Regular Paper

COIM: An Object-Process Based Method for Analyzing Architectures of Complex, Interconnected, Large-Scale Socio-Technical Systems

Carlos A. Osorio,[1,*] Dov Dori,[2] and Joseph Sussman[3]

[1]*Business School, Universidad Adolfo Ibáñez, Diagonal Las Torres 2700, Santiago, Chile*
[2]*Faculty of Industrial Engineering and Management, Technion-Israel Institute of Technology, Haifa 32000, Israel*
[3]*Massachusetts Institute of Technology, 79 Massachusetts Avenue, Cambridge, MA 02138*

Received 11 October 2009; Revised 21 November 2010; Accepted 21 November 2010, after one or more revisions
Published online in Wiley Online Library (wileyonlinelibrary.com).
DOI 10.1002/sys.20185

ABSTRACT

There is growing evidence for the relevance of human behavioral factors in a successful a development of new products, processes, and services. The evidence is even clearer when the forces affecting the development and evolution of long-lived, large, and open complex socio-technical systems are examined. Methods that study the architecture of these systems can help scholars and practitioners to better understand, manage, and develop socio-technical systems. We propose an approach and a method to address these needs that is grounded in the theory of systems architecture and builds on the strengths of Object Process Methodology (OPM) and the process for representing Complex Large-scale Interconnected Open Socio-technical (CLIOS) systems. We do so by integrating these methods into the CLIOS-OPM Integrated Method (COIM). COIM is conducive to studying a system's architecture and its evolution, as it is enhanced by a set of qualitative methods for answering questions about the reasons (*why*) and process (*how*) of change in human-made systems over time. © 2011 Wiley Periodicals, Inc. Syst Eng 14

Key words: system architecture; evolution; legacy; Object Process Methodology (OPM); complex large-scale interconnected open socio-technical (CLIOS) system

1. INTRODUCTION AND RESEARCH OBJECTIVES

This paper is motivated by an effort to study and model the forces affecting the development and evolution of long-lived, large, and open complex socio-technical systems, building on research outcomes of scholars who have effectively worked in this area.

Osorio [2007] focused on studying the architectural evolution of a specific type of long-lived socio-technical system, namely, municipal electric utilities (MEUs). In doing so, he encountered limitations on the theory and available methods and processes for answering his research questions, as there was no single satisfactory method for studying the evolution of the architecture of MEUs and explaining their diversification into broadband communication services. Since this evolution was contrary to predictions made by scholars and

*Author to whom all correspondence should be addressed (carlos.osorio@uai.cl; dori@mit.edu; sussman@mit.edu).

Contract grant sponsors: MIT Communications Futures Program (CFP), its industry sponsors, and NSF Grant #EIA-0306723.

practitioners as to how and why this would happen, a new way to study and understand this evolution became necessary.

Beyond the particular interest in understanding the phenomena underlying the evolution of MEUs into broadband providers, there is a more general need to understand the underlying factors affecting the architectural evolution of complex, interconnected socio-technical systems. This need is highlighted when we study long-lived, large, and complex socio-technical systems that are open to the influences of social actors or have significant impact on society. Some examples are networks of critical infrastructure, such as transportation, electric power, and telecommunication systems, aircrafts and spacecrafts, such as the B52 and the International Space Station, and software systems, such as SAP Enterprise Resource Planning or Chile's nationwide Government Procurement Online System.

The original architecture, design, and implementation of this type of system are normally carried out through careful study and documentation of the functional, structural, and procedural aspects of the system under development. However, throughout the lifetime of such systems, humans change them in attempts to enhance their performance, maintain some or all of their components or subsystems, update some of their underlying technologies, and adjust their structure or functions to the changing needs of varying environments. A recurrent problem with these changes is that they are rarely documented, and even if they are, the documentation is often inaccessible to the numerous architects and designers involved in such changes over timeframes that can last many decades and often transcend the work lifespan of the involved individuals.

Improving this state of affairs requires adequate tools for studying and representing the architecture of this type of systems. Building on prior research, this paper proposes COIM—CLIOS-OPM Integrated Method—as an integrated approach, method, and notation for representing and studying architectures of complex, large-scale interconnected socio-technical systems and their evolution over time.

While no single approach can serve all purposes, a combination of methods under a guiding theory could facilitate the study of the architecture of socio-technical systems. We base our analysis on the study of system architecture, which provides a useful overarching framework for understanding the various factors influencing the overall form, function, and concept of a system's architecture.

We review the literature on systems architecture and system representation, with special focus on methods, strategies, and processes of representing the structure and behavior of systems. Based on the guiding study of system architecture, COIM is an integrated methodology that builds on the representation stage of the Complex Large-scale Interconnected Open Socio-technical (CLIOS) Process and Object Process Methodology (OPM).

When appropriate and necessary, we illustrate COIM with examples from our previous research on the diversification of municipal electric utilities into broadband services, intelligent transportation systems, and others.

2. LITERATURE REVIEW AND CONTRIBUTIONS TO THEORY

System architecting is a developing field of study [Whitney et al., 2004]. One question that has remained open is how to study the architecture of large-scale socio-economic systems and their evolution. Necessary steps in our research are to understand (i) what do we mean by system architecture, (ii) what evolution a system's architecture can undergo, and (iii) what modeling methods might be useful and effectively combined for representing the architecture and its evolution.

2.1. What Is System Architecture?

Many disciplines in engineering and management sciences provide their own definitions for system architecture. Computer science has been an important source of contributions to the study of the architecture of complex systems. Blaaw [1997zaq;1] defined computer architecture as "the minimal set of properties that determine what programs will run and what results they will produce." It is thus concerned with the "functional appearance" of the computer, or "what should it do." In Blaaw's view, the structure is not part of the architecture; but rather corresponds to different domains of computer *design*: implementation (logical structure) and realization (physical structure). In this perspective, the nature of computer architecture is no different from that of language or software architecture, ranging from microcode to application. Consequently, Blaaw defined computer architecture as the outcome of the design of a "programming language when expressions are costly.zaq;1"

Also from the field of computer science, Black [1989] presented a view for analyzing network architectures and protocols. For Black, network architecture is the definition of "what things exist" in a network, "how they operate" (protocols) and "what form they take" (topology).zaq;1 A common feature of the different network architectures analyzed by Black is the decoupling [Suh, 1998] or orthogonality [Blaaw, 1997] between major functions arranged in "layers." These layers provide (i) a decomposition into logical subsystems, (ii) standard interfaces among them, (iii) functional symmetry across nodes (or peer elements), and (iv) command and control.

In layered architecture, each layer operates independently and interacts with others via a set of protocols. This idea underlies each of the network architectures that Black [1989] studied in his work: (i) Open Systems Interconnection (OSI), (ii) U.S. Government Open Systems Interconnection Profile (GOSIP), (iii) Systems Network Architecture (SNA), (iv) Digital Network Architecture (DECnet), and (v) the Defense Data Network (DDN). Interestingly, all these networks presented the same underlying concept of layered architecture described by Black.

Hans van Vliet [2001] presented the role of architecture in software development by identifying and characterizing different architectural *styles* and different *forces* affecting architecture. His focus was on how to identify and describe different software architectures: shared data, abstract data type, implicit evocation, and pipe-and-filter. He proposed that software architecture has three major objectives: (i) communicating among stakeholders, (ii) capturing early design decisions (legacy and compatibility with early versions), and (iii) transferring and reusing the system as the basis for a *product family* [Meyer and Utterback, 1993] by providing access to common code.

In the context of information systems, the Open Group Architectural Framework [TOG, 2001] has defined architecture as "a set of elements (sometimes called *building blocks*) depicted in an architectural model, and a specification of how these elements are connected to meet the overall requirements of an information system."zaq;1 This definition requires that connections among elements, which are a structural aspect of the architecture, be specified *in a model*.

From the literature of product development, Ulrich [1995] defined architecture as the *scheme by which function is allocated to physical components*, and argued for its importance in manufacturing. He stated that architecture is a *key driver of performance* and managerial decision-making. More importantly, however, he argued that architecture is specifically relevant to product change, variety, and performance, component standardization and managing the product development process. Ulrich argued that architecture is central to the change of products along their lifetime, across generations, and within a company's variety of products. He suggested that product variety results from flexibility in architecture rather than from core capabilities (e.g., a factory's equipment). Ulrich proposed product variety as a function of the flexibility in product architecture, component processes, and standardization.

Finally, similarly to Ulrich [1995], Crawley and Weigel [2004] proposed that the "architecture" of a technical system is defined as the way in which its concept matches its form to its function. Recent work stating various architectural "views" provides additional insights into this area [Rhodes, Ross, and Nightingale, 2009].

We can find parallels among many of the works discussed in this section by defining architecture in terms of *form*, *function*, and *concept* [Crawley and Weigel, 2004]. From the perspective of Black [1989], the protocols, topology and "what things exist" can correspond to function, form, and concept. We can also draw analogy between these ideas and the terms of Ulrich [1995], *scheme* (concept), *function* (function), and *physical components* (form).

Dori [2002: 263] defined system architecture as follows: "System architecture is the overall system's structure-behavior combination, which enables it to attain its function while embodying the architect's concept." This definition combines the elements of concept—how the architect envisions the solution of the problem of combining structural elements and their behavior in a way that enables the system to function as expected, thereby providing value to its beneficiaries.

Table I summarizes the various definitions of system architecture from the surveyed authors along with the relationships between form, function, and concept, and, in some cases, their relation to behavior.

If we define "form" or "physical components" as "structure-behavior combination," then the definitions of Black [1989], Ulrich [1995], Dori [2002], The Open Group [TOG, 2001], and Crawley and Weigel [2004] coincide, expressing the same idea, which we state as follows:

A system's architecture is the embodiment of a concept for achieving the desired system's function in terms of its form, i.e., structure-behavior combination.

This definition is valid for the architecture of any human-made system, from architecture of instruments and buildings, through software systems, to complex socio-economic engineering systems.

The general idea is illustrated in the tree in Figure 1: Each architecture is the distinct combination of Function-Concept-Form nodes that is a path in the tree. Accordingly, Figure 1 depicts at least three distinct architectures: (1) Function—Concept 2—Form 1, (2) Function—Concept 2—Form 2, and (3) Function—Concept 2—Form 3.

