering, Luxembourg, Luxembourg.

Regeneration Engineering for Assessing Weapon System Availability

Maxime Monnin, maxime.monnin@incose.org; Benoit Iung, benoit.iung@incose.org; and Olivier Senechal

These days, controlling the availability of a system is a key factor in industry, which makes dependability important as well. Many systems that perform critical missions have to function in hostile environments where operational availability can be affected by internal system failures and external factors such as damage. This is increasingly the case for weapon systems that operate in a battle context. Accomplishing the mission is thus directly linked to system reliability, vulnerability, and regenerability. This last item, system regenerability, is defined as the capacity of a system to recover operational capabilities after failure or damage; this characteristic has become a requirement in the design of weapon systems. According to recent work supported by the Resilience Engineering Network (http://www.resilience-engineering.org/intro.htm), system regeneration contributes to adaptive capacities of systems to enhance system resilience. Traditionally, system dependability has focused on internal causes (e.g., failures), while system survivability has focused on external factors such as damage to the system. These two types of studies tend to be considered separately. However, working on system regeneration in order to improve system availability implies assessing the impact of both failure and damage to the system. Research considering both failure and damage is scarce, and as Campbell and Starbuck have mentioned (2005), there are currently no methods for modeling or simulation that allow the impact of regeneration actions to be assessed dynamically.

To deal with this problem, we have previously proposed a unified, multi-step failure/damage modeling approach (Monnin, Senechal, Iung, Lelan, and Garrivet 2007), developed in partnership with NEXTER Group, a French weapons systems manufacturer, and the French Procurement Agency. When following a systems engineering process, the regenerability potential of new systems must be assessed in the design phase during dependability studies, but tools and methods are still needed for both the modeling and evaluation processes. In this way, during Mr. Monnin's PhD research, we developed a technique of regeneration engineering that provides systems designers a modeling methodology to assess operational availability. The method extends notions of dependability studies to allow failure, damage, and regeneration to be taken into account in a unified way. We propose to analyze the following factors in the same way: (1) the functional impact of failure (failure mode and effect analysis), (2) both the functional and the physical impact of aggression (damage mode effect analysis) and (3) the functional impact of regeneration. Indeed, the way a system or component can be regenerated strongly depends on both its functional and physical state. In order to formalize the knowledge related to the system itself and the extended dependability studies, we developed a static model called the Structural Model, derived from the architecture view of the systems engineering data model of AFIS, the French chapter of INCOSE. The structural model is based on relationships defined according to three modeling axes, namely, the decomposition axis, the interaction axis, and the contribution axis (figure 1a). It is then refined according to both the application handled and the precision level required. The refinement allows the engineer to define specific regeneration patterns from the system description: for example, a component's interaction is refined to accommodate the system topology (i.e., the component's location) in order to allow damage propagation to be considered (figure 1b). Once the refined structural model is obtained, it is instantiated in a database in order to provide the description of a specific architecture (figure 2).

> continues on next page

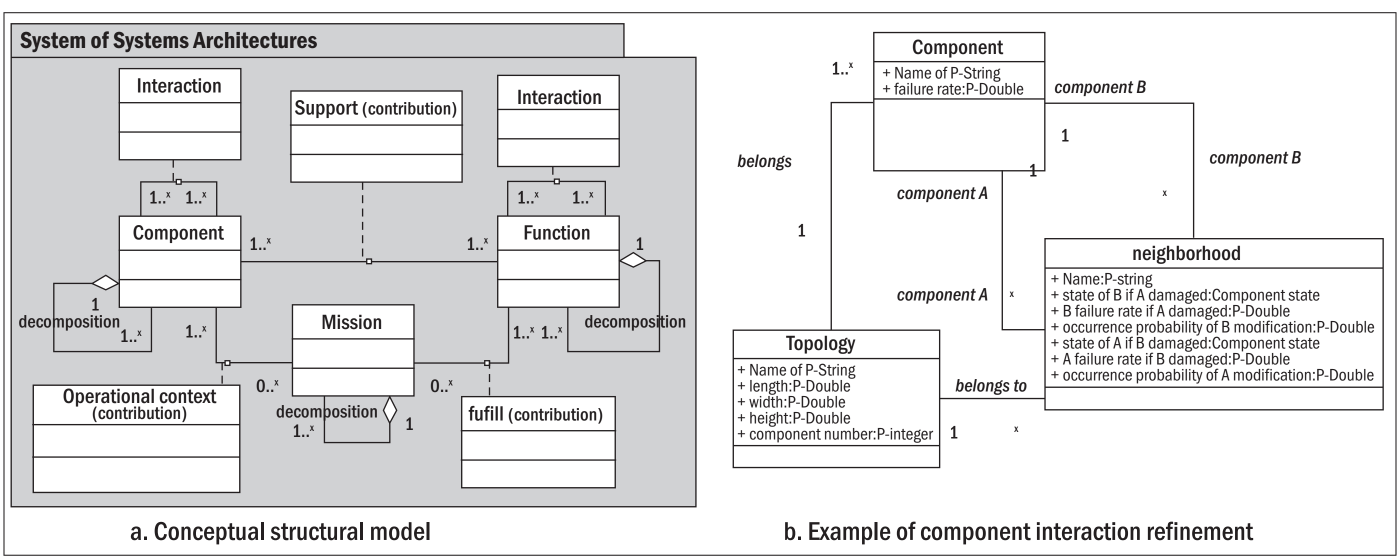


Figure 1. Conceptual view of the structural model and refinement

Monnin *continued*

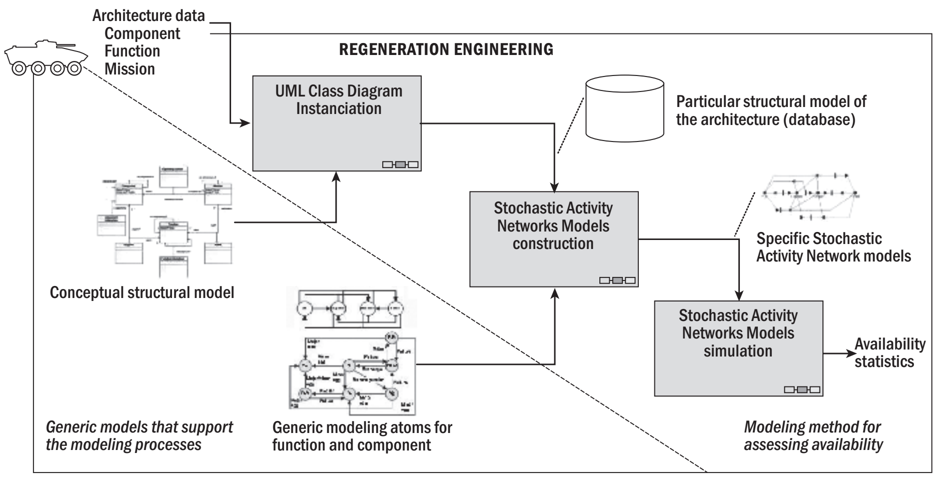


Figure 2. Process diagram of an application of the regeneration engineering modeling method

To support assessing availability, we define a dynamic model based on state-space modeling using stochastic activity networks. Generic modeling atoms representing the dynamic behavior of components are aggregated to catch the behavior of the overall system. These atoms allow the failed and damage state to be represented in a unified way for assessing availability. We use discrete-events simulations to evaluate the dynamic model and to obtain availability statistics. Since the structural model holds the knowledge that will be needed for assessing availability, the dynamic model is built using construction rules that we derived from the structural model. We have developed specific applications with our partners NEXTER and the French Procurement Agency to show the added value and the feasibility of our modeling approach (Monnin, Senechal, Iung, and Lelan 2008). Since the French Procurement Agency is currently developing architectures for systems of systems, it is necessary to carry out further implementations of those architectures to first highlight the gaps, and then define future prospects for filling the gaps.

References

Campbell, C. B., and D. W. Starbuck. 2005. Methodology for predicting recoverability. Paper presented to the American Society of Naval Engineers' Reconfiguration and Survivability Symposium, Atlantic Beach, FL.

Monnin, M., O. Senechal, B. Iung, and P. Lelan. 2008. Dynamic model for assessing impact of regeneration actions on system availability: Application to weapon systems. In *Proceedings of the 54th Reliability and Maintainability Symposium* (Las Vegas, NV). Los Alamitos, CA: IEEE.

Monnin, M., O. Senechal, B. Iung, P. Lelan, and M. Garrivet. 2007. A unified failure/damage approach to battle damage regeneration: Application to ground military systems. In *Proceedings from the 6th IFAC SAFEPROCESS 2006* (Beijing, People's Republic of China), vol. 2, 348–353. Amsterdam: Elsevier Ltd.

Modeling Strategic Alignment with the INSTAL Method

Laure-Hélène Thevenet, laure-helene.thevenet@malix.univ-paris1.fr; **Colette Rolland**, rolland@univ-paris1.fr, **and Camille Salinesi**, camille.salinesi@incose.org

The alignment issue is an important concern in systems and information systems engineering. System alignment aims at ensuring that systems are consistent with different items such as business processes, legislation, and strategy. In this paper, we focus primarily on alignment between information systems and an organization's strategy, but the principles described here can be applied to deal with alignment in other contexts. We consider alignment to be an engineering activity. As such, dealing with alignment requires us to model it and analyze it in order to improve it.

Alignment implies a relationship between several entities. Although alignment is often mentioned, it is rarely considered as a concept and modeled. We think it is inevitable to model alignment in order to be able to reason with it. Figure 1 illustrates (in the left part) the complexity of modeling alignment. Our approach indicates that correspondence links can exist between strategic elements (e.g., strategic objectives and strategic documentation such as business models, business plans, and roadmaps) and operational elements (e.g., systems or business processes). These links are simple, and therefore can be represented in a matrix. However, experience shows that alignment is often more complex than that: for example, elements are defined at different granularity levels, strategic elements are often cross-cutting to operational elements and lead to the combinatorial explosion of links, or we do not know what to align. We believe that an intentional approach (that is, a goal-oriented approach) is needed to deal with the practical difficulties generated by significant increase of complexity.

A general overview of the INSTAL (Intentional Strategic Alignment)

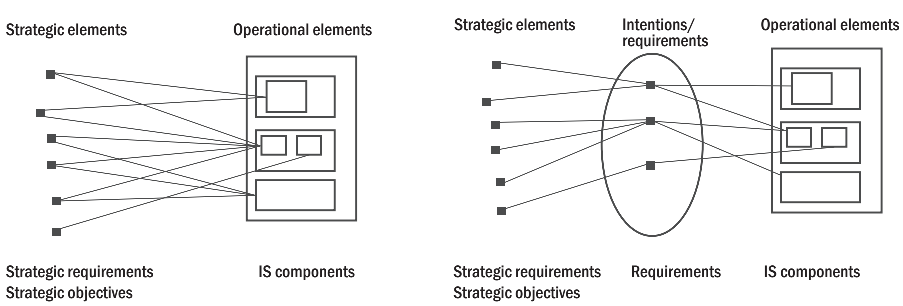


Figure 1. Correspondence links (at left) and Intention-based Alignment (at right)

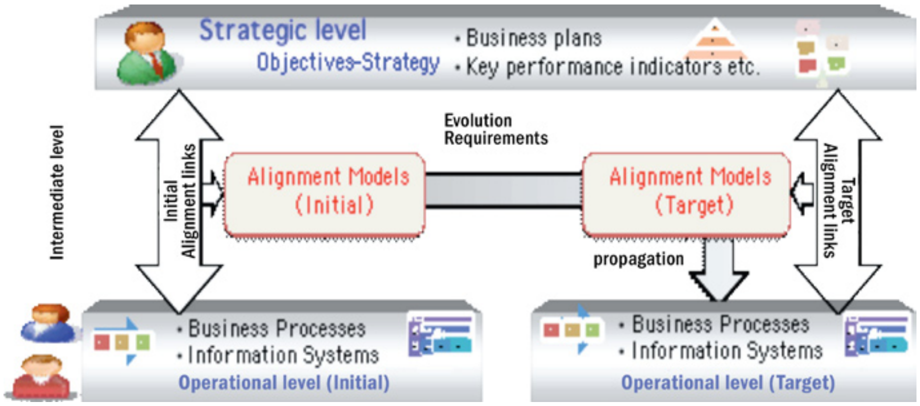


Figure 2. INSTAL method overview

method is shown in figure 2. An essential hypothesis of the method is that we do not control evolution of the strategy defined by managers and so we are not authorized to modify it. INSTAL proposes to guide the evolution of the operational level (processes for both information systems and business).

As the figure shows, the intermediate level is composed of intermediate models and strategic alignment links. The intermediate models (strategic alignment models) allows the engineer (1) to link the elements of the strategic and operational levels, (2) to model the strategic alignment and diagnose the current ("as is") strategic alignment, (3) to guide the discovery of evolution requirements, and (4) to outline the target ("to be") strategic alignment.

The intermediate models provide an ideal view of the intended strategic alignment in focusing on intentions (or goals), these being shared between the strategic and operational level. Strategic alignment links aim to highlight how operational elements do or do not contribute to the strategy. These links can be complex, implying more than one operational element and more than one strategic element.

Each link is related to an intention of the intermediate intentional models. In fact, an alignment link can only exist between entities that share the same intention. Thus, intentional models and alignment links are connected and play a complementary role in strategic alignment analysis. The intentional model proposes an intentional view of alignment and allows the engineer to organize alignment links with the refinement mechanism.

The INSTAL methodological process is composed of three steps: (1) diagnose strategic alignment, (2) discover evolution requirements, and (3) propagate and validate the evolution requirements.

1. *Diagnose strategic alignment.* INSTAL proposes to define strategic alignment models complied with the alignment requirements. Each strategic alignment link is linked to an intention of these models and identifies the elements at the strategic and operational levels (and their roles in the alignment link, i.e., how they do or do not contribute to the alignment link).
2. *Discover evolution requirements.* Obtaining a "to be" alignment that complies with alignment requirements can require a number of changes at the operational level. Evolution requirements can appear when alignment models and links are defined. Metrics are associated with strategic elements, and measures are associated with operational elements in the alignment links, in order to evaluate in a quantitative manner if the alignment is satisfied.
3. *Propagate and validate evolution requirements.* Evolution requirements are specified, validated, and prioritized if needed before being applied on the operational elements.

More details can be found on the INSTAL method in the references below.

References

Thevenet, L.-H., and C. Salinesi. 2007. Aligning IS to organization's strategy: The INSTAL method. Paper presented at the annual International Conference on Advanced Information Systems Engineering (CAISE), Trondheim, Norway.

Viscusi, G., L.-H. Thevenet, and C. Salinesi. 2008. Strategic alignment in the context of e-services: An empirical investigation of the INSTAL approach with the Italian e-government initiative case study. Paper presented at the annual International Conference on Advanced Information Systems Engineering (CAISE), Montpellier, France.

Managing Product Specifications' Dependencies in Product Development Systems

Mohamed Zied Ouertani, mzo21@cam.ac.uk

The product development process is a complex process composed of an integrated set of tasks that collectively accomplish a defined objective, i.e., collaborative product design (Browning, Fricke, and Negele 2006). Although in most product design processes, collaboration entails clear communication between designers, the real reason for this collaboration is not for communication but for resolving dependencies between product specifications (Wang and Jin 1999). Design is constraint-oriented, and comprises many interdependent parts. A change in one part may have consequences for another part, and designers cannot always oversee these interdependencies and consequences. Many engineering changes, especially when they are late, are very costly for any product development project. Engineering changes consume one-third to one half of the total engineering capacity and represent 20-50% of total tool costs (Terwiesch and Loch 1999).

Current tools for managing product data are not able to overcome all the specifications' dependencies, especially when they are not trivial and explicit. Several researchers, such as Browning (2001) and Bustnay and Ben-Asher (2005), have investigated specifications' dependencies. Most have proposed a representation of specifications' dependencies using systems engineering tools to improve understanding and analysis of these dependencies, such as the Design Structure Matrix (Browning) or N2 (Bustnay and Ben-Asher). However, none of this research has shown how to obtain these representations, nor proposed any

> continues on next page

Ouertani *continued*

mechanisms to identify dependencies. These studies have addressed the specifications involved in a design process that has already been completed, whereas the usefulness of identifying specifications' dependencies is primarily during the design progress, to help designers to perform their activities and resolve interdependency issues. Moreover, all studies reported to date have only investigated the case of two dependent design activities belonging to the same decomposition level of the design process, in a situation where a higher-level activity feeds specifications to lower-level activities. During my doctoral studies at the Research Center for Automatic Control in Nancy, France, I collaborated with a turbocharger manufacturer to address these issues.

I developed a solution called DEPNET (product specifications' DEPendencies NETwork) to explicitly capture product specification dependencies, insert them in a dependency network that is maintained throughout the design process, and assist designers in resolving dependencies during the design process, particularly when conflicts or engineering changes occur. The DEPNET solution differs from the previous research efforts in three major aspects: (1) it proposes a method to identify specification dependencies and defines concepts to qualify the discovered dependencies, (2) it accounts for the predefined specifications as well as the emerging specifications resulting from non-planned activities, and (3) it considers the design process in a more realistic way and seeks to identify product specification dependencies among sets of dependent activities belonging to various decomposition levels.

The main concept behind the proposed solution is the dependencies network, which is an oriented graph composed of nodes (referring to the product specifications handled during the design process) and arcs (corresponding to the dependency relationships between these specifications). The product specifications correspond to the various product descriptions elaborated by designers during the development process. They can be, among many others, structural, geometrical, functional, and behavioral. As for the dependency relationships, they correspond to the input/output links emerging among product specifications, after an activity produces an output specification based on an input specification. After a review of the academic literature and industrial practices, I identified two types of dependencies: forward dependency and feedback dependency. Forward-dependent specifications are those that require input from other activities, but not from themselves, while feedback-dependent specifications are those that need input from other activities, including from themselves. Some part of the dependency network may therefore be cyclic.

I distinguished different kinds of dependency relationships: redundancy, consistency, completeness, variability and sensitivity.

Two product specifications are said to be redundant when both of them describe the same entity and are expressed differently. This could occur when the product specifications belong to different product model views. We deal with consistency when two product specifications do obey some prescribed relationship between them. This relationship between descriptions can be expressed as a constraint, against which specifications can be checked.

Completeness, variability, and sensitivity describe how the dependency relationship behaves if the input specification changes after being released. The completeness is used to draw the actual product specification variation interval. The variability expresses the likelihood that the output specification provided by one task would change after being initially released, while the sensitivity describes the degree to which work is changed as the result of absorbing a transferred product specification. Since they are complementary, the last three attributes (i.e., completeness, variability, and sensitivity) were aggregated to form one criterion expressing the dependency degree.

After I defined the different concepts to qualify a dependency relationship between two product specifications, I developed a traceability prototype to (1) keep track of the progression of product development, and (2) to extract the dependencies network. In order to store the various records tracing the design progression in a database system, I formalized the various elements presented previously (among others) in a UML class diagram, which I then used as a specification for the traceability prototype. This tool can be seen as an *a posteriori* workflow engine to declare the ongoing product development. I have tested this proof-of-concept within a turbocharger manufacturer in order to validate it. The case study showed interesting results in using DEPNET to support designers in conflict management and engineering change management (please see Ouertani and Gzara 2008 for further details). I should note that DEPNET could play a central role in executing an engineering program, where the coordination of information exchange between groups is still a challenge. However, further points remain to be considered on the issue of managing specifications' dependencies. First, an automated tool for charting and maintaining the coherence of DEPNET would be helpful. Second, mechanisms to check specification consistency need to be proposed to make it possible to reduce the number of specification dependencies.

References

Browning, T. R. 2001. Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE Transactions on Engineering Management* 48 (3): 292–306.

Browning, T., F. Fricke, and H. Negele. 2006. Key concepts in modeling product development processes. *Systems Engineering* 9(2): 104–128.

Bustnay, T., and J. Ben-Asher. How many systems are there? Using the N2 method for systems partitioning. *Systems Engineering* 8 (2): 109–118.

Ouertani, M. Z., and L. Gzara. 2008. Tracking product specification dependencies in collaborative design for conflict management. *Computer-Aided Design* (in press), available online at http://www.sciencedirect.com, doi:10.1016/j.cad.2007.07.002.

Terwiesch, C., and C. Loch. 1999. Managing the process of engineering change orders: The case of the climate control system in automobile development. *Journal of Product Innovation Management* 16(2): 160–172.

Wang, K. L., and Y. Jin. 1999. Modelling dependencies in engineering design. Paper presented at the ASME Design Theory and Methodology Conference. Las Vegas, Nevada.

INSIGHT

Systems Engineering Education at Nancy University

David Gouyon, david.gouyon@incose.org; Eric Levrat, eric.levrat@incose.org; Hervé Panetto, herve.panetto@incose.org; and Jean-François Pétin, jeanfrancois.petin@incose.org

Nancy University is a federation of three universities located in Nancy, France. Among a set of skill-oriented courses, this federation confers master's degrees in electronics, electro-mechanics, manufacturing, networks, and automatic control (Master of Science in Systems Engineering) and in software engineering (ESIAL Engineering School).

About three hundred multidisciplinary students are involved in the Master of Science in Systems Engineering, led by Professor Gérard Morel, who is strongly involved in the educational efforts of AFIS (French Association of System Engineering, the French chapter of INCOSE). This degree is affiliated with three research laboratories: CRAN, for the areas of automatic control, manufacturing, and networks areas (http://www.cran.uhp-nancy.fr); GREEN, for electro-mechanics (http://www.green.uhp-nancy.fr); and LIEN for electronics (http://www.lien.uhp-nancy.fr). For students, the main objective of this systems engineering education is to learn by doing, practicing a multidisciplinary and collaborative approach that allows them progressively to define, develop, deploy, verify, and validate a solution for a particular system. Indeed, the complexity of the technical objects to be implemented in industrial environments (which are themselves becoming more and more complex) requires us to enlarge the classical polytechnic training of students by making it a multidisciplinary one. The goal is to offer them a comprehensive approach for solving problems and linking the facts and knowledge needed to develop solutions from needs. Specific attention is paid to the modeling procedure so that students appreciate the interest of model-driven systems engineering and formal systems engineering. It is fully justified when digital business and engineering are used by engineers in remote way, and where systems embed more and more software to increase their evolution capacity while developing issues about their actual productivity and dependability.

The multidisciplinary training in systems engineering consists of the kernel and the relationships between all the educational components of the degree, in the form of courses and collaborative projects. The training aims to show the link between the processes of project management and best practices of specific jobs (for example, between systems engineering and *IEC 15288*). This is done in closed collaboration with enterprise management in order to balance the constraints of cost, quality, functions, and delays within a project. Student evaluation during these projects (developed during one or two semesters) is partly based on the results in terms of the polytechnic system implemented (system of interest) but also on the way students are following good practices to deploy the system (systems engineering; multidisciplinary engineering).

Located in the scientific campus of Nancy University, ESIAL (http://www.esial.uhp-nancy.fr) is one of the France's leading computer science and engineering schools, well known for its strong relationship to research — teachers at ESIAL are researchers at CRAN and LORIA, the Lorraine Laboratory of IT Research and its Applications, http://www.loria.fr — as well as for its outstanding teaching and the extensive educational and research opportunities it provides to its students. ESIAL is actively engaged in projects that not only impact the Nancy region but also relate to global challenges software intelligence.

ESIAL engineering school deals with systems engineering to provide computer science expert with a system vision to consider software embedded into its environment. This approach is justified when, on the one hand, considering information systems together with the human systems that use them and, on the other hand, when considering embedded software systems together with the hardware systems that implement them.

For systems information issues, it is a fact that enterprise resource planning systems alone do not cover all business needs. Thus, current practice involves many other enterprise information systems such as APS, SCM, CRM, and MES.[1] These software applications share a number of corporate data and can sometimes implement complementary or redundant processes. Ensuring the interoperability of these enterprise applications is therefore a real industrial challenge. One project-oriented ESIAL course deals with applying the best practices of system-of-systems engineering to study and implement data models that map between two information systems (interoperability between an ERP and a MES), based on the *IEC62264 (Business to Manufacturing)* standard.

For embedded software systems, the objective is to initiate students into model-based systems engineering. An ESIAL project-based course introduces requirements analysis, allocation of requirements onto software components architectures, and finally, designs and code generation. This course deals with informal methods and tools such as SysML for requirements and system modeling, but also more formal approaches (such as the "B" method or synchronous languages) to establish relationships between all heterogeneous models involved in a system engineering process and to help in issues of verification, validation, and qualification. The considered systems are related to real-time embedded control applications, which are constrained by high safety and dependability concerns, as standardized in the *IEC61508*.

1. APS (advanced planning and scheduling system),CRM (customer relationship management), MES (manufacturing enterprise system), SCM (supply chain management system)

Teaching Innovation with a Systems Engineering Perspective

Pascal Lhoste, pascal.lhoste@incose.org; and Frédérique Mayer, frederique.mayer@incose.org

ENSGSI-Specific Training

The ENSGSI (Ecole Nationale Supérieure en Génie des Systèmes Industriels / Engineering School in Industrial Engineering) engineering school offers engineer's and master's degrees along with research projects in innovative processes. A key feature of these degrees is that they develop an integrated approach of both innovation engineering and management. This type of program produces engineers with professional competencies to manage complex projects, including their social aspects. The school provides engineers with career prospects not only in industrial systems engineering but also in the fields of environment and design.

The key objective of the graduate program is to help students to seek the link between different engineering disciplines and the management of complex projects by adopting systems thinking. For that, systems engineering provides opportunities for students to understand this system perspective of the courses and the complexity of their future jobs. Thus, the pedagogy in use at ENSGSI allows links to be established across courses by combining programs (see the figure below) that center on the systems approach and programs that center on industrial engineering with industry-sponsored projects in order to concretely practice an innovative management of complex systems.

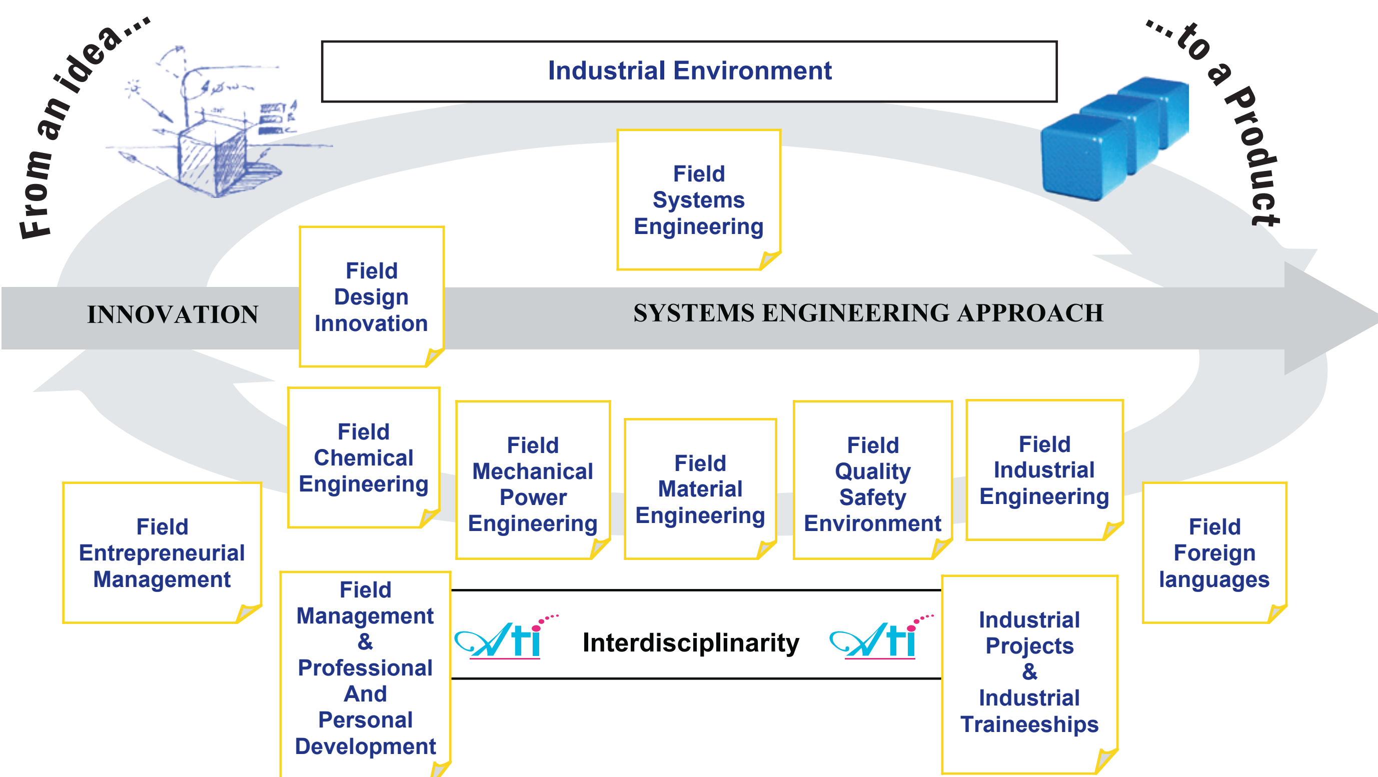
Experiential Level

These industry-sponsored projects are key elements of the pedagogy of "learning by doing" instituted at the school. These projects are developed in collaboration with industrialists for an experimental level of the courses and for an external work experience of the students. Each student does at least three projects during his or her time at the school. The objective of this experiential program is to help students gain practical knowledge that would improve their practice of systems engineering and innovation at ENSGSI.

In particular, a new type of project, the ATI (Working Group for Transfer of Competencies and Innovation), was launched to fill a need at ENSGSI for more synergy with local industry. Through a combination of both courses and work experience, an ATI project brings together different actors that include students of engineering schools, students of business schools, industrialists, researchers, and regional players.

The aim of an ATI project is to create the conditions of innovation that often fail to emerge in small and medium enterprises. The ATI concept enables the actors to go from the requirements expressed by owners to the execution of the project via the specifications to be respected by the people in charge of the project.

Systems approach and engineering span different steps of the ATI project life cycle in order to help the different actors deal with the complexity of an innovation process and to understand how this complexity should be addressed in both the project and the graduate program. Each ATI project is evaluated through milestones and deliverables. Competencies transfer and training as a project practitioner are thus developed throughout the project.



Team-based Systems Engineering of a Flexible Assembly Cell Control Architecture

Thierry Berger; Laurent Cauffriez; Sébastien Delprat; Dominique Deneux, dominique.deneux@univ-valenciennes.fr; Jean-Christophe Popieul; and Yves Sallez

The faculty at the ENSIAME Engineering School of Valenciennes, France, have proposed a new project-based course, "Informatics and Systems Engineering," intended for bachelor's degree students during their last academic semester. The objectives of this integration module, which is much appreciated by the students, are to revive their technical knowledge, to assemble the students within a significant project, to provide evidence of the value of rigorous methods, as well as to develop their sense of responsibility, and their ability to work together as a team.

The course project is to develop a distributed architecture for the flexible control of an automated production system. The physical system mainly consists of a conveyor, structured in a graph topology with twenty nodes and thirty-five arcs, permitting up to ten automated shuttles to be routed to seven different workstations, where automated processes implement services. The processes include four industrial robots, one manipulator, one inspection system, and one manual operator. The manpower for this exercise typically is a team of twelve to sixteen students, working fifteen weeks, six hours a week on this project.

Weeks 1 thru 5 are dedicated to analyzing the relative performance of candidate routing strategies, by means of simulation software. A robotic CAD system is used to model the elementary pick-and-place services that robots can implement. The durations of elementary assembly tasks become the inputs of a flow simulation model that reproduces the circulation of shuttles. This latter is used to evaluate the relative performance of several routing strategies.

Weeks 6 thru 15 are dedicated to implementing the strategy. The team is decomposed into small groups (one to three students) and the development tasks are distributed. The technical tasks include programming the robot tasks, developing automated sequences for the five programmable logic controllers in charge of a zone of the conveyor, programming a production pilot that makes real-time routing decisions (systematic application of the strategy), configuring an intelligent inspection camera so as to detect defaults, implementing a remote supervision system, and developing a flexible man-machine interface for the human operator in charge of managing defaults. Basically, the students must integrate pieces of known technologies to obtain a global behavior.

The complexity of this project is mainly caused by the requirement of productivity (implying real-time routing and dynamic load balancing), flexibility (every component can be picked from two different sources), and robustness to failures (every resource can be logically removed from the resource list, e.g., for maintenance purposes). Since a universal

> *continues on page 34*

A New Way of Learning Component-based Approaches for Integrating Expertise, Reliability, and Maintenance into Design of Automated Systems

Laurent Deshayes, laurent.deshayes@ifma.fr; and Khalid Kouiss, khalid.kouiss@ifma.fr

Production and maintenance departments cannot wait any longer for equipment to fail before planning their operations. In today's businesses, the maintenance function must allow for the prediction of anomalies in the production system and prevent any downtime. Classical approaches are based on management methods like TPM (total productive maintenance), reliability-based maintenance, or failure mode and effects analysis, which requires a high level of expertise. These methods generally

- are badly integrated with the execution level through the MES (manufacturing execution system),
- are based on experience without using physical models to evaluate the equipment's "health" and reliability, and
- require a great deal of time to improve new equipment. Moreover, today's application in maintenance involves designing and redesigning equipment to improve its reliability and productivity. As a consequence, it is now necessary to integrate the following factors on industrial equipments:
- reliability based on physical models to estimate remaining life time,
- expertise on diagnosis and maintenance operations, and
- functions, indicators, and Human Machine Interfaces toward the MES.

The proposed approach to achieve this goal is *component-oriented* for obtaining a great modularity of the designed system toward maintenance applications. It is built using the model-driven engineering method and is constituted of two simultaneous and iterative analyses, such as shown in figure 1:

> *continues on page 36*

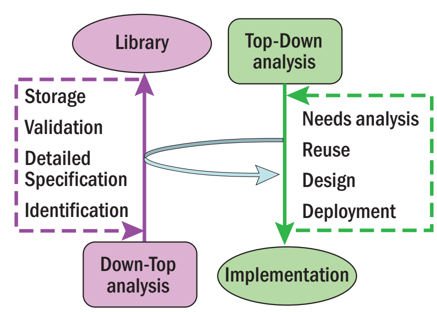


Figure 1. Proposed approach for a component based design of expert reliability based systems

Hands-on Experiences in Project-based Systems Engineering Education at INSA Toulouse

Jean-Charles Maré, jeancharles.mare@incose.org; and Marc Budinger

The Institut National des Sciences Appliquées (INSA) in Toulouse, France, has recently begun offering some interesting project experiences to its students in complex technological systems engineering. The first of these projects aims at introducing systems engineering through an optional course of forty contact hours to the second-year students (in France, engineering studies last five years after high school). This course focuses on the design process, from the functional analysis level to the multi-domain, one-dimensional simulation. We selected automotive steering systems and aircraft flight control actuation systems, since they efficiently illustrate the variety of industrial solutions (e.g., electric or hydraulic supply, electrical or mechanical signaling). Throughout this project, each student group must analyze the architecture, verify the power sizing, discover the closed loop control principles, and produce a simplified virtual prototype of the system under study. At the end, each group presents its results to the whole class, which learns about each specific industrial solution.

The second project, which lasts thirty-five hours, is offered on the first week of the fourth year, for students preparing the systems engineering degree. The simple project introduces students to the need for a systems engineering approach. In the first two days of the week, the student groups must compete to submit an architecture proposal in response to an industrial need (typically, a mobile robot). During the second part of the week, each group must collaborate as a partner to produce the robot subsystems according to the selected architecture using LEGO® MINDSTORMS© (http://mindstorms.lego.com), and to deliver an operational robot on Friday morning.[1] The last afternoon is dedicated to a debriefing where students, tutors, observers, and professors summarize the lessons learned. The need for knowledge is clearly identified and linked with the fourth- and fifth-year programs. On the basis of the four-year experience with this curriculum, we noticed that all of a student's major difficulties are pointed out during this week, not only those related to process or scientific skills but also including issues with interpersonal relationships, organization, communication, and teamwork. Also, we appreciate that students in their final year of study contribute actively as tutors for this project.

In the third project, the fourth-year students are invited to apply a verification and validation process to a simplified scaled mockup of an aircraft air-conditioning system. Based on the requirements, the student groups have to propose and validate the digital controller. In this attempt, they must develop a virtual prototype of the air conditioning devices that in practice have uncertain operating parameters. Consequently, they must specify and perform partial experiments to feed the model with accurate parameters. The validated virtual prototype of the hardware is used to design the real-time controller that is later implemented within the Dspace© or XPcTarget© environment. As a final test, the proposed controller (hardware and software) is connected to the real-scale model of the air-conditioning system to demonstrate the quality of the performed work.

The example of a pressure regulator for direct high-pressure gasoline injection is the frame of the multidomain and multiscale simulation project. This simple component involves hydraulics (variable orifice function), mechanics (dynamics of the valve) and electromagnetic (pulse-width-controlled solenoid) that require the students to combine different dedicated software and modeling methodologies. This project aims at providing to the systems engineering students the competence to interface with domain specialists and to establish bridges between domains and between global and local modeling.

This sample of projects proposed to systems engineering students illustrates the importance paid by the education team to know-how and practice to complement lectures in developing the scientific, technological, and human skills of future systems engineers.

1. See B. Doucet and J.-C. Maré, "Introduction active et collaborative à une formation à l'Ingénierie des Systèmes: Des manques aux besoins, des besoins au cursus," paper presented at the conference, "Questions de pédagogie dans l'enseignement supérieur," Louvain, France, 2007.