Our working definition of a system's architecture, as presented in Table I, is the way in which a *concept* maps the system's *form* (structure-behavior combination) onto its *function*. A first step in the analysis of the architecture of a socio-technical system is to identify the dominant influences that affect these three major dimensions of the architecture at the highest hierarchical level.

A useful distinction can be made between intended and emergent architecture. Intended architecture is a result of an orderly architecting process, which accounts for requirements and weighs in alternatives before committing to a specific architecture—the intended one. Emergent architecture results from natural or social evolution over long periods of time. Examples of intended architectures are those of a jet aircraft or a multicore processor. Examples of emergent architectures are those of a living organism [Jacob, 1977] or a city.

The theory of system architecture provides a guiding framework for the analysis of the form (structure-behavior combination), function, and overall concept of a socio-technical system and its subsystems, as well as the sources of dominant upstream and downstream influences.

Dominant Upstream Influences (DUIs) include the regulatory and legal environment that affects form and function, corporate and marketing strategy, the influences imposed by customers and beneficiaries through their stated needs, the effects of the competitive environment, the evolution and availability of technology, and other strategies and internal competencies. Dominant Downstream Influences (DDIs) include those arising from the design, implementation, operation, and evolution of the system.

A system's architecture can be affected by various factors not only during its design, construction, and first implementation, but also throughout its entire operational lifetime via changes in its form, function, or concept. In the case of

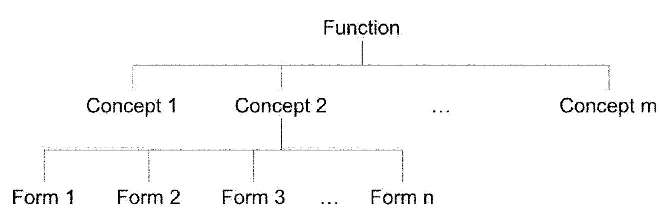


Figure 1. Different architectures of the same system, fulfilling the same function.

Table I. Summary of System Architecture Definitionszaq;1

Definition of System Architecture	Function defined in terms of	Concept defined in terms of	Form defined in terms of	Source
"the minimal set of properties that determine what programs will run and what results they will produce"	Architecture as the functional appearance of the computer	Not explicit	Structure is not part of the architecture, but rather falls into a design domain	[Blaaw, 1997]
"what things exist" in a network, "how they operate" and "what form they take"	The effect created by network components, topology and protocols, decoupled by layers	Network topology,	Network components are the basis of form in a network, and protocols drive their behavior	[Black, 1989]
(i) a vehicle for communication among stakeholders, (ii) capturing early design decisions, and (iii) providing a basis for transferring and reusing parts of the system.	Overall desired performance of a software	Understood as different architecture styles	Code, which is affected by external forces	[van Vliet, 2001]
"a set of elements..depicted in an architectural model, and a specification of how these elements are connected to meet the overall requirements of an information system"	"Overall requirements of an information system"	Model of an information system	Elements and interfaces among them in a model	[TOG, 2001]
"scheme by which function is allocated to physical components"	The function of a product	Scheme	Physical components	[K. T. Ulrich, 1995]
"the overall system's structure-behavior combination, which enables it to attain its function while embodying the architect's concept"	Overall system's function	A particular architecting concept	Structure-behavior	[Dori, 2002]
"conceptually design, evaluate and select a preferred structure for a future state enterprise to realize its value proposition and desired behaviors"	The enterprise's value proposition and desired behavior	A particular concept of design, evaluation and selection	The preferred structure of a future state enterprise	[Rhodes, Ross and Nightingale, 2009]
Proposed Definition: A system's architecture is the embodiment of a concept for achieving the desired system's function in terms of its form, i.e., its structure-behavior combination.	A desired system function	A particular architecting concept	Form as a structure-behavior combination	This paper

long-lived systems, the architectural evolution during its operational life is highly relevant for our research. For example, we wish to understand how the architecture of a MEU of interest has evolved, and how the legacy aspects of its architecture have been playing a role in shaping its current architecture.

Theory of system architecture cannot yet answer questions of this type, because (i) its theory is still not yet sufficiently developed across all fields, (ii) system architecture provides guidelines for system decomposition, but it does not offer methods for representing CLIOS-like systems, and (iii) system architecture does not satisfactorily address nontechnical issues that greatly impact socio-technical systems. To gain more insight into what is still missing, in the next section we discuss the evolution of a system's architecture and methods for representing system architecture.

2.2. Evolution of System Architecture

While there are solid foundations in systems engineering, system architecture is in its early stages of development as an overarching theory across engineering disciplines. However, it can provide a useful framework for analyzing complex socio-technical systems and studying their evolution. Using this framework requires understanding of a given system's building blocks and its representation, how its architecture evolves, and to employ useful ways to represent the system.

Herbert Simon [1997] presented four principles for complex systems design useful for our purpose: specialization, homeostasis, membranes, and near-decomposability.[1] Specialization refers to the diversification of complex functions and helps a system to focus on its externally delivered function and its subsystems to perform particular tasks. Homeostasis helps keep the system under control in the face of internal or environmental changes via feedback, or "balancing loops" in system dynamics jargon [Sterman, 2000]. Membranes help insulating the system and its subsystems and allowing interactions among them. Near-decomposability is the basis a stable system's separation into subsystems. These principles are different facets of the flexibility of complex systems, enabling them to adapt and perform new functions. This is also aligned with the concept of autopoiesis, a system's abilities for self-creation [Maturana and Varela 1980], and Winograd and Flores' [1990] perspective on cybernetics, to explain how systems can adapt and evolve. This architectural flexibility enables system evolution.

From the perspective of van Vliet [2001], the evolution of architecture is affected by the organization actually "creating" the system, and the cyclical relationship between the system and its environment. Van Vliet focuses on the architecture of software, in contrast to the approach of Blaaw [1997] to computer (hardware) architecture.

All the cases of computer architecture discussed by Blaaw [1997] occurred prior to 1985. Once computer and software architecture shifted towards the personal computer, and the operating system (OS) was separated from the computer, his vision of architectural evolution ceased to be applicable. Computer and OS architectures are currently distinctive, and their evolution is marked by different, yet interrelated, clock-speeds [Fine, 1998]. Computer architecture evolves about every other year, characterized by increasing memory and processing capacity, while software architecture evolves continually through updates. To a great extent, this gradual software evolution is due to the constant feedback between (i) changes of user demands, (ii) security concerns and problems, (iii) detection of bugs, (iv) conflicts between new and old subroutines, and (v) corrections and new design features created by development teams that are updated and distributed over the Internet-through patches and updates.

This development in the information technology arena has relevant implications for technology management, in cases where a second line of business emerges from internal functions of a socio-technical system. Studying the relationships between form and functions of components of the infrastructure supporting both hardware and software can shed light on such evolutions [Osorio, 2007].

A key concept in this context is the system's Externally Delivered Function (EDF), which is the system's function at its highest hierarchical level [Crawley and Weigel, 2004]. Alternatively, in Object-Process methodology (OPM) [Dori, 2002] terminology, EDF is the system's (only) function, whereas internal functions are called processes. We shall henceforth refer to EDF as "function." The system's function is understood by decomposing and disaggregating it into several processes (*internal* functions). These are usually mutually exclusive, but comprehensively exhaustive [Pimmler and Eppinger, 1994]. Thus, studying the evolution of architectural system includes three possible cases: (i) a change in the actual function, (ii) the emergence of a new function, and (iii) internal process changes with no change to the existing function.

The first case, change in the system's actual function, pertains to the basic form of architectural evolution. It is related to changes in processes and associated form designed to enhance the original function of the system—the reason for developing it in the first place. Sometimes, however, intended architectural evolution of a system can result in severe hindrance of its performance, as was the case with the change from the old to the new Public Transportation System of the city of Santiago, Chile, also known as Transantiago [Mardones, 2008; Pelayo, 2007].

In the second case of architectural evolution, emergence of a new function, the deployment of a new function is a consequence of the many external and internal influences affecting the system [Crawley and Weigel**,** 2004; Osorio**,** 2007; Whitney et al.**,** 2004]. Investigators need to focus on the engineering system as a technical system embedded in a defined policy, social, and economic context [Dodder et al., 2005; Mostashari and Sussman, 2009]. In the policy context, several actors can affect the form, function, or concept of the system by various regulatory means. In the social and economic contexts, residents and businesses exhibit certain needs and demand certain services. The combined effect of changes of the various types adds uncertainty to the system, creating new design spaces [MacCormack, 2005; MacCormack and Verganti, 2003], many of which are not observable to the managers of the system [Osorio, 2007] due to bounded rationality [Simon**,** 1991].

The effect of regulations and of social and economic needs on decision-making is central to the evolution of an organization, accounting for part of the third case of architectural evolution—internal process changes with no change to the existing function. Analysis of these regulatory, social, and economic factors can help explain why architectures of long-lived systems evolve in certain ways. These systems are of special interest, because, in most cases, their evolution is partially unintended—it has not been planned by a single architect throughout the system's life. Rather, their architecture has resulted from a series of incremental unplanned, or planned, changes by different actors. In these cases, possible paths for evolution are constrained by the legacy of the preexisting architecture, and limited by each architect's bounded rationality and insufficient information about the evolutionary possibilities.

The open question we face is how to represent system architecture in a way that would best cater to studying its evolution.

[1]The concept of *Membranes* is the basis of a work by Pimmler and Eppinger [1994], proposing strategies for product architecture and decomposition. *Specialization* has been used in methods such as Axiomatic Design [Suh, 1998], decomposition strategies for products [Koopman, 1995], and decomposition of the design process [Eppinger et al., 1994].

2.3. Decomposition and Representation of System Architecture

A useful representation of a system's architecture has the following features: (i) It results from a strategy for decomposing, relating, and representing its underlying structure (objects), behavior (processes), and design concepts, and (ii) it expresses function—the high-level goal or purpose of the system, and the main reasons and objectives for the particular system's architecture.

Koopman [1995] proposed a framework for decomposing and representing system architecture based on structures (form), behaviors (processes or internal functions), and goals ("desired emergent properties" or needs), creating a taxonomy that allows for comparison across different strategies. While this framework considers both technical and non-technical criteria, it is explicit in *not* considering technical, regulatory, and political or business influences.