Berger *continued from page 33*

solution to this complex problem does not exist, compromises are necessary. The students are already acquainted ahead of time with most of the technologies involved in this project. The major difficulty for them is more organizational than technical: they can achieve a functional and efficient result only if they can be coordinated. Actually, they are free to develop, and also to specify, the components and individual behaviours of the control command architecture. Common decisions are necessary concerning the nature of information to be exchanged among equipment, the format for representing this information, the logical and physical means to support them, and the protocols to be implemented. Only a planned and methodical progression, controlled by a quality approach, organized by the project manager (one of them dedicated to this role), can produce a performing outcome in due time. Two teachers supervise the project to ensure security, clarify the requirements from a customer's viewpoint, facilitate the resolution of technical problems, and play the role of consultant. At the end of week 15, a global demonstration of a production batch, introduced by a conference by the project manager and followed by technical poster sessions by each group, defines the formal evidence of the group's efficiency.

A Student Challenge in Systems Engineering: RobAfis '07

David Gouyon, david.gouyon@incose.org; and Jean-Claude Tucoulou, jeanclaude.tucoulou@incose.org.

RobAfis is a challenge open to student teams. The objective is to get to implement a systems engineering approach, while working as a team to design and assemble a robot.

The general phasing of the challenge, covering about ten months, was as follows:

- Teams' registration
- Registration confirmation and sending of the request for proposal
- Receipt of proposals
- Evaluation of proposals and results publication
- Sending of a reference document for the development phase and of robot kits
- Receipt of development documents
- Evaluation of development documents
- Presentation of robots to a jury during AFIS academic forum 2007, in workshop session
- Operational demonstration during AFIS academic forum 2007, in plenary session

Challenge Management and Allocation of Responsibilities

The acquisition agency was composed of David Gouyon and Jean-Claude Tucoulou, while a team of six students of the specialized master's degree SYVAT of Arts et Métiers Paris Tech (www.ensam.fr) played the role, in a pedagogical project, of prime contractor. The missions of the acquisition agency were to define the main objectives and principles of the project, to do a preliminary study of feasibility, to define the general calendar, to manage the whole operation, to validate the documents produced by the prime contractor, and to evaluate the proposals and development documents. The missions of the prime contractor were to build the rules of the challenge, the request for proposal, the development documents, and to participate during the evaluation phases.

Final Phase Schedule

The first job of the teams was to present their system and the systems engineering processes they used to a jury composed of industrials and academics. The second one was to validate their robot's performance in operational conditions in front of all the participants of the academic forum. The last step consisted of a quick presentation of the system to the whole assembly, before the distribution of prizes.

Robot and Challenge Specifications

The robot had to be able to collect bricks situated next to a black line and bring them back to a specific zone materialized by the light source, move along a specified environment (land characteristics, dimensional constraints, etc.), be smaller than a maximum size, be made of an imposed set of components, be autonomous in energy and guidance, and be maintainable.

To participate in the challenge, the teams had to complete the various phases of selection, proposal and development, build a robot, and participate in the final phase to compete against the other teams.

Teams

Four teams have participated in the final phase of the challenge: École des Mines d'Alès (EMA), École Supérieure d'Informatique et Applications de Lorraine (ESIAL), Institut National des Sciences Appliquées de Toulouse (INSA Toulouse), and Master Ingénierie Système en Electronique, Electrotechnique, Automatique, Productique et Réseaux (Nancy-Université).

Benefits for Participants

All of the participantsreceived benefits from participating to the challenge. For the prime contractor and the challenging teams, these included:

- collective work in project teams during the phases of proposal and development;
- use of collaborative team work places, such as video-conferencing, audio-conferencing, document sharing, instant messaging; and
- project phasing and control.

For the prime contractor, the benefits included:

- collaborative work during the elaboration of the request for proposal and the development documents,
- participation in the evaluation of the documents, and
- an experience of assistance to the acquisition agency.

There were also many take-aways for the challenging teams:

- realizing the impact of the quality of documents on the end product,
- achieving a global view of systems engineering processes, activity and deliverables during proposal and development phases, furniture of intellectual service
- providing of intellectual service
- sensitization to the fundamental role of project management to coordinate systems engineering processes and activities, integration of an operational enabling system, maintainability, risk, cost, configuration and traceability management, structuring of documents, interest of validation and verification of the product
- sensitization to the interest of theoretical justification and manufacturer validation, in order to master performances in operational exploitation,
- sensitization to the importance of maintainability, which has been accounted for at the same time by the integration of corresponding

> *continues on next page*

Gouyon *continued*

studies in the whole set of engineering processes, and by an operational demonstration in front of the jury.

Evaluation of Proposals and Development Documents

Proposals have been evaluated among various criteria which are listed in table 1.

Development documents have been evaluated among various criteria which are listed in table 2.

Table 1

Criteria	Rank
Requirements referential	1
End product presentation	1.5
Justification of architectural choices	1.5
Development validation plan	1
Maintainability documents	1
Management macro plan	1
Systems engineering methodologies implementation	1.5
Systems engineering tools implementation	0.5
Innovative aspects of the product	1

Table 2

Criteria	score
Requirements referential	/20
End product presentation	/40
Definition documents	/30
Design solution verification documentation	/30
Integration, Verification, Validation Plan	/20
Maintainability study and maintenance definition	/20
Management plan	/30
Editorial quality of the document	/10

Results of the Final Phase of the Challenge

The results of the challenge are the following:

1st: INSA Toulouse
2nd: Master IS-EEAPR
3rd: EMA
4th: ESIAL

Analysis of the Challenge

The uniqueness of this challenge is that beyond the implementation of systems engineering processes which are necessary for the good development of a project, student teams have learned and understood the various contracting phases of an industrial project (proposals, evaluations, development). This is something quite impossible in the context of a traditional academic project. Teams had to follow imposed batches at macro and micro levels in order to allow a clear structuring of their work and to facilitate the evaluation by the acquisition agency.

They noticed the importance of the deliverables of the various phases of an industrial project, and understood how a precise definition of the end system is very crucial (though by itself not completely sufficient, either) to be able to reproduce and industrialize the system.

Deshayes *continued*

- A *down-top analysis* for the design of individual and elementary standardized equipments, called "components." This analysis follows four steps: identification, detailed specification, validation, and storage into libraries. Maintenance department experience plays an important role in helping the engineer to evaluate faults and determine what needs to be improved.
- A *top-down analysis* is applied to all new projects, consisting of integrating and connecting standardized components to implement a system that satisfies design specifications. This analysis also constitutes four steps: needs analysis, reuse of existing components, design to integrate them, and deployment toward targeted application.

Using MDE and the specification language SysML, it is now possible to specify the command application of an automated production system by delaying technological choices for the final application. Thanks to SysML, it is easy to establish direct relations between components defined during the design stage and the Function Block proposed by the *IEC 61131* standard. Components identification allows an engineer to capitalize a specific function for reuse in new projects. Three kinds of modules are generally distinguished: modules describing physical parts of the equipments, modules for expert reliability-based maintenance, and collaborative functions between these modules. Because these modules can be different physical aspects of the components, they are called multi-facet components. Therefore it is possible to specialize a component for the application needs without redesigning the entire system.

Expert reliability-based maintenance is implemented by developing specific modules for maintenance application. These are then specialized to satisfy maintenance specifications. Teaching such an approach involves identifying the different necessary methodologies, software, and equipments, in order to integrate this set into an adapted formation by developing different teaching and application modules, and adapting them to existing teaching programs.

Mapping Design Innovation Processes: A Contribution to Reflexive Learning

Denis Choulier, choulier@utbm.fr

Innovation processes aim at proposing new products or services to users, and are developed inside a competitive environment. Executing these processes requires us to view products from multiple aspects; forecasting projects commonly involve both engineering designers, strategic planners (from fields like patent rights, or economic intelligence) and users' spokesmen (such as product designers, or marketing specialists). These processes can be supported by tools to gather, structure, and present information; most of them are methodological tools and help innovators to reflect on the product definition.

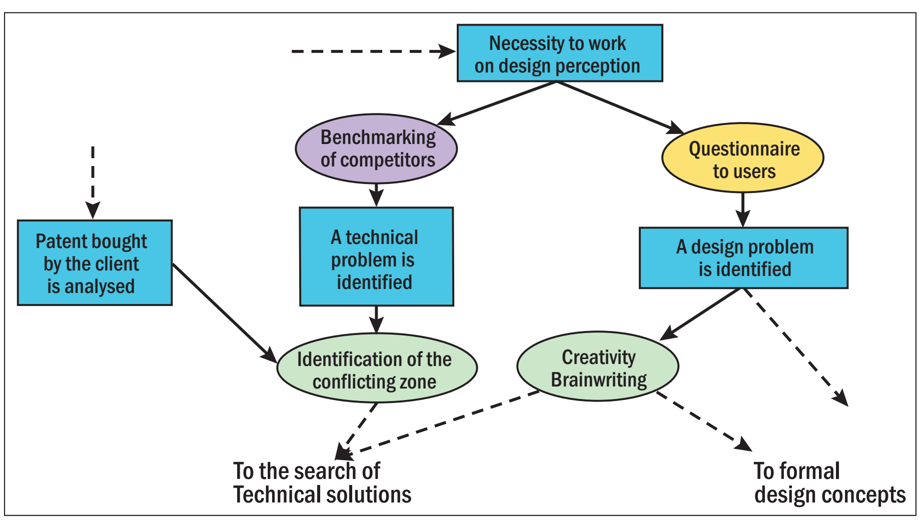
In order to teach innovation processes, it is necessary to help students learn how to construct different points of view on products by the use of tools, and to apply the tools in representative situations. Nevertheless, when projects are limited to the application of taught concepts and procedures, knowledge acquisition is at best individual and implicit. This is especially the case for procedural knowledge such as the choice and adaptation of tools, project piloting, or learning to work with someone really different. In order to favour the emergence, formulation, and sharing of these knowledge, specific means are necessary. The concept of reflexive practice is used in that students are asked to take a distant view of the processes they follow, to analyse them, and to give an account of them. In this way, learning becomes reflexive and collective.

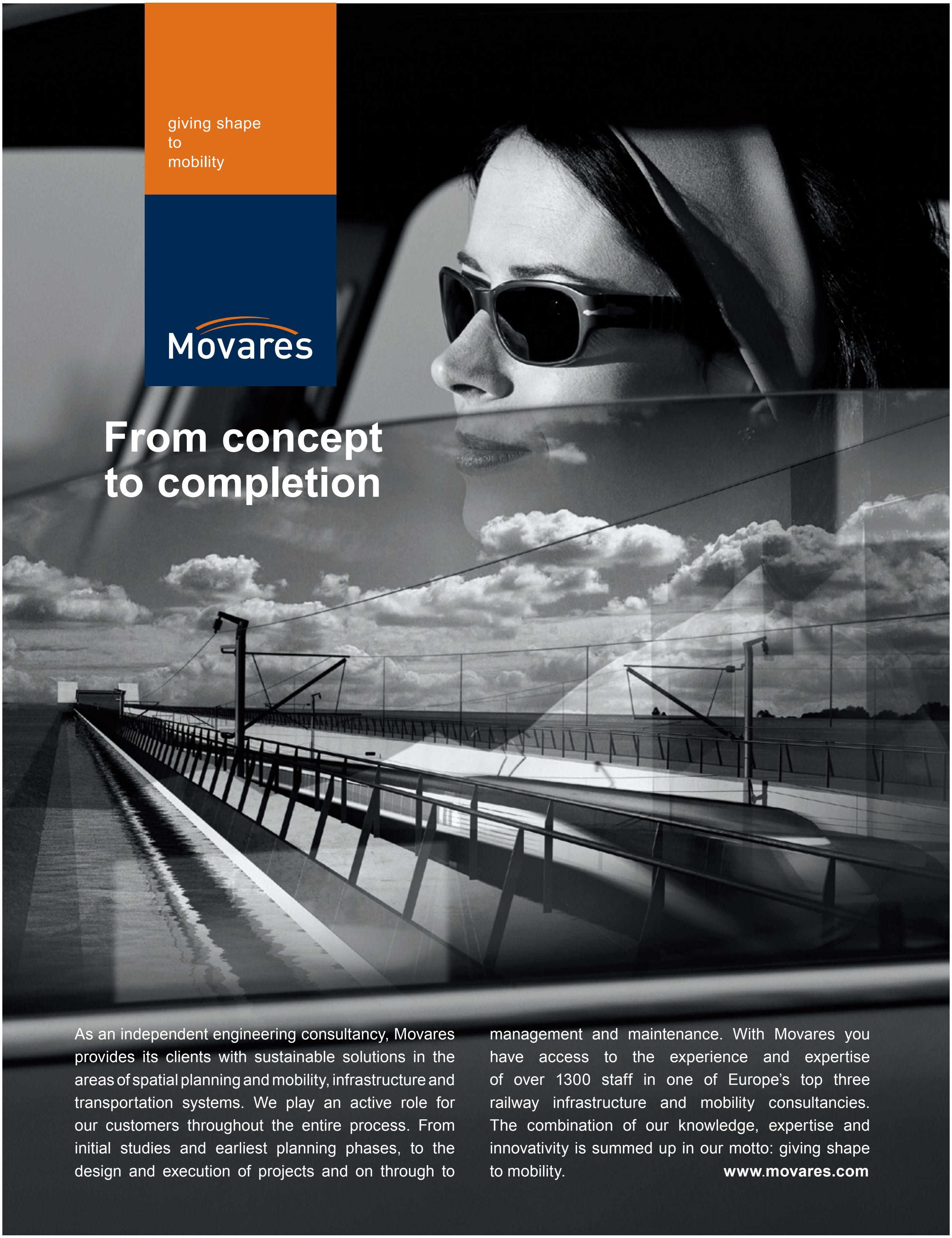
The faculty of the Université de Technologie de Belfort Montbéliard in France conducts an experiment on teaching innovation in one master's degree program, according to these principles. We make sure the program is multidisciplinary both by admitting students with diverse initial competencies and by separating them into three groups at the beginning of the final year, where each group investigates one set of tools. The first set of tools covers strategy development methods such as PEST, Porter, strategic patent analysis, and competitors' investigation. The second set is more technical and covers product analysis and creativity techniques such as functional analysis, Teoriya Resheniya Izobretatelskikh Zadatch (TRIZ), and brainstorming. The third covers product design and marketing: it introduces the voice of the user into the innovation process using tools such as questionnaires, product testing, scenarios. Typically, an industrial "client" assigns an innovation project to three students—one from each group. Teachers play the role of mentors (for the use of tools) and coaches (for project piloting and observation). During projects, student reflection is assisted by discussions, teachers' observations, diaries, presentations, and debriefings.

One specific tool for reflection is presented here. During the projects, we ask the student to draw a graph of the process linking the tools with what they produce. Each tool gives "results" such as information, effects on the group, decisions, or unexpected discoveries. Based on these results, the innovation project team chooses to investigate another field using other tools. Our concept of "tool" is large; it encompasses any means consciously used to obtain some result. Aside from the tools from one of the three groups mentioned above, generic tools can also be considered (such as a clients' meeting, a visit to a trade show, CAD modeling, building a mock-up, or mind mapping).

An extract of such a graph is given in the figure. Because of the necessity to improve the perception of an industrial client's product, students tried two parallel actions. First, they investigated the products of competitors, by consulting catalogues and the Internet, and visiting supermarkets (this fell under the first group of tools). Incidentally, this research revealed that the technical problem covered by the client's patent was partly treated with other means by a competitor. They decided to engage in the improvement of the solution by using tools of the second group. Second, the students drew up a questionnaire for potential users (this fell under the third group of tools). This questionnaire confirmed the need for working on the design of the product (including elements like shape and colors), and collected users' preferences for various product features. Then, a session of "brain-writing" (a modification of classical brainstorming) produced ideas for both problems.

The figure shows an interaction between three different "profession," but in other situations, more sequential uses of tools from the same group appear. These graphs reveal "logics" of action inside one profession as well as cross-fertilization. It helps students to become aware of the existence of these two logics.




giving shape
to
mobility
Movares
From concept to completion
As an independent engineering consultancy, Movares provides its clients with sustainable solutions in the areas of spatial planning and mobility, infrastructure and transportation systems. We play an active role for our customers throughout the entire process. From initial studies and earliest planning phases, to the design and execution of projects and on through to management and maintenance. With Movares you have access to the experience and expertise of over 1300 staff in one of Europe's top three railway infrastructure and mobility consultancies. The combination of our knowledge, expertise and innovativity is summed up in our motto: giving shape to mobility.
www.movares.com

Fellows' Insight

Decomposed Requirements versus Derived Requirements

A. Terry Bahill, terry.bahill@incose.org

Some people treat *requirement decomposition* and *requirement derivation* as synonyms. They are not!

The Systems Engineering Handbook (ver. 2a, p. 277) provides these definitions:

> "Decomposition. The process of decomposing higher-level requirements into more-detailed constituent functions and associated performance levels and allocating those requirements to specific hardware, software, and support elements."

> "Derived Requirements. Those characteristics typically identified during synthesis of preliminary product or process solutions and during related trade studies and verifications. They generally do not have a parent function and/or performance requirement but are necessary to have generated system elements accomplish their intended function."

Martin Eigner stated it this way:[1]

> "Requirement Decomposition. The activity of breaking down a text requirement into two or more text requirements whose total content is equal to the content of the original one—just expressed more explicitly or in more detail. The decomposed requirement is replaced by the resulting requirements and therefore becomes obsolete."

> "Requirement Derivation. The activity of creating new requirements based on one or more existing ones. Unlike for requirement decomposition, the statements of the newly created requirements are different from those of the existing ones. Consequently, the input requirements do not become obsolete."

The CMMI[2] does not mention requirement decomposition, but it does make this claim about derived requirements: "Derived requirements arise from constraints, consideration of issues implied but not explicitly stated in the customer requirements baseline, and factors introduced by the selected architecture, the design, and the developer's unique business considerations… Derived requirements can also arise during analysis and design of components of the product or system."

Bahill's consensus is as follows:

- Requirements decomposition breaks down a requirement into two or more requirements whose total content is equivalent to the content of the original one, but expressed more explicitly or in more detail. The decomposed requirement is replaced by the resulting requirements and therefore becomes obsolete.
- Derived requirements arise from constraints, consideration of issues implied but not explicitly stated in the customer requirements, and factors introduced by the selected architecture, the design, and the developer's unique business considerations. Unlike decomposed requirements, the statements of the derived requirements are different from those of the original requirements.
- Consequently, the original requirements do not become obsolete. In SysML, the decomposed requirement would be retained and used to show the requirements tree hierarchy.

A concrete example may make the definition of requirements decomposition clearer. An engineer has been given the task of designing a camera system to detect when an office coffee pot is empty. She thus starts with this "customer" requirement: "When an empty coffee pot is placed in the coffee machine, the camera system shall transmit a digital image to the server." The engineer can then decompose this requirement into three functional requirements:

1. The system shall sense when an empty coffee pot is being placed in the coffee machine.
2. The system shall trigger the camera when an empty coffee pot is placed in the coffee machine.
3. The system shall transmit the digital camera images to the server.

(These requirements come with the stipulation that "empty coffee pot" means one containing less than six ounces of fluid.) Now that the engineer has broken the requirement down, she can dispose of the original requirement, because she will allocate each of the three functional requirements to a particular object; the original customer requirement, however, will probably not be allocated to an object.

By contrast, here is an example of a derived requirement that would pertain to the same situation of the camera system. The original requirements would be these:

1. The system shall allow the last 10 hours of images to be viewed by a COTS image viewer.
2. The system shall store up to 2.5 images per hour at a maximum of 500 kilobytes per image.
3. Images more than 10 hours old shall be archived.

From these requirements, the engineer could derive a local storage requirement, that the system shall have a local storage capability of at least 12.5 megabytes. In this case, the original requirements remain, even after the local storage requirement has been determined.

1. http://www.requirement-management.us/requirement_management/Whitepaper_RMT_Web.pdf

2. Software Engineering Institute, *CMMI for Development, Version 1.2* (Pittsburgh, PA: Carnegie Mellon University, 2006), 394.

Forum

The INCOSE *Systems Engineering Handbook* and the *Project Management Body of Knowledge*: A Preliminary Comparison of the Lifecycle Stages with the Process Groups

Kent Gladstone, kent.gladstone@incose.org

This article lightly explores the marriage between project management and systems engineering. One discipline cannot deliver a successful result without the other. Systems engineers need someone to handle the customer and management, and project managers need the systems engineers to measure the progress, design the solution, and deliver the product.

The idea for this article came about while I was studying for certification as a project management professional at the Project Management Institute (PMI), which uses the *Guide to the Project Management Body of Knowledge (PMBOK® Guide).* My work assignment at the time involved creating a Web version of a systems engineering reference guide using several resources including the INCOSE *Systems Engineering Handbook,* version 3 (Haskins 2006). The juxtaposition of my studies and my work assignment has been enlightening, and a bit confusing. The languages of each have common elements, but their definitions may vary. Both INCOSE and PMI are trying to formalize their respective disciplines. Adding to this medley of standards and processes are the Capability Maturity Model-Integrated® (CMMI®) methodologies not included in this article. How are a program manager and a chief engineer to understand what a project needs without a clear understanding of these two (or three) disciplines?

In its *PMBOK® Guide*, PMI defines project management as

> "… the application of knowledge, skills, tools, and techniques to project activities to meet project requirements. Project management is accomplished through the application and integration of the project management processes of initiating, planning, executing, monitoring and controlling, and closing. The project manager is the person responsible for accomplishing the project objectives" (ANSI 2004, 8).

INCOSE's *Systems Engineering Handbook*, version 3, provides several definitions of systems engineering. I will quote that of Dr. Howard Eisner out of my respect for my former professor:

> "Systems engineering is an iterative process of top-down synthesis, development, and operation of a real-world system that satisfies, in a near optimal manner, the full range of requirements for the system" (quoted in Haskins 2006, 2.1).

PMI posits that the project manager is ultimately responsible for the health and progress of the project and the successful delivery of the product. A chief engineer is ultimately responsible for the design of the product, resulting in the successful delivery and implementation of the product or system. But can a project manager manage a project to the level of detail that PMI encourages? How much of the chief engineer's time is spent meeting the informational needs of the project manager? And at what points in the project do these separate disciplines overlap?

Lifecycle Stages versus Process Groups

The *PMBOK® Guide* looks at project management from two perspectives: knowledge areas and process groups, where the former cuts across the later to form a matrix. The process groups incorporate five parts of a project lifecycle (or phases of the project) that are repeatable. The knowledge areas include a framework, followed by the disciplines needed to run a successful project. Not all knowledge areas fall under all the process groups. The *PMBOK® Guide* emphasizes that the difference between a product lifecycle and the project lifecycle is that the former has a longer life than the latter. The *PMBOK® Guide* states that a project is unique and temporary; the project ends when the product is delivered and is in operation.

The *Systems Engineering Handbook* also has two main focus areas: lifecycle stages and processes. Although the handbook does not directly mention project management, many project management activities are described under the project processes of planning, assessment, and control, as well as under the enterprise processes of investment, management, and resource management. As in the *PMBOK® Guide,* a project closes when the customer has accepted the product, yet systems engineering processes span the operation of the product until it is retired. Figure 1 shows the handbook's diagram of the overlap of the two disci-

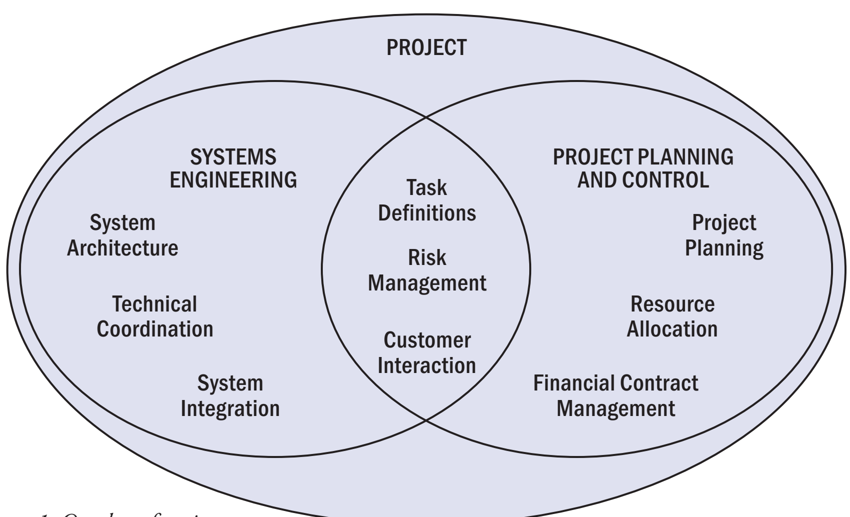


Figure 1. Overlap of project management and systems engineering disciplines

plines (Haskins 2006, 5.1).

Task Definitions, Risk Management, and Customer Interaction are depicted as the area of overlap where systems engineering provides inputs to project management and control. These metrics include completion of activities, the number of change requests, quality control, and requirements validation and verification.

The list given in table 1 is an attempt to compare the *PMBOK® Guide's®* process groups to the *Systems Engineering Handbook's* lifecycle stages. The first thing one observes is the expected surplus of systems engineering stages beyond the life of the project itself. Next, one can observe that the stages do not cleanly map on a one-for-one basis.

Table 1. PMBOK® Guide *process groups versus INCOSE lifecycle stages*

PMBOK® Guide : Process Groups	SEHv3: Life Cycle Stages
	Pre-concept Stage or R&D
Initiating	Concept Stage
Planning	Concept Stage
Executing	Development Stage, Production Stage
Monitor and Control	Development Stage, Production Stage
Closing	
	Utilization Stage
	Support Stage
	Retirement Stage

Initiating and Planning versus Concept Stage

According to PMI, the Initiating Process "defines and authorizes the project or a project phase." Initiation is really about developing a business case for the project. During this stage the project is assessed for whether it meets the business goals (for example, to introduce a new evolutionary idea to remarket a product to increase sales), whether it is required by law (for example, if a company needs to implement processes to meet the Clinger-Cohen act, this could be considered a project), or whether a customer needs it. This is the stage when a project manager decides to divide the project into phases if the project is too large, when a customer provides budgetary and schedule constraints and a preliminary scope, and lastly but most importantly, a project charter is established that authorizes the project manager to manage the project.

The *PMBOK® Guide's* Initiating process group is different from INCOSE's Pre-concept or R&D stage. The R&D stage could be interpreted as providing the proof of the utility of a capability required by a project before authorization to proceed with Initiating. I listed it ahead of Initiating purposely to demonstrate that without proof, the project may not go forward. Of course, there are instances when this stage is bypassed.

The Planning group includes several activities: starting the project management plan, completing the scope from which the project manager creates a work breakdown structure, deciding on a schedule and budget, identifying resources, and identifying risks. During this stage, the criteria metrics for success are listed, and plans for quality assurance and control are generated. The *PMBOK® Guide's* language remains rather high-level at this stage; I assume that this is where systems engineering comes into play.

It seems that INCOSE's Concept stage supports the initiating and planning stages of project management. Systems engineering activities are requirements-driven as opposed to product-driven. The requirements create the foundation from which the work breakdown structure, project schedule, and project cost are derived. It is during this stage that prototyping and modeling will give both the project manager and the chief engineer an idea of the best design solution for the product in question.

Executing, Monitoring, and Controlling versus Development and Production Stages

The *PMBOK® Guide* Executing process group covers the work of carrying out the project management plan by integrating resources and people. It is during this cycle that the project manager may implement requests for change, corrective actions, or defect repair. Under the Monitoring and Controlling Process Group, one finds advice about what needs to be done to monitor and control project performance against the plan.

The objective of the systems engineering Development stage is to do the work of developing a product that meets the requirements of the customer. This stage includes development, integration, verification, and validation activities.

I would argue that the Production stage of systems engineering also falls under Executing. Where engineers call it incremental development, project managers call the same progressive elaboration "developing in steps." Once there is enough information, production can begin. However, the project manager still needs to track costs, change control, and team integration. I would think that it is the chief engineer's role as advisor to the project manager to track completion rate of work packages, needed changes, and the best resources for a particular job.

Systems engineering is concerned throughout the processes with configuration management, quality assurance, testing, verification, and validation. Metrics collected during these activities are shared with project management.

Closing versus Utilization, Support, and Retirement Stages

The *PMBOK® Guide* has two types of closing: Administrative and Contract Closure. Although the project ends, the systems engineering life-cycle continues. When the product is completed, the project manager verifies that the defined processes are completed within all the process groups, has the deliverable validated, creates and stores lessons learned, and closes the contract through customer acceptance of the deliverable. The project manager's final act is to hand the product over (transition) to operations.

In systems engineering the life of the product or system continues. There are processes for utilization and support until the product or the system is outdated and retired. A help desk, vendor upgrades, and daily backup procedures might support a software product until a new product replaces it. During the retirement stage, it is necessary to handle the conversion of data, training of users for the new system, and disposal of the old one.

References

ANSI (American National Standards Institute). 2004. *A Guide to the Project Management Body of Knowledge.* 3rd ed., version 1.3.

Haskins, C., ed. 2006. *Systems engineering handbook: A guide for system life cycle processes and activities.* Version 3. Seattle: INCOSE.

A Universal Architecture Description Framework, Requirements Analysis, and Specification Preparation

Jeffrey O. Grady, jeffrey.grady@incose.org

It is time for systems engineers to apply integration and optimization to the extensive and creative growth of modeling methods that has taken place over the past fifty years. We should applaud those who have developed these methods, while recognizing that the time is long past due when we should take a global view of the process of understanding problem space and translating it into solution space. Figure 1 is an attempt to offer this global perspective across a system's lifecycle, with emphasis on the system definition component and the content and formatting of the principal product of this work, the specifications. There are many models from which to choose, but at present, despite the heroic work of many individuals over the last fifty years, there are no existing models that have been developed that are so comprehensive that every requirement appearing in every specification on a development program can be derived using just that one model. At the same time it is possible to devise such a modeling approach using some combinations of artifacts that existing models employ.

The advantages of following this line of reasoning to a conclusion are that we can (1) foster improved hardware–software integration, (2) create business opportunities for the tool makers to expand their tools to universally support the development process and (to the extent they choose not to do so) to work cooperatively to interface subset tools more effectively, (3) provide a template within which an effective training program can be framed, (4) provide a comprehensive modeling framework that systems engineers, even hardware-dominated ones, can master and employ in their important work, (5) close the effort well-begun in *CMAN 80008A*[1] and *MIL-STD-498*[2] and carried forward by *MIL-STD-961E*[3] to evolve a single specification format for all specifications, and (6) clearly connect the modeling work with the content of specifications such that all requirements may be derived through modeling and the relationships published.

Table 1. Comparison of applicable models for architecture descriptions for systems, hardware, and software

Initials	Title	Sys	Hw	Sw
TSA	Traditional Structured Analysis	Y	Y	N
MSA	Modern Structured Analysis	N	N	Y
PSARE	Process for System Architecture and Requirements Analysis	Y	Y	Y
UML	Unified Modeling Language	N	N	Y
SysML	Systems Modeling Language	Y	Y	N
DoDAF	(U.S.) Department of Defense Architecture Framework	Y	N	Y

I have chosen six models as the current modeling set: they are compared in table 1. It is probably clear that it should be possible to formulate comprehensive architecture description models from two subsets of these models.

The word *architecture* has many masters so it was necessary to be selective in picking a meaning for use in this paper. For this paper, I am using the definition contributed by Mr. Brian Wells, chief system engineer at Raytheon, who offered that it is an inherent property of a system created by the parts, their interconnections, and their arrangement. I have used the DoDAF[4] meaning for the phrase *architecture description*: "An architecture description is a representation of a defined domain, as of a current or future point in time, in terms of its constituent parts, what those parts do, how they relate to each other and the environment, and the rules and constraints governing them." We model these architecture descriptions using the techniques suggested in figure 2. The human seeking to understand a problem space creates models of the space using simple graphical renditions from one of several perspectives because one view is insufficient to express the complexity of the problem space.

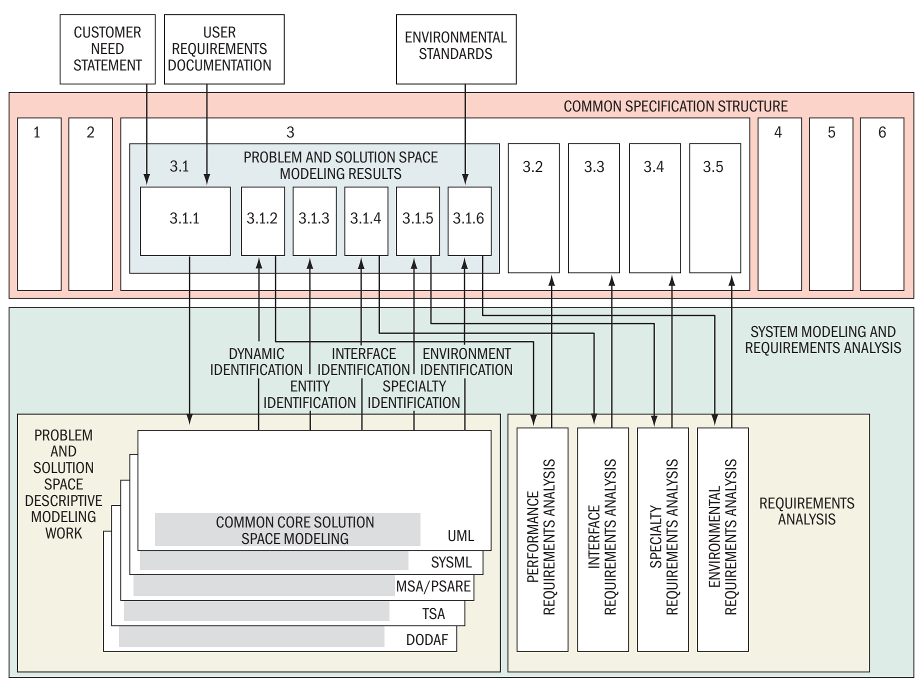


Figure 1. Summary view of system development

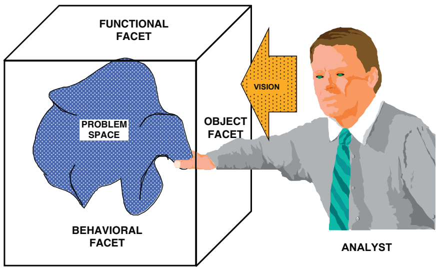


Figure 2. The relationship between the systems engineer and models

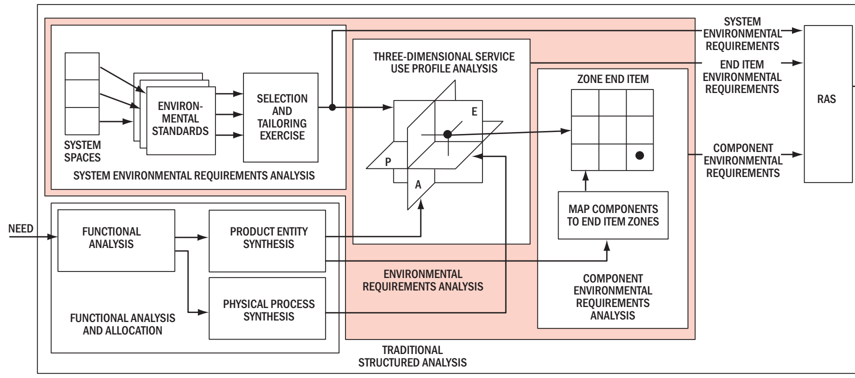


Figure 3. Three-layered environmental analysis model (from Grady, System Requirements Analysis[7]*)*

We must simultaneously understand what the system must do to successfully solve the problem space objectives (expressed in the need statement), what it must consist of, and how it must behave in so doing. The hand-eye coordination applied (pencil and paper or computer keyboard, mouse, and monitor) in building these images is helpful in cementing the meaning into the mind of the creator, and the simple graphical expressions provide the most effective way to transfer ideas from one person's mind into the mind of another using the most powerful means available—vision. Simple graphics encourage easy transfer while not conveying a complete story. More complex graphics make the passage more difficult while imparting a richer story. Generally, a single set of graphics is insufficient, and the set most often selected in modeling approaches includes the object (also known as the static or physical) facet (what the system must consist of); the functional facet (what the system must do); and the behavioral facet (how it must behave). The functional facet would then be used as the leading type of analysis, following the idea "form ever follows function" expressed by Louis Sullivan in his "letters to the kindergarten"[5]—actually addressed to young architects in Chicago—in the last century. In accomplishing this work the analyst stares at the problem space until it starts to differentiate into specific views that precipitate on the three facets. In the end, the analyst's uncertainty about the problem space has dissolved, and the union of the diagrams describes the architecture of the system.

It was a pleasant surprise to discover that PSARE[6] (known to many as HP or HHP) could satisfy the comprehensive requirement, as suggested in table 1. I finally woke up to PSARE's applicability not only to information flow (the limit of the MSA from which it was derived) but also to material and energy after years of encouragement by John "Mike" Green and because of a hardware–software integration class taught by Edward Fields at the University of California, San Diego. PSARE may need some of the same adjustments that the comprehensive model formed from UML and SysML requires.

One would hope that the work to create and evolve UML and SysML would result in a universal model, but to date, the Object Management Group has not yet arrived at that capability. This paper suggests that we add four artifacts from TSA to the current combination creating a comprehensive model. Those four artifacts are as follows

1. A requirements analysis sheet, ideally in the form of a computer database, like DOORS for example. This sheet would capture the relationship between the models and specification content.
2. A common product entity structure.
3. A specialty engineering scoping matrix.
4. A three-layer environmental model.

The adherents of UML will claim, of course, that their objects, classes, components, and nodes collectively satisfy the need for the product entity structure, but these static structures are not respected in SysML apparently because of a preference among systems engineers for blocks rather than classes. SysML has a construct called a requirement but it is hard to imagine that one would actually employ this construct on a system with one hundred fifty specifications, each containing one hundred to two hundred requirements. Interestingly, UML does not recognize such a construct. Neither UML nor SysML have an organized method for identifying and characterizing specialty engineering or quality requirements. My book *System Requirements Analysis*[7] pulled the specialty engineering scoping matrix out of our modeling history from *AFSCM 375-5*[8] (referred to then as a "design constraints scoping matrix") providing a simple means to connect product entities to specialty domains. This matrix depends on the principle that a marked intersection places a demand on that discipline to identify one or more requirements for that entity using the models of that discipline. Finally, neither UML nor SysML has an organized way to deal with the environment that can be satisfied by a three-layered model like that shown in figure 3 to cover system, end item, and component levels, supplemented perhaps by a special software environment that is significantly different. For software one must only deal with the machine environment, but with other entities it is necessary to consider the natural, non-cooperative, cooperative (developed as external interfaces), hostile, and self-induced environmental relationships within the context of information, energy, and material influences.