Koopman's framework is based on decomposing the design according to the three previous dimensions, which range from "pure," when performed based on only one of them, "split," when separating dimensions into decoupled subdesigns and using pure decomposition in each, and "combined," which considers two or all three dimensions at the same time. The approach departs from ad hoc decomposition by allowing options for greater modularity. It does not, however, differentiate between the *design process* and the resulting artifact—the *final design* [Koopman, 1995]. This missing distinction between processes and objects is a major tenet of OPM. From the perspective of our research this is a major limitation, and we discuss it in the conclusions section.

Suh [1998] presented Axiomatic Design as a method for designing systems in terms of minimizing the complexity arising from the interaction between functional requirements (FRs), design parameters (DPs), and process variables (PVs) for systems of fixed functional requirements. Koopman [1995] and Pimmler and Eppinger [1994] followed a similar objective.

Axiomatic Design (AD) is based on two axioms: independence among functions (Independence Axiom) and achieving the least possible information content on the design (Information Axiom). The Independence Axiom is equivalent to the *Specialization* of Simon [1997]. The method consists of the following steps: (i) defining the functional requirements (FRs) for the system, which requires finding the customers' attributes and needs, (ii) mapping FRs with physical elements in order to create design parameters (DPs) and identify process variables (PVs), (iii) testing the Independence Axiom between functions, and (iv) verifying the information content of the system. AD is also based on the hierarchies among FRs, DPs, and PVs. Hierarchy provides policy and supervisory functions that start at the highest system level and progress to lower levels of detail, tracking up and down from the beginning point, in a process called zigzagging, which is important for the design decisions between FRs and DPs.

Pimmler and Eppinger [1994] presented a method for analyzing product design decompositions and for understanding and evaluating the requirements of system engineering. The method follows a three-step process of (i) decomposing the system into elements, (ii) documenting their interactions in terms of proximity, energy transfer, information, or material interchange, and (iii) clustering them into "chunks." Interactions range from those that must be prevented to achieve functionality to those necessary for functionality. Clustering is achieved by reordering elements around a matrix diagonal in order to reduce interaction complexity. While presenting decomposition strategies, Pimmler and Eppinger do not stipulate a representation method or process.

Regarding system architecture limitations on representation, one might consider using Unified Modeling Language (UML) [Booch, Rumbaugh, and Jacobson, 2005] or the more recent System modeling Language (SysML) [Weilkiens, 2007] as a possible basis for the task at hand. However, UML is specifically targeted towards the design of components of object-oriented software systems rather than socio-technical systems. Moreover, UML 2 has 13 different types of diagrams, some of which are used for structure modeling while others—for behavior modeling. No single diagram models both aspects in tandem. The use of multiple diagrams brings up the model multiplicity problem [Peleg and Dori, 2000]. While SysML is geared towards systems in general, it too suffers from the same model multiplicity problem. The problems we face go beyond representing a system's components and functions and solving the multiplicity problem, as they also include social, nontechnical components and aspects affecting the system.

There is, however, a methodology that includes a language and a modeling approach that draws on the theory of systems architecture to enable representation and study of systems. There is also a process for analyzing socio-technical systems. Together, they provide for representing the architecture of socio-technical systems and studying their evolution. The method, Object-Process Methodology (OPM), developed by Dori [2002], has been used for conceptual modeling and system architecting in many domains, notably in product development, and specifically by Crawley and Weigel [2004], to represent a system's architecture based on form, functions, and concept. The process of analyzing socio-technical systems was developed by Sussman's research group at MIT [Mostashari and Sussman, 2009] for analyzing Complex, Large Interconnected Open Socio-Technical (CLIOS) systems. OPM and CLIOS are explained next.

2.3.1. Object Process Methodology

Object Process Methodology (OPM) offers a consistent method for the study of the architecture of a system. It provides operators for hierarchical decomposition, and helps identify the underlying processes that link function and form. The result is a single graphical and an equivalent textual model combining the structure and behavior of the system under scrutiny at varying levels of detail. OPM has been used primarily in product design and systems engineering. Recently, it has also been employed for modeling biological systems at the molecular level [Dori and Choder, 2007]. In OPM, a system is defined as an object that exhibits a function, where the function is "the main intent for which [the system] was built, the purpose for which it exists, [and] the goal it serves" [Dori, 2002: 251]. With respect to our study, OPM is limited in that it does not specifically recognize the relevance of social, organizational, and contextual dimensions; nor does it provide a way to explicitly include them in the analysis.

The major features of OPM allow for hierarchical decomposition of the system into objects, or physical elements, and processes—internal functions—in a well-defined hierarchical manner at various levels of refinement. This is done by expressing relationships between objects and processes via structural and procedural links and via refinement-abstraction mechanisms. Agents, in OPM terms, are human operators—individuals that make the system work. Operands are the most important objects on which processes operate in order to carry out the system's externally delivered function.

The OPM language employs symbols to represent the elements (form) and processes (functions) of a system's architecture [Dori, 2002]. OPM includes notation for representing states (e.g., an alarm can be on or off), and can also be used to study a system's life cycle and evolution [Dori, 2002: 289]. OPM is bimodal, expressing the system model in both graphics and text—a subset of English, called Object-Process Language (OPL), that is generated automatically on the fly if one uses OPCAT [Dori, Reinhartz-Berger, and Sturm, 2003] for OPM-based modeling. The details of symbols of OPM entities (objects, states, and processes), as well as of structural and procedural links and corresponding OPL sentences can be found in Dori and Choder [2007].

We borrow from system architecture the focus on form, function, and concept, along with identification of needs, intent, processes, and objects. We focus on the representation approach of OPM, which enables us to understand the extent to which the system meets the needs of beneficiaries through its function. An effective way to achieve this is through hierarchical decomposition. OPM naturally represents the decomposition of the system's function (processes) and form (objects, or components) in a top-down fashion, from the most general to the most detailed abstraction levels.

The OPM modeling process starts with the examination of the way in which (i) the *externally delivered function* (EDF, or simply function) is associated with the needs of beneficiaries from using the technical system, (ii) the intent to fulfill their needs, (iii) the operands and value attributes associated with the function, (iv) operators of the system, and (v) first-level decomposition of the technical system among functions and objects in at least one of the concepts for architecting it.

Form decomposition via aggregation-participation (whole-part) relation includes links only among units of form. Likewise, functional decomposition using the same relation includes links only among processes—functional elements. For any given level of abstraction, the form and function decompositions can be shown in tandem in a single Object-Process Diagram (OPD) at some level of detail. Procedural links in the OPD connect objects to processes, expressing the dynamic aspect of the system. This is normally done up to the third or fourth hierarchical level, which is possibly relevant for system design and development but not necessarily for our research. Depending on the architecting task at hand, it is necessary to find the "adequate" level of disaggregation in form and function that allows understanding of the underlying architecture or its evolution. The decision about the level of decomposition is based on the complexity of the system and on the analysis goals.

From this perspective, OPM has many characteristics one might want to use for studying a system's architecture. However, for modeling socio-technical systems, OPM has two limitations:

i. It does not consider the broader social, organizational, or institutional context under which the system operates, is designed, or is developed. While all these can be modeled as objects and processes, OPM does not have specific means to relate to these entities as elements that need special treatment.
ii. OPM does not explicitly consider social dimensions affecting form, function, or concept beyond the interactions of the system with its operators and consumers. Social dimensions include critical influences on a system's architecture [Crawley and Weigel, 2004; DeSanctis and Poole, 1994]; its organization [W.J. Orlikowski, 2000], especially if it is public [Fountain, 2001]; and the overall strategy that governs it [Arcelus and Schaefer, 1982].

To complement the strengths of OPM with a method that could offset its limitations, we have adopted the Complex, Large Interconnected Open Socio-Technical (CLIOS) Process.

2.4. The CLIOS—Complex, Large Interconnected Open Socio-Technical—Process

CLIOS is a process for analyzing Complex, Large Interconnected Open Socio-Technical systems in an iterative manner. It consists of three stages: (i) representation, (ii) design, evaluation, and selection of strategic alternatives, and (iii) implementation of the chosen alternatives. The details on the CLIOS Process can be found in Mostashari and Sussman [2009].

In this research, we focus on the Representation Stage, the objective of which is to "convey the structural relationships between the components of the CLIOS system" [Dodder et al., 2006a]. CLIOS Systems are composed of a complex Physical Domain (PD) that is nested into another complex Institutional Sphere (IS), as represented in Figure 2. The

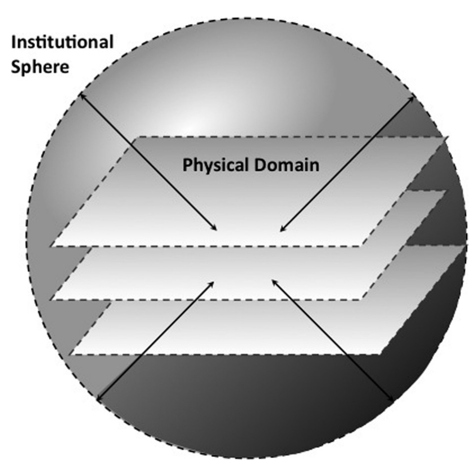


Figure 2. Illustration of nested complexity. (Source: Dodder et al. [2005].)

institutional sphere is formed by formal and informal organizational actors that interact with the Physical Domain. The interactions between the PD and IS create "Nested Complexity" [Dodder et al., 2006a]. From the perspective of organizational theory, the concept of Nested Complexity could explain some of the dynamics among the technical and organizational dimensions of systems, as well as problems in their performance and integration. The CLIOS Process approach to studying Nested Complexity through the representation of the PD (Physical Domain) and IS (Institutional Sphere) makes CLIOS especially fit as one of the bases of our research.

The major strengths of the representation stage of the CLIOS Process are the following:

i. The CLIOS representation is explicit in considering the physical and institutional domains of socio-technical systems, allowing for analysis of the relationships and possible behavioral interactions among them. The definition of the institutional sphere is a major contribution to studying the effect of external institution-based influences in architecture. Using this representation, one can analyze how the various dominant influences on system architecture affect form, concept, and function of a given socio-technical system.
ii. The physical domain in the CLIOS representation considers not only the technical subsystem of interest—for instance, the electric power infrastructure of a municipal electric utility (MEU)—but also other subsystems that explain the broader context in which a technology is embedded, e.g., the economic activity and municipal subsystems. These subsystems can have important effects on the evolution of the architecture of the technical components of a socio-technical system.
iii. The CLIOS representation allows for identifying various types of components and links, and includes the category of "common drivers." These common drivers identify relationships among physical components and between them, as well as organizations in the institutional sphere.

The Representation Stage of the CLIOS Process provides an additional framework for the analysis and representation of a socio-technical system. This distinction complements the analysis of dominant influences in a system's architecture. Common drivers help identify objectives and elements that can affect system evolution by being common to two or more subsystems. The CLIOS Process, however, has two limitations: (i) It does not provide a modeling framework for representing hierarchical relationships among elements, be they elements of form (objects) or function (processes), and (ii) it does not have operators that represent functions or processes as opposed to form. Since functions are one of the three main components of a system's architecture that need to be represented, this lack is critical.

Considering the issues discussed in previous sections and the needs of our research, we provide for representing functions and hierarchical decomposition. We need to identify functions and describe the relationships between existing and new functions of a system. The ability to represent hierarchical relations among functions is important, as it allows us to identify new functions added by design, new emergent functions, which appear unexpectedly without being explicitly designed, and changes in an existent function. This limitation of the CLIOS process can be solved by adopting ideas and notations from OPM. To overcome these limitations, we have created a derivative method—the CLIOSP-OPM Integrated Method (COIM). COIM, presented next, builds on the strengths of OPM and the CLIOS process while offsetting their limitations.

3. AN INTEGRATED METHOD FOR REPRESENTING THE ARCHITECTURE OF A SOCIO-TECHNICAL SYSTEM

The previous sections presented the literature of system architecture, as well as the benefits and limitations of OPM and the representation stage of the CLIOS process for studying the architecture of complex socio-technical system and its evolution. Studying the architecture of this type of system requires a combined approach. This section discusses how CLIOS and OPM are combined into the CLIOS-OPM Integrated Method (COIM), a robust analytical framework that offsets the limitations of each one of its components when used separately (see Table II).

With systems architecture providing the theoretical framework, the combination of the CLIOS Process and OPM is the basis for our analytical framework. The simplicity of the CLIOS Process for studying large-scale socio-technical systems is enhanced with the rigor and orientation to details of OPM. An important outcome of this OPM-CLIOS Process integration is that the analysis of upstream influences on system architecture becomes explicit in the system representation process through the inclusion of the first stages of the CLIOS Process under a System Architecture framework. Upstream influences include regulation, the organization's strategy, needs and goals of customers and beneficiaries, competitive environment, and technologies. We continue with a specification of COIM.

3.1. COIM Terminology and Representation Operators

An important aspect of a system's representation is the extent to which it can provide insight into that system's behavior [Dodder et al., 2005; Dori, 2002; Sterman, 2000]. This is particularly relevant when we wish to understand the reasons for the evolution of a system's architecture over long periods of time, and where numerous architects and other agents and factors have been making intended and unintended changes that contributed to the current architecture. To this end, we have adopted a distinction within the CLIOS Process between the physical domain and its institutional sphere. We have also adopted OPM's distinction between elements of form—objects and those of function—processes. The analysis is thus inspired by both the CLIOS Process and OPM.

The COIM representation operators have evolved from their original counterparts in the CLIOS Process, adding the differentiation between elements, or components, of form and

Table II. Summary on Focus, Strength, and Limitations of Approaches

Method/Process	Focus	Main Strength	Limitations
System Architecture (SA)	Study of System Architecture	Conceptual frameworks, analysis of dominant influences	It needs a representation method for socio-technical systems
Complex Large Interconnected Open Socio-Technical System (CLIOS) Process	Analysis of Socio-Technical Systems	Physical domain, institutional sphere, social/organizational dimensions	No differentiation of elements and functions. No hierarchical decomposition
Object Process Methodology (OPM)	Product Design and Development	Elements and functions, allows hierarchical representation	No explicit consideration of institutional or social dimensions

Source: Osorio [2007].

function. In Figure 3, these components are depicted and exemplified using examples from the electric power industry.

- *Components of form***,** indicated by a solid-line oval, are defined as the physical elements that comprise a system. Some examples from the electric utility domain are transformers and transmission and distribution.
- *Components of function***,** represented by a dotted-line oval**,** characterize the purposes and goals of the physical elements of a system, or, equivalently, represent the functions associated with elements of form. For a distribution line, for instance, the associated function would be distribution of electric power from the distribution transformer to the customer premises.
- *Policy levers*, indicated by a white rectangle, are "components within the physical domain that are most directly controlled or influenced by decisions of actors….on the institutional sphere" [Dodder et al., 2006bzaq;1]. Thus, policy levers are a way by which institutional actors can affect form or function components of a subsystem in the physical domain (see Fig. 4, case 3, for example).
- *Common drivers*, represented by a diamond, are "components that are shared across multiple and possibly all subsystems of the physical domain" [Dodder et al., 2006azaq;1]. Common drivers are important influences on architecture, especially due to their influence across subsystems of the physical domain.
- *External factors***,** represented by a shaded rectangle, are exogenous parameters that affect the system but are not affected by the system.

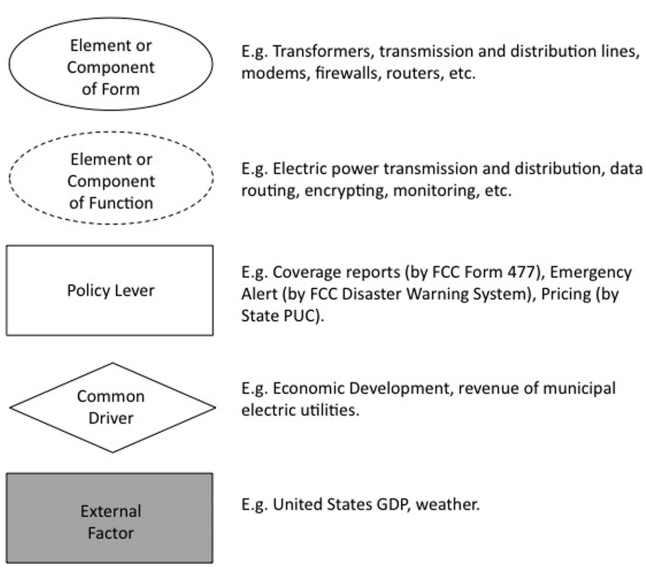


Figure 3. COIM components. (Source: Osorio [2007].)

Based on the CLIOS Process and OPM, COIM also includes the following set of links, shown and exemplified in Figure 4.

1. *Relationship of effect between components of form and function*: These links are used to represent the way in which a function affects an element of form, similar to the effect link in OPM.
2. *Relationship between functions and common drivers:* The performance of functions can affect a common driver in the same way that a common driver can affect a function.
3. *Effect of policy levers on functions:* These are used to represent the way in which functions are affected by specific components in the physical domain that are controlled by actors in the institutional sphere.
4. *Control of a function by a component of form:* This type of relationship is used to reflect the control of functions by specific elements of form, and is represented by a solid line and black dot.
5. *Control of a policy lever by an institutional actor:* This is represented by a dotted line and black dot, similar to the effect link in OPM.

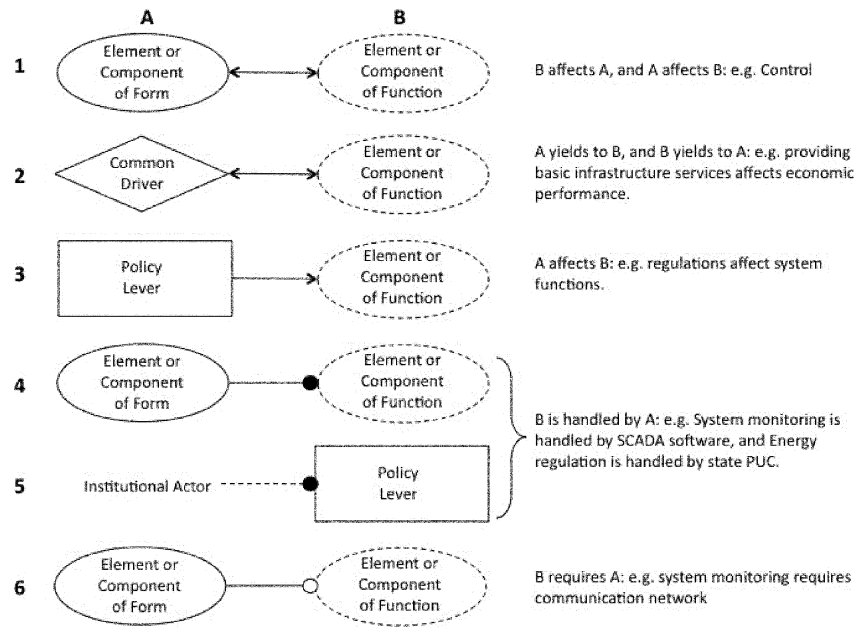


Figure 4. Links for COIM components. (Source: Osorio [2007].)

6. *Functional requirement of a component of form:* A solid line ending in a white dot represents the infrastructure requirements for performing a function.

In summary, we might have a function affecting a component of form, or a component of form affecting a function. Also, a function can affect a common driver. For instance, the provision of broadband services by municipal electric utilities affects economic development. In the same way, a common driver can affect a function or element of form. For instance, adoption of new information technologies by MEUs can affect Supervisory Control and Data Acquisition (SCADA) and Automatic Meter Reading (AMR) systems, and lead to the adoption of Internet Protocol (IP)-enabled solutions. Functions might also require a component of form, but are controlled by another element. For instance, system monitoring requires a communication network, but is controlled by the SCADA system.

Institutional actors can affect functions directly or indirectly by regulating the policy levers affecting the function. These relationships between the institutional sphere and physical domain, called "projections," are represented by dotted lines. Functions and policy levers are controlled by components of form and actors in the institutional sphere, respectively.

The symbols described above can be used to represent the relationships among components of form, function, common drivers, policy levers, institutional actors, and external factors. We can use them to represent relationships of cause and effect (A affects B), control, or requirement among components.