Interest in this universal approach was initiated by a long-term effort on my part to more closely link the modeling work that one accomplished in TSA with the specification structure. The results of that effort are visible in the relationship between the universal specification and

continues on next page

Grady *continued*

"modeling" and "requirements analysis" blocks of figure 1. The thought finally occurred to me that this same pattern could be realized for other modeling approaches. At the same time as this project was just about to come to fruition, the Los Angeles chapter of INCOSE showed some interest in a tutorial on what I called at the time "combined modeling," which forced my thought process along more rapidly than it would have evolved without that stimulus.

The numbers in the common specification blocks of figure 1 are specification paragraph numbers, and paragraph 3.1 is focused on modeling the problem space, dealing more generally with that space than the states and modes reference in the military standards. Paragraph 3.1.1 captures the non-modeling sources (need in a system specification and an item purpose in lower-tier specifications plus ad hoc stakeholder requirements). Paragraph 3.1.2 provide a space to capture the results of problem space modeling work accomplished using one or some set of modeling methods, which in this paper is intended to be a universal framework for describing problem space. Paragraph 3.1.3 provides a space to capture the result of solution space (product entities, interfaces, specialty engineering, and environmental) modeling. Paragraphs 3.2 through 3.5 capture the requirements derived from the modeling work, with capabilities and related performance requirements in 3.2, interface requirements in 3.3, specialty engineering requirements in 3.4, and environmental requirements in 3.5. The relationships between the specific modeling artifacts and the requirements are captured in a requirements analysis sheet that is located in Section 6 with all other traceability data.

I do not recommend that you include the results of the modeling work in the specification unless you are developing only a single specification for a program. Alternative and preferred capture methods include (1) a reference to the computer modeling tool used to accomplish the work or (2) an applicable document reference to a separate document that I call a "system architecture report," within which one would find all of the modeling work products, the versions of which could actually be configuration-controlled as part of the system baseline.

It is possible today for a development organization to select a set of modeling approaches, train their employees to apply them well, and employ them on programs to describe the architecture of systems under development, populating a set of program specifications with requirements clearly derived from models. The advantages of this capability over an ad hoc approach or even the application of a disconnected modeling pair (one for system and hardware and one for software) seem obvious, but recent experience suggests otherwise. There exists a resistance to modeling but those resisting do not seem to be able to offer an effective alternative. The difficulty is not in writing requirements; it is in knowing what to write them about and in assigning appropriate values to the attributes they are intended to control. Success in the former is encouraged by modeling that provides the attributes we must control; and the latter, by good domain engineering. A modeling approach encourages that the specifications contain all of the essential characteristics and no unnecessary content, a target that will never be struck in the bulls-eye through an ad hoc, flow-down (or any other cloning method), or customer question-and-answer approach. The method for creating a universal architectural description framework should be coupled with a top-down development direction and a "form-follows-function" orientation when developing unprecedented systems for both hardware and software entities. The original object-oriented analysis (OOA) approach encouraged people to first discover the objects and then to examine them from a functional perspective with data flow diagrams and from the behavioral perspective with state diagrams. This approach encourages a "function-follows-form" sequence that drives a stake into the heart of systems engineers. It is entirely possible to operate UML and SysML as well as PSARE in the top-down, form-follows-function pattern, with the result that all development activity follows the same development pattern. This method simplifies the management that is applied to integrate product- and process-development teams that are formed around the product entities. Some software engineers claim that the product entity structure should be identified as a prerequisite to functionality to avoid the problem of lower-tier teams solving the same problem differently. There is no reason why software development cannot apply the hardware development pattern of progressive integration and optimization plus the use of standard parts (why not including software?), materials, and processes.

An alert software engineer might inquire at this point if we are not a little too late with an effective way of linking modeling artifacts to specification content, given that printed specifications are not really necessary any longer. As an alternative to capturing requirements in published specifications on paper that contain complete sentences, one could employ a primitive form expressing the attribute being controlled, the value and units, connected by a relationship such as "equals," "less than," or several other possibilities, and then capture them in a database, where they may be viewed directly.

Our gifted software engineer might further inquire why we bother writing down the requirements separately from the model. Some people believe that if we model the problem space well, the requirements should be expressed in the model—they must be in there for we claim we derive them from the model. If we extend the problem space model properly into a solution space model, that new model will respect the requirements. If the models were assembled using formal methods and the models were executable, there might be added encouragement for this attitude. As appealing as this picture is, there are many management, legal, configuration management, verification, computer tool, and contracting issues to be resolved before we can enter this world.

Perhaps one could claim that the proposed universal architecture description framework is a model-driven process. But we should recognize that there are three different plateaus in model-driven development: manual, semi-automatic, and fully automatic. Programs are commonly organized today into teams, with each of the specialized members of those teams employing some form of computer database for their work products. These databases form islands, and integration occurs internally within the teams as well as between these islands through human communications in meetings, reports, and conversations. In what could be called a manual model-driven implementation, a conclusion by a mass properties engineer on one team from his mass properties model that should have an effect on the gain in a control system equation under the responsibility of an engineer on another team is communicated verbally and approved if it is deemed appropriate through some form of integrated evaluation.

It is entirely possible that data may be included on the islands that are in error or inconsistent and that these problems may go unnoticed, as a result of an ineffective integration effort that would lead to the problem being discovered too late. What if these islands were interconnected with appropriate computer software to detect data on the different islands that are inconsistent or in error? We could consider this a semi-automatic application of model-driven development, which would result in notification of the appropriate team members, followed by the actions discussed under manual operation. If the sensing of errors were coupled with automatic changes to the data on the islands to remove the inconsistency, we would have a fully automatic application of model-driven development. This relationship will cause a great deal of disbelief and dislike on the part of program managers but if this capability were perfected it would be possible for an enterprise program to accomplish the development process with competitive advantage over those who are incapable of applying the automatic approach. It is likely that the automatic implementation will eliminate a lot of errors of omission, but it is also likely that it will it cost a lot in errors of commission during its introduction.

Industry may progress through these three model-driven capabilities over an extended time frame, but our immediate problem is to make the manual method serve our needs more perfectly in the near term. I believe that the universal architecture description framework approach introduced in this paper will have that effect. I am continuing to work on (1) developing an inventory of modeling artifacts that will be linked to specific models that have been described in textbooks and standards; (2) for each modeling artifact, defining identifications that can be referenced in a requirement allocation sheet for requirements derived from the model; (3) doing a final evaluation of PSARE for completeness; and (4) preparing a clear description of the application of the universal architecture description framework approach in some combination of courses and books. The Los Angeles INCOSE chapter's tutorial at Cal Tech and a one-day course at the University of California—San Diego in January 2008 were initial steps in this process. UCSD will offer a quarter-length course in the fall of 2008, and Elsevier is evaluating a new textbook on the universal architecture description framework.

End Notes

1. U.S. Department of Defense, *CMAN 80008A, Data Item Description, System/Segment Specification* (Washington, DC: Department of Defense, 1988).
2. U.S. Department of Defense, *MIL-STD-498, Software Development and Documentation* (Washington, DC: Department of Defense, 1994); now maintained as *EIA J STD 016*.
3. U.S. Department of Defense, *MIL-STD-961E, Defense and Program-unique Specifications Format and Content* (Washington, DC: Department of Defense, 2003).
4. U.S. Department of Defense, *DoD Architecture Framework, Volume 2: Product Description*, Version 1.0 (Washington, DC: Department of Defense, 2003), paragraph 1.1.
5. Louis H. Sullivan, *Kindergarten Chats (revised 1918) and Other Writings* (New York: Wittenborn, Schultz, 1947).
6. D. Hatley, P. Hruschka, and I. Pirbhai, *Process for System Architecture and Requirements Engineering* (New York: Dorset House Publishing, 2000).
7. Jeffrey O. Grady, *System Requirements Analysis* (Burlington, MA: Elsevier Academic Press, 2006).
8. U.S. Air Force Systems Command Manual 375-5, *System Engineering Management Procedures* (Washington, DC: U.S. Air Force, 1964).

INSIGHT

Addressing Complex Systems

John Colombi, john.colombi@incose.org; and Joseph Carl, joseph.carl@incose.org

INCOSE's Systems Science Enabler Group organized a themed issue of ***INSIGHT*** in January 2008 on the theoretical foundations and scientific underpinnings of complexity in systems engineering. In response to a widely-held opinion that systems are getting more complex (Rouse 2003; Ottino 2004), it therefore seems useful to understand some sources of the increased complexity in modern physical, organizational, and software systems. In that issue (White 2008), several authors offered new extensions of systems engineering to address complex systems. Hybertson and Sheard presented characteristics of complex systems that our community must address: evolution/change, scale, risk and control. Definitions for complexity were offered by Hayenga, and several properties and perspectives were discussed for analyzing "complex" systems by Honour, Schoening, Ryan, Warfield, and Ferris. These properties included boundary, emergence, team ethnicity, and behavioral considerations.

So it seems useful to focus on a few practical aspects of system engineering to best address complexity in the present. This work builds on Doug Norman's question, "So what should I [the program's chief engineer] go do?" Complex interactions may overwhelm our ability to analyze and predict overall system behavior whenever we integrate adaptive software systems, real-time hardware, users, their business processes and entire organizations. But perhaps current systems engineering practices, principles, and tools can already address some of the complexity in modern programs.

Complexity of Systems

With the advent of computers that communicate and communications systems that compute, we now observe that the information-technology infrastructure of an enterprise is a complicated set of systems. These diverse types of equipment serve multiple stakeholders, support multiple activities and tasks, and are built by many different vendors. In the defense community, an even more complex system of systems may include constellations of space surveillance spacecraft, unmanned combat air vehicles, advanced manned strike aircraft, together with ground or airborne command-and-control systems. These components may be operated and maintained by different agencies, organizations, services, or coalition partners. We see that the complexity of modern systems does not necessarily have to come simply from the increased complexity of the individual components, even though the components are individually more complex. The interfaces among these diverse components are also more complex. The number of involved owners and stakeholders is greater, so we need to consider aspects of modern systems that are not just hardware and software. And since no commercial product or government system is static, the components today undergo frequent revisions on their own varying schedules, and these effects on the enterprise must also be considered.

We see that the components we use to construct modern systems are complex enough to be called systems in their own right, hence the phrase *system of systems*. Much has been written on the characteristics of system of systems, which include managerial and operational independence of the component systems, evolutionary and adaptive capabilities, emergent behavior, and distributed components in space and time (Maier 1998). Some have stated that complex systems simply have "properties not fully explained by an understanding of its component parts" (Gallagher and Appenzeller 1999). Thus, a full understanding of these adaptive systems may not arise just from the added complexity of the constituent components, but from the unique characteristics of the components' relationships, as mediated by their interfaces. Does this imply that the skills needed to deal with the complexity are any different from the skills historically employed by systems engineers?

Physical Complexity

Let's begin examining how complexity arises by considering that the protons and neutrons that comprise the nucleus of an atom have no discernable chemistry because their behavior is entirely explained by quantum physics. But as soon as we add electrons to a nucleus to form an atom, a vast spectrum of chemical interactions opens up. Electrons are "captured" by an atomic nucleus to form an atom, and then atoms can bond chemically through the exchange of electrons with other atoms. Thus the components of atoms are complex in their own way, but there is no hint of the complexity of chemical interaction immanent in either nucleons alone or electrons alone. One way to describe what atoms do is to say that the addition of electrons to nucleons opens up a new kind of interactive freedom. This is perhaps a good example of what we mean by emergent behavior in fundamental physics.

Clearly, it is not at all easy to predict what the interactions of different complex physical components will produce when the interactions result in new degrees of freedom. Moreover, a useful heuristic is that unconstrained interactions of different complex components will typically result in new kinds of interaction freedom.

Biological Complexity

Biological entities, including people, possess a wide spectrum of characteristics and abilities, and some degree of independence of action (even if limited) within an environment. They have motivations that are modulated by their experiences and genetic inheritances. The environments of most biological entities are physical, and these entities compete for resources within their physical environments. In addition, the physical environments provide non-linear feedback that influences the future behavior of the entities (e.g., such as when anaerobic bacteria produced so much oxygen as a byproduct of their metabolism that they poisoned themselves), and that feedback drives evolution.

Each human being is a very complex, system, which is non-stationary, non-linear (e.g., evidence has accumulated that we can not talk on a cell phone while driving a car and execute either task as well as we could individually, so we see a kind of saturation non-linearity: our multitasking output is not the sum of individual outputs), and vaguely causal (e.g., humans have internal drives and intentions that partially determine their behaviors absent of external inputs). Karl Popper (1994) has argued that interacting people create cultures that are so complex that they become a virtual environment that surrounds groups of people, and recent evidence supports the notion that people also adapt to their cultural environments. Thus when we think about an organization, a community, a society, or a culture we see that they are all comprised of extraordinarily complex components that interface in non-linear ways that often defy explanation. We can expect interacting people to create and exhibit new kinds of interactions that are not evident in individual behaviors.

Considering the ten billion neurons in the human brain and the trillions of biochemical neural interactions, it should not surprise anyone to observe complex, unpredictable human behavior. The six billion people on the planet connect to form the system known as humankind.

Software Complexity

Having discussed two kinds of complex system interaction that result in new degrees of interaction freedom, let us now consider what happens when we introduce software systems into the mix. Unlike physical and biological systems, software is only virtual. It is a logical entity that has virtual stored knowledge and structure. This implies that software complexity may be more difficult to deal with than physical complexity, because software is today only loosely constrained by physical machinery.

Like the large numbers of human neurons, or the large number of people making up cultures, software components are also complex—increasingly so. Take the F-22 aircraft: over 80% of its functionality is either software or software-controlled hardware with about 2.5 million lines of code. The F-35 Joint Strike Fighter will use about 5.6 million lines of code to handle its variety of missions. But does software have emergent behaviors like atoms and human cultures? Any emergent behavior would not be within a software component, but would instead arise from the interaction between interconnected adaptive software components. It is the coupling or interfaces that facilitate emergence, and not all emergent behaviors are good.

Interface Management

Without trivializing complex systems, perhaps more emphasis could be placed on interface management. It has always been an important aspect of systems engineering. Maier and Rechtin (2002, 274) state this design heuristic: "The greatest leverage in architecting is at the interfaces… the greatest dangers are also at the interfaces." According to the Defense Acquisition Guide, an interface is defined as "the functional and physical characteristics required to exist at a common boundary or connection between persons, between systems, or between persons and systems." Interface management can then be broadly considered the management of communication, coordination, and the transfer of responsibility across a common boundary between two organizations, project phases, or physical entities that were formerly independent.

During an interface control document (ICD) review of one space program, we examined the increasing challenges of interface documentation and interface management. Over a three-year period following the award of the contract, we examined 596 program engineering items. A partial list of these submitted and processed items included requirements changes, specification updates and clarifications, ICD changes, verification plans and a variety of other systems engineering management documents. ICD-related actions comprised 190, or one third, of the total number of actions. A second aspect of interface management to this program was found in relation to contract modifications. As of this study, 77 contract modifications had been issued after the critical design review. Of these 77 changes, 43 (over 50%) were in some way related to ICDs either through studies, updates to an interface specs or implementation/ requirement changes. Interestingly, ICD-related issues resulted in nearly $31.5 million (or 44%) of the cost impact to this space program. Although just one example, it is an indication of the huge challenges relating to managing interfaces, ICDs, and configurations facing complex space systems and their logically-connected payloads, space vehicles, ground terminals, mission control, and cross-links.

But unfortunately, interface management alone is insufficient to understand the complexity within a system-of-systems. If we reconsider the sharing of electrons between atoms, we find we must look to the net polarity of atoms that may lack an electron in order to understand molecular binding. Such net positive polarity in an atom attracts electrons that belong to other atoms, and the two atoms share the electrons. This is a new degree of interaction freedom. Thus, chemistry as an emergent property arises from the sharing of electrons among atoms, binding atoms into molecules. Likewise, for network-centric information systems, perhaps focusing on the network interface may be equally insufficient. One must scrutinize the internal properties and behaviors of the logically-connected systems, as well as the end-users who find, fuse, modify, and ultimately use the shared data.

Interoperability Measurement

Software and information systems bear a significant portion of responsibility for the existence of interoperable and effective complex system of systems. Two competing measures of "goodness" in software design and development are the design patterns of low coupling and high cohesion. Coupling strives to reduce dependence between components within an alternative definition of software architecture. Cohesion strives to keep a component functionally related, given only a few related responsibilities. Cohesion helps to accomplish software maintenance efficiently, but with a consequence of producing many smaller components. This high-cohesion principle breaks a design apart, competing with low coupling, which pulls components together. Therefore, these fundamental design principles must be considered across complex systems: highly coupled collections of simple cohesive compo-

continues on next page

Colombi *continued*

nents may interact to create dynamic, unpredicted, emergent behavior.

The software community has long examined measures of complexity, especially during initial design and the post-release evaluation analysis. Both syntactic and semantic expressions have been proposed. Such a famous work as McCabe's graph-theoretic Cyclomatic complexity measures the number of possible logical paths through a software component. Several authors have noted that many measures of software sometimes link to failure rates, modification effort, or maintenance costs. This is predicated on the idea that easily obtainable properties of a software module (or change to a module) have a significant impact on the risk of failure. In another study, researchers identified such factors affecting complexity as size, module coupling, logic structure, information flow between modules, and data structures of module interaction. While the number of references to software complexity is outside the scope of this short paper, much work has already been realized and is readily and practically applicable. The ability to measure complexity, and the coupling of our large software-intensive systems may increase our ability to manage and control the development of these systems. With the growing percentage of integration and interfacing to commercial off-the-shelf software, interface management becomes more important to this domain.

Related to interface management is the ability to measure characteristics of a system of systems. For nearly thirty years, both government organizations and industrial firms have actively researched interoperability measurement with the goal of creating a straightforward way of measuring and reporting, and then improving the interoperability of complex networks of people, equipment, processes, and organizations. Researchers have created frameworks and models, proposed measures, described levels, and listed a variety of qualitative factors in support of an interoperability measure. We have uncovered nearly three dozen definitions of interoperability, over five dozen distinct types of interoperability, numerous interoperability attributes, and fourteen foundational interoperability measurement models and methodologies. Thus, analysts and engineers have attempted and need to continue to quantify or qualify aspects of the interfaces between man-made physical systems, software and information systems, humans, and organizations.