Finally, we borrow a third group of operators from OPM that are especially useful for representing structural and hierarchical relationships. As Figure 5 shows, these are:

1. *Representation of hierarchies:* A black triangle signals the decomposition of a system into subsystems that are "mutually exclusive and comprehensively exhaustive" (MECE). The hierarchical decomposition can be performed at as many hierarchical levels as the modeler or

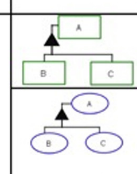
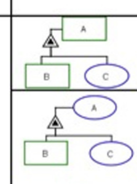
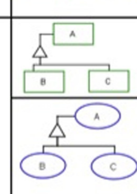
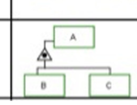
Name	Symbol	Semantics
Aggregation-Participation		A is the whole, B and C are parts.
Exhibition-Characterization		B is an attribute of A and process C is its operation (method). A can be an object or a process.
Generalization-Specialization		A specializes into B and C. A, B, and C can be either all objects or all processes.
Classification-Instantiation		Object A is the class, for which B and C are instances. Applicable to processes too.

Figure 5. Hierarchical operators. (Source: Dori [2002].) [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

researcher considers appropriate for the purpose of the research.
2. *Representation of attributes*: A black triangle inside a white triangle signals the attributes of a system, common drivers, components of form, or functions.
3. *Specialization of components:* A white triangle signals the characteristics of elements in terms of their specialization.
4. *Class of components:* A circle inside a white triangle signals inclusion into types of classes.

3.2. Differentiating between the External and Internal Institutional Spheres

The CLIOS Process definition of nested complexity relates to the complexity created by the embeddedness of the physical system in its surrounding organizational and institutional domain (the institutional sphere). All technical systems are directly affected by their immediate organizational environment, culture, practices, and structural embeddedness.

We examine the effect of organizations on physical system at two levels: *internal* and *external*. We thus extend the CLIOS Process by separating the institutional sphere according to the types of organizational actors and their relationships. This distinction is required due to the different nature of internal and external influences on architecture.

The *internal* level of the institutional sphere is formed by the immediate organizational environment of a technical system or, in other words, the organization holding and operating the technical system. This defines the internal institutional sphere (IIS). Many scholars in organizational theory and behavioral policy sciences have studied the interaction between the internal institutional sphere and the physical domain [DeSanctis and Poole, 1994; Fountain, 2001; W. Orlikowski and Baroudi, 1991; W.J. Orlikowski, 2000; Perrow, 1986].

Several formal and informal actors outside the IIS form the *external* level of the institutional sphere, creating the external institutional sphere (EIS). Components of the EIS can directly or indirectly affect the technical system under study through federal or state regulation, local government rules, the practices of suppliers, or national changes in customer needs. Scholars of the history of technology, privatization, and regulation have analyzed various ways in which public policies have affected the architecture, design, or operation of technical systems in areas such as emissions control and pollution, car safety, and electric power and telecommunications [Donahue, 1989; Hughes, 1983; Laffont and Tirole, 2000; Newbury, 1999; Savas, 2000; Viscusi, Vernon and Harrington, 1997]. Theories about system architecture and product development have explained how technical systems are shaped by changes in customer needs and preferences [Crawley and Weigel, 2004; Maier and Rechtin, 2000; Ulrich and Eppinger, 2004].

Relationships between actors in the IIS and actors in the EIS give rise to a third type of relationships that can have an indirect effect on the technical system. Several scholars have analyzed the different ways in which organizations interact in different economic and regulatory settings [Podolny and Page, 1998; Polenske, 2004; Powell, 1990; Powell and Smith-Doerr, 1994], In such interactions, the institutional sphere might impact the organization and affect the system's performance in ways other than the direct effect on the technical system.

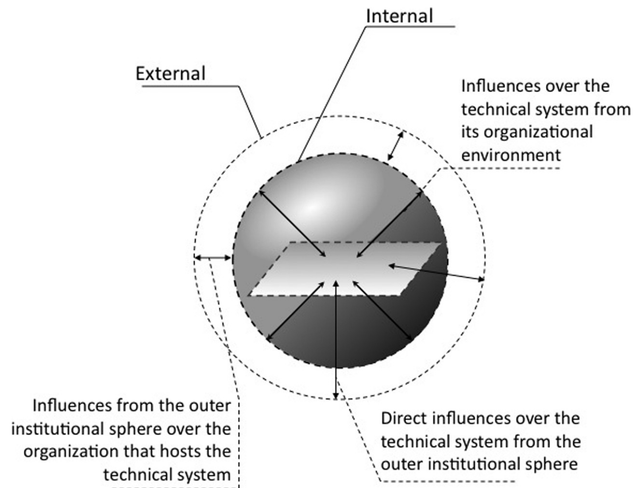


Figure 6. Illustration of extended notion of nested complexity. (Source: Osorio [2007].)

Figure 6 shows the differentiation between the internal and external institutional spheres in our modified model.{FIG6} The internal Institutional Sphere (IS) represents the organizational environment created by the firm or institution managing the system, while the external IS represents all other public and private institutions affecting in some way the technical system and/or its hosting institution.

The influences from within and outside the organization on the technical system are of special interest, because it is necessary to show how actors from the external institutional sphere are related to components of each subsystem and to the organization that governs and directly manages the technical system. The way in which the organization is related to the main components of the physical system needs to be represented as well. To this end, COIM includes two distinctive features for representing a subsystem and the institutional sphere. First, the representation of subsystems in the physical domain includes projections from the actors on the institutional sphere affecting the subsystems. Second, the representation of the institutional sphere differentiates between IIS and EIS. It includes projections from the EIS and IIS to a summarized representation of the physical domain.

3.3. Combining Research Approaches in COIM

The study of the evolution of system architecture requires OPM to be capable of historical analysis and analysis of dominant influences in system architecture. Historical analysis allows for identifying how "path dependence"[2] [Hughes, 1987] can affect a system's organization (structural embeddedness and culture) and technical system (legacy of its architecture). Path dependence and legacy in architecture are

[2]Path dependence refers to how decisions and possibilities for adjusting of changing a system are limited by past decisions taken during the system's design and development, or at any point during its lifetime.

especially important when studying the evolution of a system's architecture. We therefore analyze the physical domain by investigating components of the technical infrastructure that are likely to exhibit legacy influences and can affect the evolution of the architecture of our system.

Analysis of dominant influences allows identifying and understanding the factors, sources and effects of regulatory, social, institutional, organizational, and contextual changes on system architecture. In what follows, we discuss relationships between dominant influences in system architecture and the CLIOS Process at the IIS, EIS, and common drivers levels.

At the internal institutional sphere (IIS) level, an organization's corporate and marketing strategy can affect the functions or forms of its technical infrastructure. New organizational competencies can be generated by decision making at operational and tactical levels. Other dominant influences in the IIS include internal policies or decisions about operation of the technical system, and strategies based on the identification of needs in the user base.

At the e*xternal* institutional sphere (EIS) level, organizations exert three of the most important types of dominant influence in a system's architecture: regulations, changes driven by competitive environment, and changes to the needs to be satisfied by the system. From the perspective of system architecture, the institutional actors can be divided into public and regulatory organizations, private companies providing competitive services, and other formal or informal organizations concerned about the direct or indirect effect of the system on customer needs, regulatory aspects, or other issues such as environment, labor, etc.

At the common drivers level, the dominant influences include two major drivers of the evolution of a system's architecture: new technology and operational costs and efficiency. We consider the technical architecture of a CLIOS as a major component of its physical domain. This technical architecture can be represented as one or more subsystems. If there is more than one, then new technology and efficiency will be drivers common to all subsystems, including parts of the technical infrastructure of the system.

Historical analysis and analysis of dominant influences are included in the analysis of system architecture, which can also be performed using OPM. We can draw analogies between analysis of dominant influences and the CLIOS process. Thus, by integrating system architecture with OPM and the CLIOS process, we make it possible to integrate historical analysis and dominant influences into the representation process. The integration among system architecture, OPM and the representation stage of the CLIOS Process is illustrated in Figure 7.

In this integration, OPM contributes with the detail and rigor from the object-process decomposition. CLIOS Process adds the analysis of the Internal and External Institutional Spheres from the Representation Stage, and System Architecture adds historical analysis, and the study of dominant influences. The purpose of this integration is creating a research method for studying the architectural evolution of a system where most of the learning comes from (i) understanding the system's setting, (ii) understanding and identifying its sources of legacy, influences, and path dependence, and (iii) learning

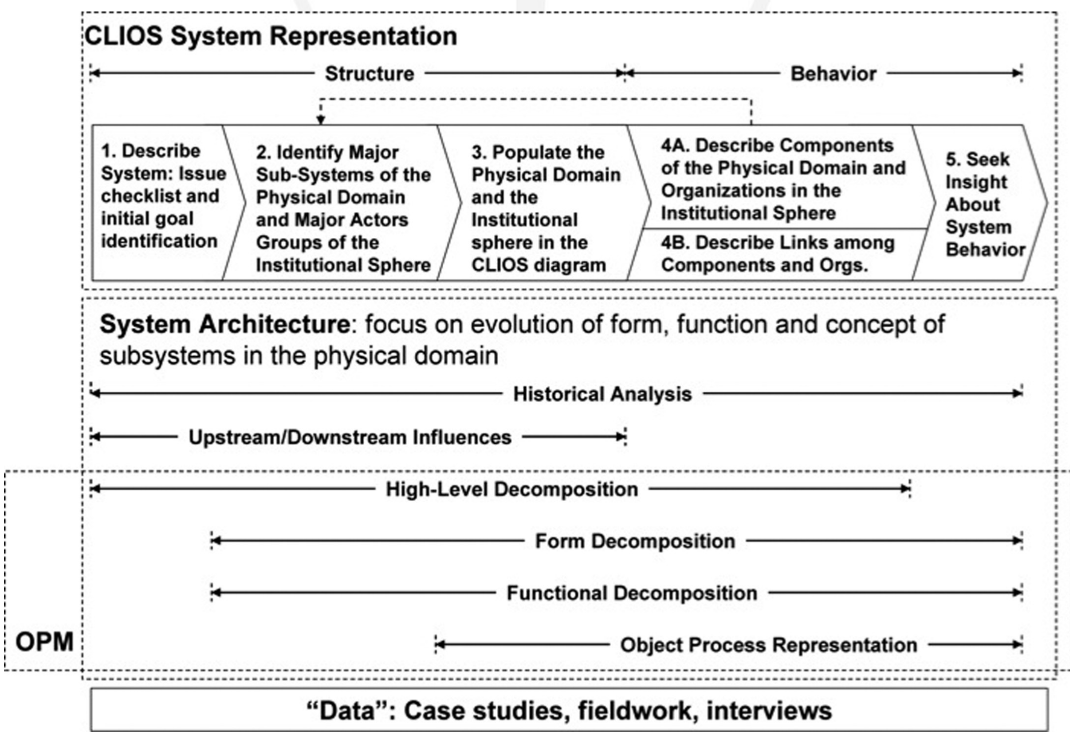


Figure 7. Combined activities of CLIOS process and OPM for system architecture. (Source: Osorio [2007].)

about the system's evolution through the process of detailed representation of the system.

3.4. Focus on System Evolution

Three relevant dominant influences need to be accounted for: (i) evolution of the architecture, (ii) implementation of the architectural changes, and (iii) design. COIM is designed to focus on the process of studying and representing the forces behind system evolution, and on identifying possible path dependencies and architectural legacies. We are less interested in characterizing the behavior of the system in the short run, as we focus on modeling and trying to explain the reasons for the evolution of its architecture over the long run.

We test hypotheses by comparing how a COIM representation of a system should look and how this representation compares with information gathered from the field, including case studies, interviews, and review of documentation. Then, we can revise our research questions and hypotheses accordingly. COIM-based research is, thus, iterative; we refine the representation based on gathered data and gain deeper understanding about the research questions. The process reveals information about the factors affecting the evolution of the system's architecture and exposes evolutionary patterns of components and functions at different hierarchical levels.

Table II and Figure 8 illustrate this general approach. The figure includes a summary of the difference in focus, strengths, and weaknesses between the CLIOS process and OPM, and the value of integrating them into one representation method.

The input for COIM is the set of hypotheses about the architectural evolution of a socio-technical system. We start by studying the overall technical and contextual setting of the system, as well as analyzing its history. We can use data from the literature and documentation on social, regulatory, technical, and organizational dimensions in order to identify path dependence, legacy, and influences on architecture.

We can then use detailed representation of the system's architecture (physical domain) and sources of influence (internal and external institutional sphere, and general context) to learn about the evolution process. We can do this iteratively, constructing our representation from lower to higher complexity. The representation process results from combining the representation stage of the CLIOS process and OPM. As the analysis proceeds, we examine how each step helps us to validate or reject our initial hypotheses.

After a certain point, the researcher might find out that his or her progress in gaining insight about the system has stalled or stopped due to the decreasing marginal productivity of a more detailed historical analysis or a more detailed representation. At this point, one can turn back to fieldwork, using case studies and interviews in order to validate or reject previous findings, and get information that would allow further learning about reasons for and the process of architectural evolution.

This iterative process would allow us to conclude (i) which are the most relevant sources of influence for architectural evolution, (ii) how relevant was the system's architectural legacy and what role did it play in its evolution, (iii) how do the various subsystems of a system interact to foster or hinder its architectural evolution, and (iv) why did the system's architecture evolve.

In some cases, the process will end with no changes to the original research questions and hypotheses, while in others the researcher would have to revisit them, maybe discovering a research path that was overlooked. The next section presents a formalization of this process.

3.5. Using COIM: Steps in the Research Process

COIM analysis proceeds in four stages: (i) understanding the setting, (ii) building a generic model of the system based on original hypotheses, (iii) obtaining data through field research in order to validate or refute the model and hypotheses, and (iv) validating the generic model and confirming findings. Figure 8 shows the steps in each stage.

Starting Point: Before starting a COIM analysis, the researcher must set up the research questions and hypotheses that will be tested by qualitative analysis. This also requires collecting documentation and information, securing the interviews, and making arrangements for field research in order to assemble enough data to test the hypotheses.

Stage 1—Understanding the Setting: The first aim is to understand the setting of the problem under study, identify sources of path dependence and architectural legacy, and identify the major subsystems and actors in the external and internal institutional spheres. This is done in four steps:

Step 1—System Description: COIM analysis starts by presenting a summarized view of the CLIOS System under study, including its goals, and major regulatory factors affecting its most relevant subsystems (e.g., electric power and broadband division). The objective is to highlight why the system is interesting and why the problem under study is relevant.

Step 2—Historical Analysis: We need to understand how the historical evolution of the system has affected its present architecture and culture. If the system under study is not one-of-a-kind, the researcher needs to pay special attention to studying the evolution of the architecture of a *class* of systems, rather than a particular system. The objective of the historical analysis is to identify path dependence on various fronts, including (i) regulatory, (ii) entrepreneurial activity, (iii) role in local economic development, (iv) institutional isomorphism and diffusion of practices, and (v) technology. We also need to understand how this history, which created path dependence, created a legacy that may be shaping the current state of the system's architecture.

Step 3—Identification of Major Subsystems of the CLIOS System: In this step we identify and define the subsystems in the physical domain and the organization groups on the internal and external institutional sphere that are relevant for the purpose of our research.

Step 4—Population of Internal and External Institutional Sphere: We identify the organizations in both spheres and explain their relevance and ways in which they affect the physical domain directly and indirectly. We want to understand how institutional actors affect the physical domain of our CLIOS system. The traditional system architecture analysis of dominant influences

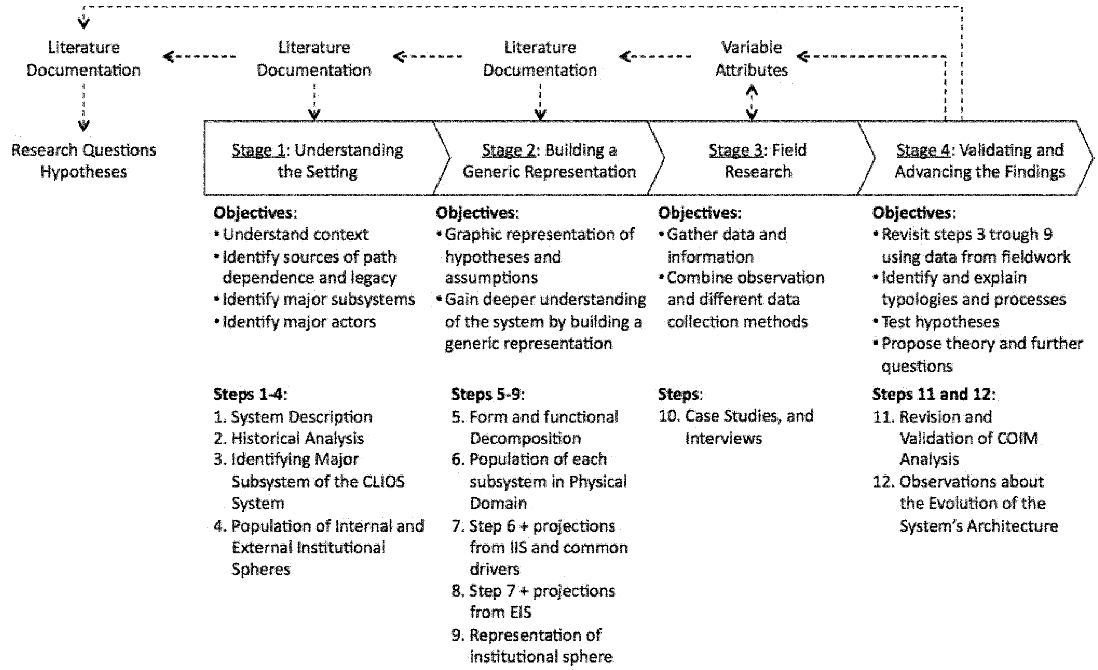


Figure 8. The steps of COIM-based analysis.

becomes part of the CLIOS representation process, making it possible to see more clearly how various actors influence the functions and forms of the different subsystems.

Stage 2—Building a Generic Representation: In this stage we build a generic representation based on our initial understanding of the problem and hypotheses. Representing the system allows us to build understanding about it, the relationships about technical components, and among them and actors in the internal and external institutional spheres. In this stage, the "data" for representing the system are found in publicly available information and literature. In essence, we use the representation process as a research method that will allows us to learn about the architectural evolution of the system and test our research hypotheses. The result of this stage will be a graphic representation of our hypotheses and assumptions. We build the model in five steps:

Step 5—Form and Functional Decomposition: We start by creating a form and functional decomposition of the subsystems in the physical domain to identify: (i) sources of architectural legacy that could explain the architectural evolution of the system and (ii) the "appropriate" hierarchical level at which we would make the object-process representation of the subsystems. The researcher is responsible for defining and justifying the "appropriate" level of hierarchical decomposition. OPM form and function decomposition of technological systems can reach four or more hierarchical levels, but this might not be necessary for all subsystems, as we are interested in understanding how the legacy of the architecture of the main subsystems creates conditions for deployment of a new external function.

Step 6—Population of Each Subsystem in the Physical Domain: We create a representation for each subsystem in the physical domain by combining our hypotheses and information from industry documents and pertinent literature. The objective is to build a generalized model by identifying common drivers among subsystems as well as dominant influences from organizations in the internal and external institutional spheres. These models will be later validated with data from fieldwork.

Step 7—Adding projections from the Internal Institutional Sphere and Identify Common Drivers: Building on Step 6, we identify drivers common to the subsystems and projections from organizations in the IIS. We continually test our hypotheses as we observe the phenomenon.

Step 8—Adding projections from External Institutional Sphere: To the representations created in Step 7, we add projections from organizations in the EIS. As in the previous step, we want to represent the way in which different institutional actors affect the various subsystems in the physical domain.

Step 9—Representation of Institutional Sphere: In this step we build a representation of the internal and external institutional spheres, which includes relationships among actors in the Internal and External Institutional Spheres.

Stage 3—Field Research: An important outcome of the previous stage is the identification of information and questions that are still needed to complete the testing of our initial hypotheses. These data and information requirements provide for the creation of a quasistructured interview protocol to be applied during field research. Data from interviews and site visits will be used to test the hypotheses.

Step 10—Case Studies and Interviews: The objective of this stage is to obtain the data to test the validity of the representation made in Stage 2. COIM is designed to be used with case studies and interviews. Observation, ethnographic studies, interviewing methods, accessing public and confidential documentation, attending industry and academic forums, meetings, and other qualitative methods are various means for collecting missing data. For example, in the research about architectural evolution of municipal electric utilities by Osorio [2007], the author applied quasistructured interviews and case study research, with special focus on why and how the MEUs' architecture evolved.