Human Systems Integration

Researchers have studied how humans work with computers since the inception of modern computing. As computers became more ubiquitous, programmers recognized the need to study best practices for the user interface. This body of knowledge has matured and expanded in perspective and is now referred to as the study of human–computer interaction (Shneiderman and Plaisant 2005). In aviation and military environments the demand for highly effective human–computer interaction is paramount. Avionics designers are now sufficiently confident with the reliability, functionality, and affordability of flat-screen displays to incorporate "glass cockpits" in current aircraft. The glass cockpit gives developers a design freedom never before experienced. However, like any frontier, the lack of constraints increases the complexity of the decisions to be made and highlights the need for sound principles.

So while it is clear that interfaces are important to most complex systems, studies support the idea that the user interface may be of the greatest importance. In an article in the *Naval Engineers Journal*, Malone and Carson (2003) document how the way to harvest the "low hanging fruit" of performance improvement lies at the interfaces between users and computers. One cost study by the U.S. Department of Defense identified manpower, personnel, and training as 40% to 60 % of the total cost over the system's lifecycle. Dray (1995) quantifies the direct tradeoff between a well-designed user interface and reductions in these costs. She cites a project in which an improved user interface on a large-scale application resulted in a 32% overall rate of return, stemming from a 35% reduction in training and a 30% reduction in supervisory time. Thus, an increased focus on human–computer interaction and the more-broadly-scoped field of human systems integration, which is represented by an INCOSE working group, could well be another current practice of systems engineering to address the design and analysis of complex systems.

Interface Emphasis

It is through the interfaces that the purposes of the greater system are realized. Novel behavior (or even emergent behavior) occurs because disparate systems can augment their capabilities through interfaces. It is the effectiveness of the interfaces that determine a system's efficiency. If these interfaces are well defined and well managed, the transitions are smoother and system performance is improved; thus creation happens more efficiently.

References

Dray, S. 1995. The importance of designing usable systems. *Interactions* 2 (1): 17–20.

Gallagher, R., and T. Appenzeller. 1999. Beyond reductionism. *Science* 284: 79-109.

Maier, M. W. 1998. Architecting principles for systems-of-systems. *Systems engineering* 1 (4): 267-284.

Maier, M., and E. Rechtin. 2002. *The art of systems architecting.* 2nd ed. Boca Raton, FL: CRC Press, 274.

Malone, T. B., and F. Carson. 2003. HSI top down requirements analysis. *Naval Engineers Journal* 115.

Ottino, J. M. 2004. Engineering complex systems. *Nature* 427: 399.

Popper, K. 1994. Knowledge and the mind-body problem. In M. A. Natturno, ed., *Defense of interactionism.* London: Rutledge.

Rouse, W.B. 2003. Engineering complex systems: Implications for research in systems engineering. *IEEE Transactions on Systems, Man and Cybernetics*, part C, 33 (2): 154–156.

Shneiderman, B., and C. Plaisant. 2005. *Designing the user interface*. 4th ed. Boston: Pearson/Addison-Wesley.

White, B., ed. 2008. Systems science: Deepening our understanding of the theory and practice of systems engineering. Theme section with articles by B. White, J. N. Warfield, S. Cook, D. Hybertson, C. Hayenga, E. Honour, B. Schoening, A. Ryan, D. Norman, T. Ferris, and S. Sheard. *INSIGHT* 11 (1): 5–32.

Technical Directions

From the Technical Director

Dick Kitterman, richard.kitterman@incose.org

In the last issue of ***INSIGHT***, I wrote about the need to work with people to build consensus, to further evolve our technical operations, and to engage the working groups and the other parts of the Technical Leadership Team with each other. This will allow us to better contribute to INCOSE's vision, strategy, and value. Since that article was published, my new deputy technical director Regina Griego and I have been working to do just those things. This article summarizes what we have accomplished so far and projects the direction in which we are heading.

First, let's talk about consensus building and evolving our technical organization, because they go together. INCOSE's technical organization has evolved over the years. Our Technical Leadership Team was put in place in 2004 after considerable thought and effort. The current structure of the Technical Leadership Team is based on consideration of what is needed to perform systems engineering, such as is illustrated in figure 1.

Certain things, such as knowledge, processes, and technology are required to enable system engineering to be done. Further, once enabled, systems engineering can be done in the application sectors that INCOSE's mission indicates: industry, academia, and government. If this concept is taken through further steps of analytical decomposition, the result is a technical matrix as shown in figure 2. The current Technical Leadership Team is a direct realization of this matrix in an organizational construct.

After quite a bit of "floating of ideas" (so-called "thought models"), as well as e-mail discussions and telephone conferences, we have arrived at a consensus basis for a simpler organizational approach that follows figure 1. We believe that this consensus will help us encourage more communication and interaction among INCOSE members. In addition, in every way we can think of, we are ensuring that the working groups are clearly identified and treated as an integral, dynamic part of INCOSE's technical organization.

While merely changing the label doesn't change what's in the can, we are renaming the organization from the Technical Leadership Team to Technical Operations. Technical Operations' structure and who's who will be addressed in a separate announcement and reflected in updated material on the INCOSE Web site at http://www.incose.org/practice/techactivities/index.aspx. The mission of this technical infrastructure is to provide technical information by means of technical events, technical products, technical interactions among stakeholders, and technical information repositories. By taking this evolutionary step we intend to provide Technical Operations with a supporting technical infrastructure that anticipates and responds to the technical information needs of all INCOSE stakeholders.

SE Enablers \ Application Sectors	Aerospace & Defense	Market Driven Products	Emerging Technologies	Enterprise	Information Systems	Infrastructure	Public Interest	Transportation
Systems Science								
SE Technical Process								
SE Management Process								
SE Support Process								
Modeling & Tools								
Specialty Engineering								

Figure 2. The INCOSE technical matrix

This is a start, but not an end. We are considering other "thought models" that aim at several targets: increasing the value to our stakeholders of our standards activities by increasing interaction on a planned basis with our working groups; increasing our focus on products; following a well-thought-out product roadmap (including analysis of what structure, content, and form of publication our next *Systems Engineering Handbook* should take); and building a model of Technical Operations that aligns with a model of INCOSE and its mission. At the same time, we continue to work to increase the visibility of our vital working groups and their contributions.

We will undoubtedly make a few stumbles along the way, and not everyone will agree on everything. But for now we can do our best to engage everyone in an open process of figuring out where we should go, and then ask in return that they join with us on that journey.

My e-mail address is given above, and I solicit your thoughts.

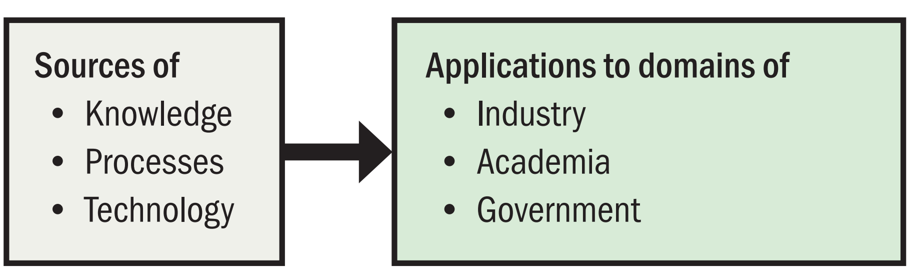


Figure 1. Systems engineering draws on knowledge, processes, and technology to develop its capabilities, and then applies them to multiple domains

Technical Activities

Standardized Approaches to Modeling Systems of Systems

Ron Williamson, ron.williamson@incose.org

INCOSE's model-based system engineering (MBSE) initiative is focused on promoting, advancing, and institutionalizing the practice of MBSE to attain the goals of the INCOSE's Vision 2020. To accomplish this, the initiative includes a set of strategic activities to develop methods, tools, education and training programs, standards, and research; the initiative also includes the Model-based Systems Engineering Challenge, in which a set of teams prototype and demonstrate the practices. One of our activities deals with systems of systems: this MBSE activity seeks to determine what modeling and system engineering capabilities are necessary to develop enterprise-wide, system-of-systems solutions in a more cost-effective, timely and high-quality manner than has been done before. A system of systems may be defined as "a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities."[1]

Developing the vocabulary, methods, and tools in support of enterprise architectures is a critical element of the system-of-systems project's top-down strategy for developing needs, as well as its bottom-up strategy to use best practices and tools from industry. For model-based development to succeed, we will have to develop model-based standards that clearly define the meta-models for enterprise architectures and other models related to enterprises. The partnership and cross-membership between INCOSE and the Object Management Group (OMG) has produced a productive synergy of ideas and methods and continues with the ongoing efforts to develop a unified profile for military architecture frameworks.

One important aspect of this project is the ability to model the concepts, relationships, attributes, and constraints associated with architecture frameworks for enterprises and systems of systems. A team, composed of INCOSE and OMG members, was formed to build on previous efforts within the OMG to develop a modeling standard that supports the architecture frameworks of both the United States and the United Kingdom (the U.S. Department of Defense Architecture Framework [DoDAF] and the U.K. Ministry of Defence Architecture Framework [MODAF]). This modeling standard is called the UPDM, which stands for Unified Profile for DoDAF and MODAF.

UPDM defines an industry standard description of enterprise architectures. The architecture description complies with DoDAF (version 1.5) and MODAF (version 1.2). UPDM, an initiative of the Object Management Group, intends to standardize the language for defining architecture, using UML, SysML and other OMG standards to represent the architecture frameworks of both countries. This new unified profile is expected to result in significant improvements in the consistency, quality, and tool interoperability of enterprise architectures that comply with these frameworks. In addition, it is expected to be fully compatible with SysML models for modeling system-level specification and design.

In addition to developing a specification that fully supports both architecture frameworks, which is essential for organizations developing systems for network-enabled capability, the UPDM team will also make use of NATO's recently-adopted standard for architectural frameworks, NATO Architecture Framework (version 3.) This standard is based on MODAF but has been extended to support service oriented architectures. Although the UPDM team is independent of the Object Management Group, it plans to submit a new specification to the OMG using its fast-track "request for comments" adoption process at the OMG conference as early as September 2008, with the goal of having it adopted by the OMG Architectural Board at the next quarterly meeting.

The UPDM team has already defined working groups to focus on specific aspects of the specification, and plans to set up a forum to enable interested parties to keep up-to-date with progress on the specification. The membership of the UPDM team includes vendors of development tools and contractors for the defense industry, along with representatives of the American and British defense administrations.

In addition to its effort to develop the UPDM, the MBSE Initiative's system-of-systems activity intends to leverage and influence the ongoing standards-development efforts to develop model-based approaches to engineering enterprises and systems of systems. Based on feedback from industrial firms, the system-of-systems activity will focus on other important issues, such as executable models, business structure and behavioral models, service-oriented models, security models, and information models.

1. Director of Systems and Software Engineering, *System of Systems Systems Engineering Guide, version 1.0 draft* (Washington, DC: Deputy Under Secretary of Defense Acquisition and Technology , 2007), 12, § 1.5.1, lines 397–398.

INCOSE Operations

CSEP Beta Test Is Completed and the New Tests Are Ready to Go Live

Kevin Forsberg, kevin.forsberg@incose.org

At the end of March 2008 the final scoring for the Certified Systems Engineering Professional (CSEP) beta tests were completed, and the pass-fail criterion was established for the two new systems engineering certification exams. Although the operational core exam has 120 questions and the operational acquisition exam has 60 questions, the beta exam had 537 questions covering both. The 80 beta test volunteers were given ten hours to complete the entire set. The beta exam was conducted at two sites, first on 30 January 2008 at the International Workshop in Albuquerque, New Mexico (with 25 people), and second on 9 February 2008 at Ft. Belvoir, Virginia (with 55 people). All of INCOSE owes especial thanks to the international team of 80 volunteers, and to Karl Geist, who solicited and organized that crew. The beta test group consisted of 19 systems engineering novices, 34 "average" systems engineers, and 27 "experts." Just as in the first beta test in 2004, this mix was deliberately chosen to give the exam developers the necessary insight to determine the cut-score (the pass/fail limit).

The CSEP core exam is a two-hour, 120-question test that all candidates for Associate Systems Engineering Professional (ASEP) and Certified Systems Engineering Professional (CSEP) must pass. An additonal one-hour, 60-question exam is for those seeking certification as U.S. Department of Defense Certified Systems Engineering Professionals for Acquisition (CSEP-Acq); this test requires that the candidate already be a CSEP, or that the candidate take (and pass) both exams at the same time. Those who pass the CSEP core exam must also submit a completed application and a written evaluation of their experience before they can receive the CSEP certification. (Note that the process has been revised to allow the candidate to file the application either before or after taking the CSEP exam.)

The new core ASEP/CSEP exam is based on version 3.1 of the *Systems Engineering Handbook.* The acquisition exam is based on the latest draft (forthcoming) of the *Defense Acquisition Guidebook*, chapter 4. Both exams will be available at the Prometric testing sites starting on 1 July 2008.

Based on the strong recommendation from our professional test providers, Prometric and Certification Management Systems, INCOSE will not publish either the pass/fail score or the operational percentage of exam takers who pass. However, as in 2004 for the first beta test, we will reveal several aggregate numbers. The overall pass rate for the CSEP core exam was 70%. The overall pass rate for the CSEP acquisition exam was 59%.

Potential trainers and students alike note that training has proven to be effective. Thirty-eight beta test takers did attend a CSEP core preparation training course (from the several available) a week or more before the test. It is especially instructive that the mix (novice/average/expert) of the two groups—those who did and those who did not take training—was essentially the same. The group of 38 who took training had a pass rate of 79%. For the 42 who did not take training the pass rate was 62%.

From the Member Board Co-Chair

Jonette Stecklein, jonette.stecklein@incose.org

Local chapters play an essential role in the achievement of INCOSE's objectives. Far more than local administrative units, chapters organize a multitude of professional and social programs, conduct membership recruitment and retention drives, support technical activities striving to advance the state and art of systems engineering, and market INCOSE as the international authoritative body on systems engineering. Through the annual chapter awards program, our Council recognizes the valuable contributions of individual chapters as they strive to enrich, educate, and enlighten the membership while improving recognition of INCOSE and the systems engineering profession. The annual awards program includes the Bronze Circle, Silver Circle, and Gold Circle awards to recognize the chapters that meet and exceed INCOSE's standards for local service and contributions.

One of the important events at the International Workshop each year is the assessment of submissions to the annual chapter awards program. Congratulations to the following chapters for the contributions they have made to their communities. These chapters were formally recognized at the International Symposium.

The award-winning chapters for 2008, based on their performance in 2007, are as follows:

§ **President's Award for Outstanding Chapter:** United Kingdom

The President's Award for Outstanding Chapter is presented every year to the one INCOSE chapter that best embodies INCOSE's goals and standards. The United Kingdom chapter has consistently performed as a top chapter in INCOSE and has been a Gold-rated chapter since 2003. Along with a high total scoring, they demonstrated a strong and balanced performance across all seven award areas. The United Kingdom chapter continued follow-up of previous efforts by influencing the strategy of the U.K.'s defense industry through a survey on the current conduct of systems engineering by major industrial players. The United Kingdom chapter has consistently provided contrib-

continues on next page

Stecklein *continued*

utors working at all levels of INCOSE's international organization.

Director's Award for Most Improved Chapter: Colorado Front Range

After being awarded Silver last year and being challenged in prior years, Colorado Front Range had the largest points improvement from 2006 to 2007. Colorado Front Range provides a balanced set of services to their members, across all categories of scoring criteria. In addition, the awards review committee recognizes the Huntsville and Singapore Chapters for making exemplary improvements during the past year. Huntsville has shown consistent improvement over the past three years; Singapore, a new chapter, has matured rapidly.

Gold Circle:

Association Francaise d'Ingenierie Systeme (French Affiliate), Chesapeake, Colorado Front Range, Enchantment, Gesellschaft für Systems Engineering, e.V. (German Chapter), Hampton Roads, Huntsville, Israel, Los Angeles, Netherlands, North Star Chapter, North Texas Chapter, Southern Maryland, South Africa, United Kingdom, Washington Metro Area

Silver Circle:

Atlanta, Finger Lakes, Heartland, Midwest Gateway, New England, San Diego, Singapore, Texas Gulf Coast, Wright Brothers

Bronze Circle:

Liberty

The Member Board appreciates the efforts of the award review team: Don Boyer, cochair and Francis Peter, cochair, together with James Armstrong, Joe Carl, Mark DeSpain, Richard Grzybowski, Carol Hutchinson, Bob Levin, Bill Olson, Bruce Shelton, and Phil Simpkins. The award review team performed a valuable service to INCOSE and returned to their chapters with insight into how other chapters operate, new ideas for activities, and approaches for more effective chapter operations.

Chapter News

A Tutorial in Agile Systems and Enterprises

Jack Ring, jack.ring@incose.org

I attended the tutorial, "Introduction to the Design and Engineering of Agile Systems and Enterprises," that was hosted by the Enchantment Chapter of INCOSE on 12 March 2008. INCOSE member Rick Dove, Industry Professor at the School of Systems and Enterprises of the Stevens Institute of Technology, presented the tutorial. This was a one-day overview of a course he teaches at Stevens. Rick presented a good mix of systems principles, real examples, and case studies, along with brief sessions of audience participation.

This turned out to be well worth the trip from Phoenix to Albuquerque. This material was immediately applicable to all three projects in which I am currently engaged, even though they are quite different. Rick's lecture applied equally well to architecting a unique education system targeted to achieve one thousand campuses world-wide in five years for what is known as K-12 (kindergarten through twelfth grade) in the United States; to designing, architecting, and prototyping computer-aided model-based systems engineering; and to authoring several concepts of operation for a spectrum of situations involving intelligence data qualification and pattern recognition.

I found the material useful both as prompts for designing and as checklist for design reviews. I recommend this tutorial to any other INCOSE chapter looking for interesting program material.

Colorado Chapter Participates in Career Fair

Jerry Huller, jerry.huller@incose.org

On 27 November 2007, two members of INCOSE's Colorado Front Range Chapter, Jerry Huller and Leslie Koshigoe, participated in a career fair at Cherry Creek High School in Englewood, Colorado. The theme was "Math & Science Students: Explore Career Opportunities in STEM (Science, Technology, Engineering, Mathematics) Fields." The Cherry Creek Association for Gifted & Talented and the Cherry Creek School District Office of Gifted & Talented sponsored the event.

Jerry and Leslie presented PowerPoint charts on "Let's Build a Dalmation" to illustrate requirements development within the systems engineering life cycle. They also answered students' and parents' questions on systems engineering at the chapter's table. These photos show the beginning of the presentation to an interested student.

INCOSE Colorado member Jerry Huller and Jessica Lin discuss systems engineering aspects of building a dalmatian.

INCOSE Colorado member Leslie Koshigoe discusses systems engineering as a career with Jessica Lin.

Photos by Jenna Fleur Lin

INCOSE Events

Report from the 2008 Workshop of INCOSE's Systems Engineering & Architecting Doctoral Student Network (SEANET)

Ricardo Valerdi, ricardo.valerdi@incose.org; and **Donna H. Rhodes**, donna.rhodes@incose.org

The objective of the Systems Engineering & Architecting Doctoral Student Network is to enable doctoral-level research in systems engineering. This network has the potential to increase the quality and success rate of doctoral students through collaboration, mentorship, and learning (Rhodes & Valerdi 2007). INCOSE has made a strategic commitment to SEANET by providing funding to make these workshops a reality.