Once the interviews are done, the author needs to transcribe them and write the case studies in terms of the architectural evolution of the individual systems. At this point, there is no analysis of the case—the information is descriptive only, and needs to be totally free of judgments or opinions. The objective of field research is to advance the level of knowledge about the system and help validate or reject the research hypotheses about the architectural evolution of our system. In addition to providing important parts of the data for COIM, case study research makes it possible to link the theory behind the problem to the interaction between context and processes around the question of *why* and *how* things happen [Hartley, 2004; Silverman, 2005; Yin, 1984].

Stage 4—Validation and Advancement of the Findings: In this stage we validate the generic model using data from fieldwork and make observations about the evolution of the architecture of our system.

Step 11—Revision and Validation of COIM Representation: Once we have finished field research, we proceed to revisit our understanding of the setting and representation (Steps 3–9). The main objective of Step 11 is to test our generic model against data from the case studies, interviews, and other documentation to further our understanding based on the initial hypotheses, generate new insights, discover new research paths, and modify our understanding of the underlying phenomena involved. Again, our main focus is on the reasons for and processes of change.

Step 12—Observations about the Evolution of the System's Architecture: Finally, we should conclude by (i) stipulating the reasons and process of the evolution of the system's architecture, (ii) providing prescription about possibilities for further change and recommended courses of action for future architectural changes, and (iii) providing advice about how to manage the social, political or organizational aspects related to the system under study. We can do this by aggregating our findings from the study of the context, historical analysis, the learning that comes from the process of representing our CLIOS System, and fieldwork. We explain the architectural evolution as a process, determine whether our hypotheses hold, and use this new knowledge for further understanding and better management of the operation, enhancement, and engineering of the system. An application of COIM addressing these issues is currently under development, building on the work of Osorio [2007] that studies the architectural evolution of Municipal Electric Utilities in the United States.

Through these steps, the researcher is able to deepen her understanding about the structure and behavior of the technical subsystems in the physical domain of the CLIOS System, as well as about its social, economic, political, and organizational dimensions.

4. CONCLUSIONS AND FURTHER DIRECTIONS FOR RESEARCH

Good architecture should not prevent the evolution of the system to adjust and positively respond to future changes of its environment, demand, or other external of internal force. However, every so often, the reasons for architectural changes and evolution are lost over time and do not cater to the rationality of the numerous architects and designers of complex socio-technical systems. As result, architectural decisions sometimes prevent adequate adaptation of a system to external changes.

Our main objective in this research has been to propose an integrated approach and method for addressing the difficult task of studying the state and evolution of the architecture of complex socio-technical systems. Applying our proposed framework would allow system architects and system architecture researchers to (i) make sense and have a better understanding of the technical, social, political, and economic factors driving the evolution of the system over time, (ii) make connections among these and other issues that become relevant to their systems by studying the architecture of their systems, (iii) make choices about the possible ways to address the technical, social, policy, and economic problems that arise with change and evolution, and (iv) make or facilitate their occurrence in a possibly predictable, controllable way.

The qualitative research method we have proposed, CLIOS-OPM Integrated Method (COIM), combines the representation stage of the Complex, Large-Scale, Inter-Connected, Socio-Technical (CLIOS) Process [Mostashari and Sussman, 2009] with Object Process Methodology (OPM) [Dori, 2002]. COIM's theoretical framework encompasses system architecture, object-process based conceptual modeling, and qualitative research methods that include historical analysis, case studies, and semistructured interviews.

We suggest that COIM fulfills the needs of researchers of socio-technical systems, allowing them to:

i. Build a general model of the architecture (form, functions, and general concept) of a complex large open socio-technical system, including the relationships between organizational actors.
ii. Dissect the most important subsystems linking components and functions at a hierarchical level of decomposition that is adequate for meaningful analysis.
iii. Identify relationships between components and functions that could invoke evolution paths outside the desired original design. Such evolution paths might result from the interaction between increased structural complexity and technical, social, organizational, and contextual changes that, without analysis of the archi-

tectural evolution, would likely escape the organizational comprehension.

iv. Based on the gained knowledge and awareness of the system's architecture, propose options and courses of action to follow.

Our work is incomplete in several ways. It has not discussed the differences among small, medium, and large-scale complex systems, such as nanodevices, consumer products, and public transportation systems, respectively. It also has not addressed the relevance of industries "clockspeed," or speed of change, in the buildup of architectural legacy of significant importance. We have not discussed the applicability of COIM for studying socio-technical systems that stand alone (e.g., only an electric power system) or are interconnected (e.g., all critical infrastructure networks: electric power, telecommunications, ground and air transportation). We propose that the more interconnected a system is, the higher the need for an approach that could help to uncover and understand the state and evolution path of its architecture.

Most importantly, for space limitation reasons, this paper does not cover an application of COIM. In the second part of this research, the authors apply COIM to the work of Osorio [2007] for explaining the evolution of municipal electric utilities and their involvement on the broadband business. This companion paper is a work in progress.

ACKNOWLEDGMENTS

The authors thank the comments and suggestions from five anonymous reviewers. The original research for this article was generously supported by MIT Communications Futures Program (CFP), its industry sponsors, and NSF Grant #EIA-0306723. The opinions and conclusions of this report, however, represent our views only and do not imply any endorsement by the National Science Foundation, CFP, or its sponsors.

REFERENCES

F. Arcelus and N. Schaefer, Social demands as strategic issues: Some conceptual problems, Strategic Management J 3(4) (1982), 347–357.

G. Blaaw, "What is computer architecture?" Computer architecture: Concepts and evolution, G. Blaaw and F. Brooks (Editors), Addison-Wesley, Reading, MA, 1997, pp. 3–62.

U. Black, "Layered protocols, network architectures and OSI," Data networks: Concepts, theory and practice, U. Black (Editor), Prentice Hall, Upper Saddle River, NJ, 1989, pp. 269–302.

G. Booch, J. Rumbaugh, and I. Jacobson. Unified Modeling Language user guide, Addison-Wesley Professional, Reading, MA, 2005.

E. Crawley and A. Weigel, Esd.340 class lectures on theory of system architecture, Massachusetts Institute of Technology, Cambridge, MA, 2004.

G. DeSanctis and M. S. Poole, Capturing the complexity in advanced technology use: Adaptive structuration theory, Org Sci 5(2) (1994), 121–147.

R. Dodder, J. McConnell, A. Mostashari, S. Sgouridis, and J. Sussman, The CLIOS process: A user's guide to CLIOS, Massachusetts Institute of Technology, Cambridge, MA, 2006a, p. 13.

R. Dodder, J. McConnell, A. Mostashari, S. Sgouridis, and J. Sussman, "The CLIOS process: A user's guide to CLIOS part 2," Massachusetts Institute of Technology, Cambridge, MA, 2006b, p. 14.

R. Dodder, J. Sussman, J. McConnell, and A. Mostashari, "The concept of the "CLIOS process": Integrating the study of physical and policy systems using Mexico City as an example, MIT Eng Syst Symp, Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, 2005, p. 52.

J.D. Donahue, The privatization decision: Public ends, private means, Basic Books, New York, 1989.

D. Dori, Object-process methodology: A holistic systems approach, Springer-Verlag, Berlin, 2002.

D. Dori and M. Choder, Conceptual modeling in systems biology fosters empirical findings: The MRNA lifecycle., Proc Lib Sci ONE (PLoS ONE), 2007.zaq;2

D. Dori, I. Reinhartz-Berger, and A. Sturm, Developing complex systems with object-process methodology using OPCAT, Lecture Notes in Computer Science 2813, Springer, New York, 2003, pp. 570–572.

S. Eppinger, D. Whitney, R. Smith, and D. Gebala, A model-based method for organizing tasks in product development, Res Eng Des 6 (1994), 1–13.zaq;5

C. Fine, Clockspeed, Perseus Books, Reading, MA, 1998.

J.E. Fountain, Building the virtual state: Information technology and institutional change, Brookings Institution Press, Washington, DC, 2001.

J. Hartley, "Case study research," Essential guide to qualitative methods in organizational research, C. Cassey and G. Symon (Editors), SAGE, London, 2004, Vol. 1, pp. 323–333.

T.P. Hughes, Networks of power: Electrification in western society, 1880–1930, Johns Hopkins University Press, Baltimore, 1983.

T.P. Hughes, "The evolution of large technological systems," The social construction of technological systems: New directions in the sociology and history of technology, W.E. Bijker, T.P. Hughes, and T. Pinch, Cambridge, MA, The MIT Press, Cambridge, MA, 1987, 51–82.

F. Jacob, Evolution and tinkering, Science 196(10) (June 1977), 1161–1166.

P.J. Koopman, Jr., A taxonomy of decomposition strategies based on structure, behavior and goals, Des Eng 83, Design Engineering Technical Conferences Vol. 2, 1995.zaq;2

J.-J. Laffont and J. Tirole, Competition in telecommunications, The MIT Press, Cambridge, MA, 2000.

A. MacCormack, Innovation and uncertainty, Technol Oper Management Sem, Harvard Business School, Boston, MA, 2005.zaq;2

A. MacCormack and R. Verganti, Managing the sources of uncertainty: Matching process and context in software development, J Prod Innovation Management 20(3) (2003), 217–232.

M.W. Maier and E. Rechtin, The art of systems architecting, CRC Press, Boca Raton, FL, 2000.

R. Mardones, Chile: Transantiago reloaded, Rev Ciencia Pol 28(1) (2008), 103–119.

H. Maturana and F.J. Varela, Autopoiesis and cognition. The realization of the living, Reidel, Dordrecht, 1980.

M.H. Meyer and J. Utterback, The product family and the dynamics of core capability, Sloan Management Rev 34 (Spring 1993), 29–47.

A. Mostashari and J. Sussman, A framework for analysis, design and management of complex large-scale interconnected open sociotechnical systems, Int J Decision Support Syst Technol 1(2) (2009).zaq;2

D.M. Newbury, Privatization, restructuring and regulation of network utilities, The MIT Press, Cambridge, MA, 1999.