The first workshop was held in 2005 in Tampa, Florida, at the INCOSE International Workshop; the second was held in 2006 at the University of Southern California (USC) in Los Angeles, California, in conjunction with the Fourth Conference on Systems Engineering Research; and the third was held in 2007 at San Diego State University in conjunction with the 17th INCOSE International Symposium. The fourth workshop, which is the focus of this article, was held again at the University of Southern California in conjunction with the sixth Conference on Systems Engineering Research. Next year's workshop is planned for Loughborough, U.K., the site of the seventh Conference on Systems Engineering Research.

Overview of the 2008 Workshop

The workshop began on April 3. Building on the success of previous workshops, the focus was to provide a balance between keynote presentations, short vignettes on research, and opportunities for networking and knowledge sharing. Thirty doctoral students attended the workshop, led by Donna Rhodes and Ricardo Valerdi, from eight different countries. INCOSE president Pat Hale and president-elect Samantha Brown participated in this year's event. The agenda included the following presentations and activities:

- *Dr. Larry Head, professor of systems and industrial engineering at the University of Arizona.* He shared his perspectives on doctoral research in systems engineering, identified a range of areas where he believes research in systems engineering is needed, discussed characteristics of good dissertation research, and suggested some strategies for doing good dissertation research.
- *Lynne P. Cooper, knowledge strategist at NASA's Jet Propulsion Laboratory and doctoral candidate at USC.* She provided an overview of her twelve-year journey as a doctoral student in a motivational talk entitled "From Inspiration to Dissertation," where she discussed the challenges of being a student while working full-time, having an advisor that moved from her university to another but continued working with her, and doing qualitative engineering research. (Note: Lynne successfully defended her dissertation on May 9, making her the newly-minted Dr. Cooper.)
- *Dr. George Friedman, adjunct professor of industrial and systems engineering at USC.* He described his fifty-years of experience as a systems engineer and claimed that his education was "the key to taking me from the most humble of situations to the chief technology officer of a multi-billion-dollar corporation and then to a faculty position at USC." He added "everyone has a desire to return his gifts in return for the blessings he has received. The gifts from my body and heart are my children. The gifts from my mind are the teachings I have been privileged to give my one thousand students at USC. Without the PhD, this would not have been possible."

Seven short vignettes were included to provide workshop participants food for thought:

- Jo Ann Lane (a USC doctoral student) discussed ways in which students can make use of INCOSE's resources to help doctoral research.
- Jimmy Gandhi (a Stevens Institute of Technology doctoral student) provided an overview of the formation of the INCOSE student chapter at Stevens and offered strategies that others can use to imitate this very successful model.
- Caroline Lamb (an MIT doctoral student) shared her experiences in developing a doctoral research proposal.
- Matt Richards (an MIT doctoral student) shared his strategies in packaging doctoral research into publications.
- DeWitt Latimer (a recent USC PhD graduate) shared tips on how to prepare for a dissertation defense. (Note: DeWitt successfully defended his dissertation in March, making him the new Dr. Latimer.)
- Dr. Adam Ross (an MIT research scientist) shared his approach on continuing research after the doctorate. Dr. Ross has been building on his dissertation research to evolve a research agenda in the area of multi-attribute tradespace exploration.
- Dr. Ricardo Valerdi (an MIT research associate) shared the current list of completed doctoral dissertations and encouraged participants to contribute to this database by identifying completed doctoral dissertations from their institutions.

The workshop also offered three "birds-of-a-feather" breakout groups. Dr. Tim Ferris led a group on research methodologies for systems engineering; Dr. Rob Cloutier led one on writing a research proposal; and Dr. Lynne Cooper headed up a group on "preparing for the end game." These interactive sessions

Valerdi *continued*

From left to right: Dr. Ricardo Valerdi (MIT), Dr. Larry Head (University of Arizona), Dr. Donna H. Rhodes (MIT), Lynne Cooper (NASA JPL & USC), Dr. George Friedman (USC). (Photo by Nirav Shah.)

From left to right: Julia Nickel, Mary Bone, Craig Blackburn, Samantha Brown, Deb Chattopadhyay, Di Wu. With his back to us: Tim Ferris. (Photo by Ricardo Valerdi.)

provided the SEANET participants the opportunity to openly share their thoughts and concerns about specific aspects of their PhD journey.

Newly-minted PhDs

In the effort to cultivate the systems engineering research network, one of the functions of SEANET is to recognize the completion of newly-minted systems engineering PhDs worldwide. Below are some examples of SEANET members who recently completed their studies.

Dr. Eric Smith
Degree: Systems and Industrial Engineering
Doctoral institution: University of Arizona
Dissertation title: Tradeoff Studies and Cognitive Biases
Current position: Lecturer, Missouri University of Science and Technology

Dr. Rob Cloutier
Degree: Systems Engineering
Doctoral institution: Stevens Institute of Technology
Dissertation title: Applicability of Patterns to Complex Systems
Current position: Research Associate Professor, Stevens Institute of Technology

Dr. DeWitt Latimer
Degree: Computer Science
Doctoral institution: University of Southern California
Dissertation title: The Effectiveness of Engineering Practices to Support the Acquisition & Deployment of Robotic Systems
Current position: Captain, U.S. Air Force

We hope to continue updating this list as the number of doctoral students continues to grow.

Acknowledgements

Much appreciation goes to the workshop participants and speakers. INCOSE and Prof. Stan Settles provided funding for the workshop. Georgia Lum of USC provided logistical support.

Reference

Rhodes, D., and R. Valerdi. 2007. Enabling research synergies through a doctoral research network in systems engineering. *Systems engineering* 10 (4): 348–360.

Could You Say That Again, Please?

Cecilia Haskins, cecilia.haskins@incose.org

This is the final installment of a three-part series of articles that introduce the Region III chapters to the rest of INCOSE. As with the earlier features, each article is written in the native language of the country side-by-side with English translation. I am pleased to present these introductions in this issue; they are from INCOSE's French affiliate, the INCOSE Turkey chapter, and our colleagues in Italy who are working on forming a new chapter. We hope you have enjoyed this series and that this will start a trend among other chapters.

AFIS (Association Française d'Ingénierie Système)

Jean-Claude Roussel, jean-claude.roussel@incose.org

Crée en 1999 par treize grands groupes industriel, l'AFIS est une association encore très jeune mais connaît une progression significative depuis sa création. Elle regroupe aujourd'hui 23 sociétés membres, dont 5 universités ou grandes Ecoles et plus de 450 adhérents individuels, ce qui en fait l'un des groupes les plus importants de l'INCOSE.

L'AFIS est membre affilié de l'INCOSE, et par un protocole d'accord signé en 1999 et mis à jour en 2006, son représentant en France. Tout adhérent à l'AFIS se trouve de fait automatiquement membre de l'INCOSE. Les groupes de travail au nombre de treize actuellement sont la force vive de l'AFIS. Ils permettent aux adhérents de confronter leurs expériences et de travailler en réseau sur les différents thèmes de l'Ingénierie Système. Depuis quelques années, des liens étroits ont été établis entre certains groupes de travails de l'AFIS et leurs homologues à l'INCOSE (Ingénierie des Exigences, V&V, Systèmes Résilients).

Des conférences ou séminaires sont organisées tous les ans, en interne ou en partenariat international (EUSEC 2002, IS 2004, IS2008). A cela, s'ajoute des journées thématiques (à Paris, Toulouse, ou d'autres villes de France) ayant pour but de traiter en une journée un thème précis de l'ingénierie système à partir de plusieurs communications.

L'AFIS accorde une place très importante à l'enseignement et la recherche, et à ce titre organise également depuis quelques années des Forums Académiques (Toulouse 2006; Nancy 2007, objet de ce numéro d'***INSIGHT***; et Nîmes 2008) où se réunissent enseignants, chercheurs, étudiants et industriels.

Enfin, une nouvelle organisation est en train de se mettre en place au sein de l'AFIS pour répondre aux attentes exprimées par les adhérents et mettre en oeuvre son plan d'action tel que défini dans sa stratégie. Parmi les axes majeurs de celle-ci figure une ouverture très forte vers l'international, à travers une coopération avec l'ensemble des chapitres de l'INCOSE.

Introduction to AFIS

Jean-Claude Roussel, jean-claude.roussel@incose.org

AFIS is a French association created in 1999 by thirteen industrial corporations. Although a young association, it has grown significantly since its creation, such that today there are 23 corporate members including five universities and more than 450 individual members.

AFIS is affiliated with INCOSE through a memorandum of understanding signed in 1999 and updated in 2006. AFIS represents and promotes the INCOSE mission in France. This means every member of AFIS is automatically a member of INCOSE. Thirteen working groups are the lifeblood of AFIS. These groups allow the members to compare their experiences and to network on various systems engineering themes. A few years ago, close partnerships were set up between selected AFIS working groups and their INCOSE counterparts (Requirements, Verification and Validation, and Resilient Systems).

Annual conferences and seminars are organized throughout the year, either internally or in conjunction with international events such as EUSEC 2002, IS2004, and IS2008. Also, one-day events are held in Paris, Toulouse, or other French cities, each one centered on a specific theme of systems engineering as a way of sharing information.

AFIS attaches great importance to education and research and for the past three years has organized an annual Academic Forum (Toulouse 2006; Nancy 2007, which is the subject of this issue of ***INSIGHT***; and Nîmes 2008) as a meeting place for professors, researchers, students, and other practitioners of systems engineering.

In closing, a new organization is being established to respond to the expectations of the members and to realize an annual operational plan as defined in the AFIS strategic plan. Among the major initiatives of this plan are increased international overtures and cooperation with other INCOSE chapters.

INCOSE Türkiye Kolu Tanıtımı

Halil Kizilca, halil.kızılca@incose.org

INCOSE Türkiye Kolu 12 Nisan 2005 yılında kuruldu.

Türkiye Kolu, kurulduğundan bu yana herbirine yaklaşık 80–100 Sistem Mühendisinin katildigi beş adet Ulusal Sistem Mühendisleri buluşmasi gerçekleştirdi.

Herbirinin kendi teması olan bu etkinliklerin ortak amacı: sistem geliştirme faaliyetlerinde bulunan sistem mühendislerini, akademik kurumları ve satın alıcıları biraraya getirerek daha iyi bir Sistem Mühendisliği anlayışına ulaşmaktı.

Bu kapsamda, Teması "Sistem Mühendisliği Uygulamalari" olan Birinci Ulusal Sistem Mühendisleri Buluşması 13-14 Ekim 2005 tarihlerinde Ortadoğu Teknik Üniversitesi, Ankara'da düzenlendi.

Teması "Sistem Mühendisliği Tanım ve Kavramları" olan Ikinci Ulusal Sistem Mühendisleri Buluşması 03 Subat 2006 tarihinde Ortadoğu Teknik Üniversitesi, Ankara'da düzenlendi.

Teması "Sistem Gereksinim Mühendisliği" olan Üçüncü Ulusal Sistem Mühendisleri Buluşması 24 Kasim 2006 tarihinde Ankara Üniversitesi, Ankara'da düzenlendi.

Teması "Sistem Tasarimı" olan Dördüncü Ulusal Sistem Mühendisleri Buluşması 4 Mayis 2007 tarihinde Yeditepe Üniversitesi, Istanbul'da düzenlendi.

Teması "Sistem Testi" olan Beşinci Ulusal Sistem Mühendisleri Buluşması 2 Ocak 2008 tarihinde Ortadoğu Teknik Üniversitesi, Ankara'da düzenlendi.

INCOSE Türkiye kolu etkinlikleri üç farklı üniversite ve iki farklı şehirde düzenleyerek daha çok kişiye ulaşmayı ve ulaştığı kişilerin Sistem Mühendisliği alanındaki farkındalığını arttırmayı hedefledi.

INCOSE merkez yönetiminin katkılarıyla Türkiye Kolu gelecekte de uluslararası egitimler ve konferanslar düzenlemeye devam etmeyi hedeflemektedir.

Introduction to the INCOSE Turkey Chapter

Halil Kizilca, halil.kızılca@incose.org

The INCOSE Turkey Chapter was established on 12 April 2005.

Since then, the Turkey Chapter has organized five National Systems Engineering Meetings with 80–100 systems engineers attending each event.

The common goal of each meeting is to bring together systems engineers, academic institutions, and customers in order to reach a better understanding of systems engineering. In addition, each meeting has its own theme.

The first National Systems Engineering Meeting was held 13–14 October 2005 at Middle East Technical University in Ankara. The theme was "Systems Engineering Practices."

The second National System Engineering Meeting was held 3 February 2006 at Middle East Technical University in Ankara. The theme was "Systems Engineering Terms and Definitions."

The third National System Engineering Meeting was held 24 November 2006 at Ankara University in Ankara. The theme was "System Requirement Engineering."

The fourth National System Engineering meeting was held 4 May 2007 at Yeditepe University in Istanbul. The theme was "System Design."

The fifth National System Engineering Meeting was held 2 January 2008 at Middle East Technical University in Ankara. The theme was "System Testing."

The INCOSE Turkey chapter organized its activities at three different universities in two different cities to reach more people and to increase their awareness of systems engineering.

With the support of INCOSE, the chapter intends to continue to organize more national and international conferences and training.

INCOSE Italia

Marco Lisi, marco.lisi@incose.org

INCOSE Italia sarà il nome del Capitolo italiano di INCOSE. Il Capitolo è ancora nella fase "emerging", ma dovrebbe essere completamente costituito entro la fine del 2008.

L'attenzione al "systems engineering" è andata rapidamente crescendo in Italia negli ultimi anni, sia nel settore privato che in quello pubblico.

L'industria dell'aerospazio e della difesa, che ha una lunga tradizione in Italia rappresentata da una delle più grandi "corporation" al mondo, è particolarmente interessata al "systems engineering." La crescente complessità dei sistemi, la rapida evoluzione delle tecnologie e le sfide della net-centricità e dei "sistemi di sistemi" richiedono un approccio olistico e metodologicamente strutturato.

Anche industrie ed aziende fornitrici di servizi in ambito civile, come quelle dei settori ICT e trasporti, sentono la necessità di un approccio sistematico al progetto architetturale ed allo sviluppo di sistemi futuri.

Il Capitolo italiano è stato molto attivo negli ultimi tre anni, organizzando seminari e corsi introduttivi su numerosi temi: la gestione dei requisiti, il "systems architecting," gli "architectural frameworks," il SysML, la sicurezza dei sistemi ed altri.

Inoltre, abbiamo stabilito buone relazioni con istituzioni accademiche, che hanno permesso l'organizzazione di corsi sul "systems engineering" nell'ambito dei programmi di "Master" post-universitari. Negli ultimi due anni, più di quaranta giovani ingegneri hanno ricevuto una buona introduzione al "systems engineering," basata in buona parte sull'Handbook dell'INCOSE. Gli studenti hanno unanimemente espresso interesse e soddisfazione. Alcuni di loro stanno al momento applicando la metodologia del "systems engineering" nelle loro attività professionali e parecchi, con nostra grande soddisfazione, si sono iscritti all'INCOSE.

Un ambizioso obiettivo del Capitolo italiano è anche quello di promuovere attività di ricerca in nuovi campi, quali la sicurezza informatica dei sistemi, la stima dei costi di "sistemi di sistemi" e gli approcci concorrenti e collaborativi al "systems engineering."

Una "newsletter" elettronica dovrebbe presto vedere la luce: aiuterà il Capitolo ad allargare la sua "audience" ed a diffondere la cultura del "systems engineering".

Un nostro obiettivo a medio termine? Ospitare una conferenza regionale di INCOSE in Italia.

Concludendo, siamo grati a due gentili signore, Cecilia Haskins, New Chapters Coordinator, ed Elke Gerngrosse-Leone, INCOSE ambassador, per l'aiuto e l'incoraggiamento ricevuti finora.

INCOSE Italy

Marco Lisi, marco.lisi@incose.org

INCOSE Italia will be the name of the Italian chapter of INCOSE. The chapter is still in an "emerging" phase, but it should be fully established during 2008.

Attention to systems engineering has been rapidly increasing in Italy during the last decade, both in the private and the public sector.

The aerospace and defense industry, which has a long-standing tradition in Italy, represented by one of the largest corporations in the world, is particularly interested in systems engineering. The increasing complexity of systems, the rapid evolution of technologies, and the challenges of net-centricity and of systems of systems demand a holistic and methodologically-organized engineering approach.

Civil industries and service providers, such as the information and telecommunications technology sector and the transportation sector, also recognize the need for a systematic approach to the architecting and development of future systems.

The Italian chapter has been quite active in the last three years, organizing seminars and introductory courses on a number of topics: requirements management, systems architecting, architectural frameworks, SysML, systems security, and others.

In addition, we have established relationships with academic institutions, which resulted in the organization of systems engineering courses in the frame of post-graduate university master's programs. In the last two years, more than forty young engineers have received a more than basic introduction to systems engineering, based to a great extent on the INCOSE Handbook. The students have unanimously expressed interest and satisfaction. Some of them are presently applying the systems engineering methodology in their professional activities, and quite a few, to our satisfaction, have joined INCOSE.

A challenging objective of the Italian chapter is also that of performing research activities on new topics, such as cybersecurity, cost estimation for systems of systems, and concurrent and collaborative approaches to systems engineering.

An electronic newsletter should be started soon to help the chapter to expand its audience and disseminate the culture of systems engineering.

Our goal for the mid-term future? Hosting an INCOSE regional conference in Italy.

In closing, we are grateful to two wonderful ladies, Cecilia Haskins, New Chapters Coordinator, and Elke Gerngross-Leone, INCOSE ambassador, for the help and encouragement received so far.

Industry News

Future Acquisition Directorate Team Leading the Way In System-of-systems Engineering in the U.S. Defense Department

Penny S. Pierce, penny.pierce@navy.mil

Operating under the auspices of the United States Armed Services' Joint Program Executive Office for Chemical and Biological Defense, the Future Acquisition Directorate's Future Systems Team is extending the systems engineering of complex systems to complex systems of systems and families of systems that support the joint forces. To support this focus, the Future Systems Team is concentrating on innovative system-of-systems engineering, along with processes and methodologies for architecting. As a result of developing a "capability-to-component" process for system-of-systems engineering using architecture products, the Future Systems Team developed a process that encompasses capability analyses, system-of-system analyses, and component analyses. The team has also been collaborating with systems engineering and architecture faculty of the Defense Acquisition University (DAU) in order to incorporate the results of practical experience in system-of-systems engineering processes.

The Future Systems Team is developing and using system-of-systems architectures as an integral part of their systems engineering processes. This process couples systems engineering analyses with the architecture products that support them. With the emphasis on capability, the process ties capability to ongoing programs. This process also links gaps and shortfalls back to capability in order to supply decision makers with supporting rationales for future decisions about capability and budget. The system-of-systems engineering process consists of three main activities that are dynamically applied to produce the optimum solution: capability analysis, system of systems analysis, and component analysis.

Capability analysis decomposes operational capability into operational requirements, relating this with the Joint Requirements Office's chemical, biological, radiological, and nuclear architecture and provides inputs to the requirements documents for major defense acquisition programs.

System-of-system analysis decomposes operational requirements to system requirements and provides system-of-system alternatives to meet the requirements for chemical, biological, radiological, and nuclear systems.

Component analysis allocates system requirements to component systems. The analysis evaluates system performance against desired capabilities to identify gaps and shortfalls. Specific products for architecting system-of-systems for chemical, biological, and radiological defense demonstrate the capability threads within the architecture.

The Future Systems Team briefed the systems engineering and architecture faculty from the DAU on the details of the system-of-systems engineering process using architecture products. The purpose of this collaboration is to exchange information on the systems engineering and architecture processes. The DAU will incorporate the results of practical experience with academic principles. The Future Systems Team will adhere to established systems engineering principles as they evolve to support systems of systems.

The Future Systems Team continues to provide state-of-the-art processes, analytical methodologies, lessons learned, and results of practical experience for engineering systems of systems. The team will maintain collaboration with the DAU and other systems engineering teams as they refine and update the processes for system-of-systems engineering.

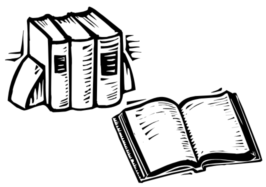
Industry News Links — *from the School of Systems and Enterprises at Stevens Institute of Technology*

■ Naval Postgraduate School and Stevens sign memorandum of agreement to collaborate on systems engineering education and research: http://www.stevens.edu/sse/about/news/single_news.php?news_events_id=1072.

■ Nanyang Technological University signs memorandum of intent with Stevens to leverage the strengths of both institutions towards the betterment of systems and enterprise engineering, architecting, and management through research, graduate education, conferences and executive education: http://www.stevens.edu/sse/about/news/single_news.php?news_events_id=935.

■ School of Systems and Enterprises establishes the Fabrycky-Blanchard Systems Engineering Doctoral Scholarship for full-time systems engineering doctoral students: http://www.stevens.edu/sse/about/news/single_news.php?news_events_id=1070.

Additional SSE news items can be found on the Web site at: http://www.stevens.edu/sse/about/news/archive.php.



Book Review

The 21st-Century Engineer

A Proposal for Engineering Education Reform

By Patricia D. Galloway
Reston, Virgina: ASCE (American Society of Civil Engineers) Press, 2008
(ISBN-13: 9780784409367)

Reviewed by Erik W. Aslaksen, erik.aslaksen@incose.org

There is a movement for engineers to take more of a leadership role in business and public policy, and a particularly eloquent and interesting case is presented in this book by Dr. Patricia Galloway. She argues that the leadership position engineering enjoyed in the nineteenth century has been eroded in the twentieth century, and engineers are in the process of being viewed as technicians, as commodities. If engineers are to establish themselves as leaders in solving many of the world's most pressing problems and compete successfully in a global workplace, she argues, they need to broaden their skills beyond the traditional engineering subjects. The interesting part of the case is that, while her argument for the roles engineers should play and in which they could make substantial contributions is well put, as is the call for additional skills, the reasons why it has not been happening are less clearly explored.

In particular, she compares the high esteem in which the medical profession is held by society with that of engineering, failing to note that this comparison would in fact, were it carried out in more depth, undermine some of her assertions about where engineers need to change their view of the profession. The first of these relate to globalisation. Engineers, she says, need to be able to operate and compete successfully in an international market, taking account of the varying cultural and political circumstances; but is not engineering already more international than medicine? Engineers can practice freely in most countries and usually do work abroad for part of their career, whereas physicians are constrained by national licensing and competency requirements.

The second questionable assertion is about the need for greater understanding of and involvement in politics. According to Dr. Galloway, too few engineers move into high political office; but are there any more physicians in politics? Isn't the issue rather that the democratic system promotes an intellectual level in the elected leadership commensurate with that of the average of the population? China has a much higher proportion of engineers in the national leadership team, which has correctly perceived that internal stability is their main concern during this catch-up phase of the development of their society (as compared with the developed Western nations), well ahead of democracy.

The third assertion is that for engineering to be accorded the same status by the public as the medical profession, engineers need to broaden their skills in the direction of business, management, and people skills. But do physicians have greater skills in these areas? Physicians are much less involved in management and business than engineers.

No, the reason for the discrepancy in status between the two professions is to be found in a very different direction: in the employment structures of the two professions. The medical profession does not (at least in theory) allow an employment relationship between physicians, since this is perceived as enabling a conflict between professional and business interests. Most physicians work either alone or in partnerships, or then as employees of institutions in which the management should have no influence on the medical standards. In the engineering profession it is precisely because engineers are employed by other engineers that engineers are being turned into commodities, as Galloway rightly perceives is happening. In their roles as engineering and production managers in industry, engineers have been more than willing to turn their fellow engineers into obedient cogs in a production process in order to increase their outputs and profits. What is worse, universities and the engineering societies have been willing accomplices in this process, meeting the call from industry for large numbers of readily employable technical workers by lowering their professional standards. The fact is that at least in Australia a large proportion of today's engineers work as technicians simply because the education of competent technicians (as exemplified by the German title "Ingenieur HTL") has been abolished for political reasons. The institutes of technology were happy to be converted overnight into universities (why be a teacher when you can be a professor?), and the engineering society gained a lot of fee-paying members.

It is, in a way, ironic that the American Dr. Galloway should lament this development and call for a change, as it is primarily the Anglo-Saxon world that has been driving that development. In the English language, "engineer" was originally the driver of a steam locomotive, and there was always a fuzzy boundary between the tradesman and the professional. In Europe, the engineer was always well respected, and generally possessed academic qualifications second to no other profession; moreover, most of the captains of industry were engineers. As for adopting a global view, we can only hope that Galloway's plea will result in industrial firms in the United States finally adopting the international system of units.

Final Thoughts

From the Chief Editor

Bob Kenley, insight@incose.org

Kent Gladstone is a relatively new member of INCOSE who submitted an article in August 2007 for me to evaluate for inclusion in ***INSIGHT***. Kent had experience with the PMI *Guide to the Project Management Body of Knowledge* and the INCOSE *Systems Engineering Handbook*, and she had a desire to share her understanding of these two documents. After reviewing her submittal, I could see that Kent has some interesting insights, and that the article would be even better if she had access to someone who had more expertise than she with our handbook. I decided to forward the article to Cecilia Haskins, who is a long-time INCOSE member and the editor of the *Systems Engineering Handbook*. Cecilia collaborated with Kent through a couple rounds of editing, and the result is that Kent's article appears in this edition of ***INSIGHT***. I applaud Kent's willingness to take a risk and submit the article, and Cecilia's willingness to also take a risk and invest the time to further Kent's understanding as well as that of all of us. This type of interaction reflects the outcomes that we expect from participation in INCOSE via chapters, working groups, and other opportunities to interact with our colleagues. I do hope that the next "old hand" that I tap to assist a "greenhorn" with an ***INSIGHT*** article will follow Cecilia's example. I also hope that others like Kent will be willing to submit the articles that create such opportunities.

Please remember that we are shifting the publication of ***INSIGHT*** to allow us to publish what would been our first-quarter edition in January 2009 to one month earlier in December 2008. To accommodate this change, the submission deadline will be in October instead of November.

Upcoming submission deadlines and themes for ***INSIGHT***

Issue	Submission Date	Theme	Theme Editor
5th Qtr 2009	15 Oct 2008**	Space Systems: Navigating Complexity to Explore the Unknown	Jim Andary
1st Qtr 2009	15 Feb 2009	Cognition: Pursuing the Next Level in System Performance	Steve Deal
2nd Qtr 2009	15 May 2009	The Interplay of Architecture, Security, and Systems Engineering	Rick Dove
3rd Qtr 2009	13 Aug 2009*	East Meets West: The Human Dimension to Systems Engineering	Pearly Chua
4th Qtr 2009	15 Oct 2009**	Model-Based Systems Engineering: The New Paradigm	Rob Cloutier

* submission deadline moves according to International Symposium date
** new submission deadline

Just One More Thing

Andrew Cashner, andrew.cashner@incose.org

Before submitting an article to ***INSIGHT***, please remember to include complete bibliographic information for all your references. I am not concerned here with format or style: I understand that the *Chicago Manual of Style*'s rules for formatting citations may seem cumbersome and confusing. The problem is that pertinent information is often missing, such as authors' names, dates, titles, places of publication, publishers, or page numbers. Whether or not you manage to format your reference list in the correct style, please include all of the information listed below. If you cannot find some of this information, please try searching INCOSE's iPub database (https://www.incose.org/ipub), http://www.worldcat.org, http://scholar.google.com, or other scholarly or bibliographic databases. It will be much easier for you to locate this information than it will be for me. If an article is available electronically and you are not sure how to cite it, you are always free to e-mail it to me and I will determine how it should be cited.

Books

- First initial and last name of author or authors.
- Year of publication.
- Full title of book, including subtitle.
- Place of publication, including both city, state abbreviation (if in the United States), and country (if not in the United States).
- Full name of publisher, as it appears on the title page.

If you are citing just one chapter, please include the chapter title and the page numbers, along with the above.

Journal Articles

- First initial and last name of all authors, in the order they are listed in

the publication.
- Year of publication.
- Full title of article.
- Full title of journal (no abbreviations).
- Volume number.
- Issue number. (Month & day are not necessary except in the absence of other identifiers.)
- Page numbers.

Conference Proceedings and Papers Presented at Conferences

Please make clear whether or not the paper appeared in a published volume of conference proceedings. If it did, then you must cite it like the same way you would a chapter in a book. This means we need the official title of the proceedings, as listed on the title page, along with the place of publication and publisher. If these last two are omitted, use as the publisher the sponsoring organization (such as INCOSE), and as the place, the headquarters of that organization (in this case, Seattle, WA).

For our purposes, an officially-issued CD or DVD of the conference papers counts as a published volume only if it has running page numbers and a title page. Otherwise, it is too difficult to cite.

In any case, we need the full title of the conference (and of the sponsoring organization, if applicable), with no abbreviations.

If the paper appeared in published proceedings, the format of your citation will be like so:

Author, A. 2008. Title. In *Proceedings of the 1st Annual Conference on Conference Proceedings* (Connersville, IN), 12–30. Oconomowoc, WI: International Association for Conference Proceedings.

If there were no published proceedings, or if you are not sure how to cite the paper, this is the safest option:

Author, A. 2008. Title. Paper presented at the 1st Annual Conference on Conference Proceedings, Connersville, IN.

Web Sites and Other Informally-Published Materials

Please include as much information as possible to allow someone else to find the item you are citing. Please check to make sure that the URL still works, and that it would be accessible to someone without any special passwords or society memberships (other than INCOSE)!
- Author or creator of Web site (or organization responsible): usually shown at the bottom of the page.
- Year. Use the current year unless the site includes a notice like "Last updated 12 March 1986."
- Title of specific page you are citing, if possible.
- Description of what the site or page is (at the very least, an indication that it is a Web site).
- Complete URL.

For example:

INCOSE Communications Committee. 2006. Editorial Guidelines: *INSIGHT*. Web site. http://www.incose.org/ProductPubs/periodicals/editorialguidelines.aspx.

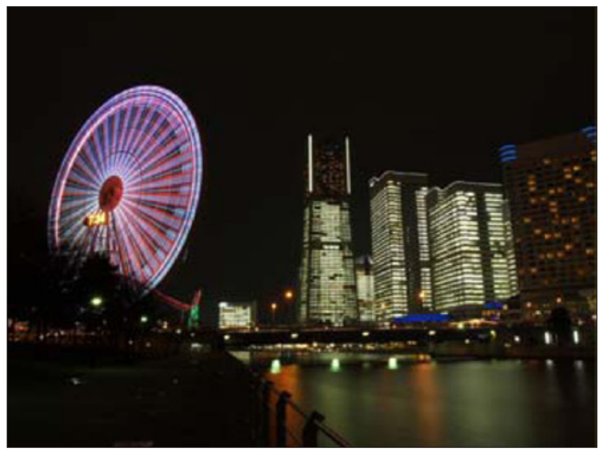
Or, in the case of a self-published paper or document posted online:

Cashner, A. 2008. Peerless brilliance: Dressler, Louis R. 1908. Alma Mater. Available at the Lawrence University Web site, http://www.lawrence.edu/about/trads/lu_alma_mater.pdf.

Systems Engineering: Wisdom for Global Harmony

The organizing committee of the second Asia-Pacific Conference on Systems Engineering (APCOSE 2008) is pleased to announce that the event will be held 22–23 September 2008 at the Yokohama Hiyoshi Campus of Keio University in Japan.

Hiyoshi campus



The city of Yokohama

This is the most important systems engineering event organized by the Region VI chapters of INCOSE. This event follows from the inaugural conference that was held in Singapore in March 2007. APCOSE 2008 is the very first conference organized by the INCOSE Japan Chapter (http://www.incose.orcl/japan) since it was approved by INCOSE as the 49th official chapter in March of 2007.

The conference chairman is Yoshiaki Ohkami, professor at Kelo University. The event is sponsored by JAXA, JAMSS, and MELCO, in collaboration with SICE, JSME, JSASS, and TEST *(under approval)*.

The deadline for submitting abstracts of papers to be presented at the conference is 30 April 2008. Authors of accepted papers will be notified on 30 June, and their final papers will be due 31 July 2008. For more information, please contact the conference's executive secretary, Hidekazu Nishimura of Keio University, apcose2008@sdm.keio.ac.jp.

The suggested keywords for APCOSE 2008 are given below. We encourage all researchers and engineers to take advantage of this opportunity to present their work and interact with others in the field.

international security / safety and security / safety and risk management / environment-conscious development / disaster mitigation / education and public outreach / systems engineering research in human factors and facilitation / systems engineering management / space systems design and operations / product line engineering / new product design and development / innovative use of information technology / multi-objective optimization / model-based systems engineering / embedded systems / architecting and integrating

Systems Engineering: The Journal of The International Council on Systems Engineering

Call for Papers

The *Systems Engineering* journal is intended to be a primary source of multidisciplinary information for the systems engineering and management of products and services, and processes of all types. Systems engineering activities involve the technologies and system management approaches needed for:

- definition of systems, including identification of user requirements and technological specifications;
- development of systems, including conceptual architectures, tradeoff of design concepts, configuration management during system development, integration of new systems with legacy systems, integrated product and process development; and
- deployment of systems, including operational test and evaluation, maintenance over an extended lifecycle, and re-engineering.

Systems Engineering is the archival journal of, and exists to serve the following objectives of, the International Council on Systems Engineering (INCOSE):

- To provide a focal point for dissemination of systems engineering knowledge
- To promote collaboration in systems engineering education and research
- To encourage and assure establishment of professional standards for integrity in the practice of systems engineering
- To improve the professional status of all those engaged in the practice of systems engineering
- To encourage governmental and industrial support for research and educational programs that will improve the systems engineering process and its practice

The journal supports these goals by providing a continuing, respected publication of peer-reviewed results from research and development in the area of systems engineering. Systems engineering is defined broadly in this context as an interdisciplinary approach and means to enable the realization of successful systems that are of high quality, cost-effective, and trustworthy in meeting customer requirements.

The *Systems Engineering* journal is dedicated to all aspects of the engineering of systems: technical, management, economic, and social. It focuses on the life cycle processes needed to create trustworthy and high quality systems. It will also emphasize the systems management efforts needed to define, develop, and deploy trustworthy and high quality processes for the production of systems. Within this, *Systems Engineering* is especially concerned with evaluation of the efficiency and effectiveness of systems management, technical direction, and integration of systems. *Systems Engineering* is also very concerned with the engineering of systems that support sustainable development. Modern systems, including both products and services, are often very knowledge intensive, and are found in both the public and private sectors. The journal emphasizes strategic and program management of these, and the information and knowledge base for knowledge principles, knowledge practices, and knowledge perspectives for the engineering of systems. Definitive case studies involving systems engineering practice are especially welcome.

The journal is a primary source of information for the systems engineering of products and services that are generally large in scale, scope, and complexity. *Systems Engineering* will be especially concerned with process- or product-line – related efforts needed to produce products that are trustworthy and of high quality, and that are cost effective in meeting user needs. A major component of this is system cost and operational effectiveness determination, and the development of processes that assure products that are cost effective. This requires the integration of a number of engineering disciplines necessary for the definition, development, and deployment of complex systems. It also requires attention to the lifecycle process used to produce systems, and the integration of systems, including legacy systems, at various architectural levels. In addition, appropriate systems management of information and knowledge across technologies, organizations, and environments is also needed to insure a sustainable world.

The journal will accept and review submissions in English from any author, in any global locality, whether or not the author is an INCOSE member. A body of international peers will review all submissions, with potential author revisions as recommended by reviewers, with the intent to achieve published papers that

- relate to the field of systems engineering;
- represent new, previously unpublished work;
- advance the state of knowledge of the field; and
- conform to a high standard of scholarly presentation.

Editorial selection of works for publication will be made based on content, without regard to the stature of the authors. Selections will include a wide variety of international works, recognizing and supporting the essential breadth and universality of the field. Final selection of papers for publication, and the form of publication, shall rest with the Editor.

Submission of quality papers for review is strongly encouraged. The review process is estimated to take three months, occasionally longer for hard copy manuscript. Five copies of your manuscript should be submitted for review purposes to

Professor Andrew P. Sage
Editor in Chief, Systems Engineering
School of Information Technology and Engineering
George Mason University
Fairfax, VA 22039-4444, USA
TEL: 703-993-1506
FAX: 703-993-1521
E-MAIL: asage@gmu.edu

Alternatively and preferably, electronic submission of manuscripts for review purposes is strongly encouraged, as this will speed up the review process considerably. Please send a copy of your complete manuscript in a single file with art and tables in approximately the proper location to asage@gmu.edu.

What's Inside

President's Corner 3

***INSIGHT* Special Feature**
The Best of France: Forum Académique et RobAFIS 5
Doctoral Symposium Papers 7
Education Symposium Papers 31

Fellows' Insight
Decomposed Requirements vs Derived Requirements 39

Forum
The INCOSE *Systems Engineering Handbook* and the *Project Management Body of Knowledge*: 40
A Universal Architecture Description Framework, Requirements Analysis, and Specification Preparation 42
Addressing Complex Systems 46

Technical Directions
From the Technical Director 49

Technical Activities
Standardized Approaches to Modeling Systems of Systems 50

INCOSE Operations
CSEP Beta Test Is Completed and the New Tests Are Ready to Go Live 51
From the Member Board Co-Chair 51

Chapter News
A Tutorial in Agile Systems and Enterprises 52
Colorado Chapter Participates in Career Fair 52

INCOSE Events
Report from the 2008 SEANET 53
Could You Say That Again, Please? 55

Industry News
Future Acquisition Directorate Team Leading the Way In System-of-systems Engineering in the U.S. Defense Department 60

Book Review
The 21st-Century Engineer 61

Final Thoughts
From the Chief Editor 62
Just One More Thing 62

INSIGHT
International Council on Systems Engineering
2150 N. 107th St., Suite 205
Seattle, WA 98133-9009

Presort Std
U.S. Postage
PAID
Seattle, WA
Permit #4