W. Orlikowski and J.J. Baroudi, Studying information technology in organizations: Research approaches and assumptions, Inform Syst Res 2(1) (1991), 1–28.

W.J. Orlikowski, Using technology and constituting structures: A practice lens for studying technology in organizations, Org Sci 11(4) (2000), 404–428.

C. Osorio, Architectural innovation, functional emergence and diversification in engineering systems, Engineering Systems Division, Ph.D. in Technology, Management and Policy, Massachusetts Institute of Technology, Cambridge, MA, 2007, p. 243.zaq;3

M.C. Pelayo, "Transantiago y su impacto: Transporte público y sus repercusiones en la salud," Ciencia y Trabajo, Santiago, Chile, 2007, Vol. 9, pp. 88–92.

M. Peleg and D. Dori, The model multiplicity problem: Experimenting with real-time specification methods, IEEE Trans Software Eng 26(8) (2000), 742–759.

C. Perrow, Complex organizations: A critical essay, McGraw-Hill, New York, 1986.

T. Pimmler and S. Eppinger, Integration analysis of product decomposition, Des Theory Methodol Conf, ASME, 1994, pp. 343–351.

J.M. Podolny and K.L. Page, Network forms of organization, Annu Rev Sociol 24 (1998), 57–76.

K. Polenske, Competition, collaboration, and cooperation: An uneasy triangle in networks of firms and regions, Regional Stud 38(9) (2004), 1029–1043.

W. Powell, Neither market nor hierarchy: Network forms of organization, Res Org Behav 12 (1990), 295–336.

W. Powell and L. Smith-Doerr, "Networks and economic life," The handbook of economic sociology, N. S. a. R. Swedberg (Editor), Princeton University Press, Princeton, NJ, 1994, pp. 368–402.

D. Rhodes, A. Ross, and D. Nightingale, "Architecting the system of systems enterprise: Enabling constructs and methods from the field of engineering systems," IEEE Int Syst Conf, IEEE, Vancouver, Canada, 2009.zaq;2

E.S. Savas, Privatization and public-private partnerships, Chatham House, Seven Bridges Press, New York, 2000.

D. Silverman, Doing qualitative research: A practical handbook, SAGE, London, 2005.

H. Simon, Bounded rationality and organizational learning, Org Sci 2(1) (1991), 125–134.

H. Simon, Can there be a science of complex systems? Unifying Themes in Complex Systems: Proceedings of the First International Conference on Complex Systems. Y. Bar-Yam (Editor), Perseus Books Group, Boston, 1997, 3–14.

J. Sterman, Business dynamics: Systems thinking and modeling for a complex world, Irwin/McGraw-Hill, Boston, 2000.

N. Suh, "Design of systems," Axiomatic design, N. Suh (Editor), Oxford University Press, Cambridge, MA, 1998, pp. 192–237.

The Open Group (TOG), Open group architectural framework: Introduction to the architecture development method, TOG, San Francisco, 2001.

K. Ulrich and S. Eppinger, Product design and development, McGraw-Hill, New York, 2004.

K.T. Ulrich, The role of product architecture in the manufacturing firm, Res Policy 24 (December 1995), 419–440.

H. van Vliet, "Software architecture," Software engineering: Principles and practice, V. Vliet (Editor), Wiley, New York, 2001, pp. 253–288.

W.K. Viscusi, J.M. Vernon, and J.E. Harrington, The economics of regulation and antitrust, The MIT Press, Cambridge, MA, 1997.

T. Weilkiens, Systems engineering with SysML/UML: modeling, analysis, design, Morgan Kaufman/OMG Press, Burlington, MA, 2007.

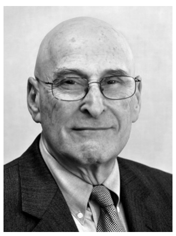
D. Whitney, E. Crawley, O. de Weck, S. Eppinger, C. Magee, J. Moses, W. Seering, J. Schindall, and D. Wallace, The influence of architecture in engineering systems, Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, 2004, p. 30.zaq;4

T. Winograd, and F. Flores, Understanding computers and cognition: A new foundation for design, Ablex, New York, 1990.

R. Yin, Case study research: Designs and methods, SAGE, Beverly Hills, CA, 1984, Vol. 5.

Carlos A. Osorio is an Associate Professor and the Founding Director of the Master on Innovation at Universidad Adolfo Ibanez in Chile, a Visiting Professor of Innovation at Deusto Business School (Spain), and an affiliate to the Berkman Center for Internt and Society. His research and teaching work focuses on innovation and new product development processes and the architecture of complex systems. He has been a visiting research scientist at MIT Media Lab and a research associate at the Center for International Development at Harvard University. Professor Osorio has consulted on innovation and new product development for firms in the energy, information technology, finance, mining, and biotechnology industries, including some Fortune 100 companies. He founded Designing Better Futures, a design and innovation consultancy focused on developing new products, services, and experiences. He received his Ph.D. in Technology, Management and Policy and an M.S. in Technology and Policy from MIT Engineering Systems Division and, as a Fulbright Scholar, a Master in Public Policy from Harvard University's John F. Kennedy School of Government. He holds a B.S. in Engineering from the University of Chile. He is member of the Academy of Management.

Dov Dori is Information and Systems Engineering Professor and Head of the Enterprise System Modeling Laboratory at the Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology, and a Research Affiliate at the Engineering Systems Division, Massachusetts Institute of Technology, where he lectures on a regular basis. Between 1999–2001 and 2008–2009 he was Visiting Professor with the Engineering Systems Division, the Department of Aeronautics and Astronautics, and Sloan Business School at MIT. His research interests include conceptual modeling of complex systems, systems architecture and design, and software and systems engineering. Professor Dori has invented and developed Object-Process Methodology (OPM), presented in his 2002 book. He won the Technion Klein Research Award for OPM, the Hershel Rich Innovation Award for OPCAT, the OPM supporting software, and the Dudi Ben Aharon Research Award for Document Image Understanding. Professor Dori has founded OPCAT Ltd., which develops an OPM modeling support environment. He initiated and headed the series of international conferences on Model-Based Systems Engineering held at Technion, Israel in 2007 and 2009, and at George Mason University, Fairfax, VA in 2010. Professor Dori has authored over 200 publications. He is a Fellow of the International Association for Pattern Recognition, a Senior Member of IEEE and ACM, and a member of INCOSE.



Joseph M. Sussman is the JR East Professor (endowed by the East Japan Railway Company) in the Department of Civil and Environmental Engineering and the Engineering Systems Division at the Massachusetts Institute of Technology (MIT), where he has served as a faculty member for 40 years. He is the author of *Introduction to Transportation Systems*, a graduate text published in 2000, in use at a number of universities in the United States and abroad. His book *Perspectives on Intelligent Transportation Systems* (*ITS*) was published in 2005. Sussman received the Roy W. Crum Distinguished Service Award from TRB, its highest honor, “for significant contributions to research,” in 2001, and the CUTC Award for Distinguished Contribution to University Transportation Education and Research from the Council of University Transportation Centers in 2003. In 2002 ITS Massachusetts named its annual “Joseph M. Sussman Leadership Award” in his honor. He became a fellow of the American Association for the Advancement of Science in 2007. Dr. Sussman specializes in the study of Complex, Large-Scale, Interconnected, Open, Sociotechnical (CLIOS) Systems, working in many applications areas, and has developed the CLIOS Process to study such systems. Dr. Sussman has focused recently on developing a new methodology for regional strategic transportation planning (RSTP) as a special case of the CLIOS Process, integrating ideas from strategic management, scenario-building, and technology architectures, and applying it to cases in the United States and abroad. Currently his work in this area deals with transportation, technology, and sustainability in Mexico City and Kuala Lumpur, Malaysia and most recently in Portugal. He initiated the transportation systems focus area for the MIT-Portugal Program, a major new $40 million, 5-year program of education and research, which began in 2006. His work here includes participation in the development of a new international MSc in transportation systems in collaboration with three Portuguese universities, and research in Regional Strategic Transportation Planning (RSTP), Intelligent Transportation Systems (ITS), and High-Speed Rail. His research in railroads focuses on service reliability, rail operations, maintenance, high-speed rail (HSR), and risk assessment; he has had a major impact on the railroad industry in the United States and abroad, and has written several prize-winning papers. He has worked extensively on Intelligent Transportation Systems (ITS), helping to build the U.S. national program. While serving as the first Distinguished University Scholar at IVHS AMERICA (1991–1992), he was a member of the core group that wrote the Strategic Plan for IVHS in the United States, a 20-year plan for research, development, testing, and deployment which has shaped the U.S. ITS program. He has worked on the development of an “intelligent corridor” in Bangkok, a comparison of ITS programs in Western Europe, Japan, and the United States, commercial vehicle operations, building regional ITS architectures, emergency response, and institutional issues concerning ITS and the provision of “flexibility” in surface transportation through ITS technologies, including studies in Houston and Portugal and in connection with Vehicle-Infrastructure Integration (VII). He was the program chair of the ITS America Annual Meeting in 2000, and has conducted short courses in ITS for practicing professionals in the United States and abroad. Currently, he serves as the chairman of ITS Advisory Committee—mandated by SAFETEA-LU—for the US DOT, charged with advising the department on all facets of the ITS program. He has worked on the application of computers to engineering problem-solving, specializing in simulation methods and their application to the transportation area, and he contributed to the development of ICES (Integrated Civil Engineering System), among the most widely used computer systems in the engineering field. Dr. Sussman chaired the TRB committee overseeing the Federal Railroad Administration’s R&D program from 1996 to 1999 and chaired, in 2006, a TRB panel reviewing the federal transportation strategic plan for R&D. Further, he chaired a TRB Task Force which produced a major report entitled “Airport System Capacity—Strategic Choices” in 1990. He has served on review panels for transportation programs at Northwestern, the University of Toronto, Cornell, and the University of Michigan (chair). He currently serves on advisory committees at CCNY and Cal-IT^2, a joint venture of UC-Irvine and UC-San Diego, and chairs the Board of Advisors at the Technical University of Delft in the Multi-Actor Systems (MAS) area in the Program on Technology and Policy. Dr. Sussman earned a B.C.E. from City College of New York in 1961, an M.S.C.E. from the University of New Hampshire in 1963, and a Ph.D. in Civil Engineering Systems from MIT in 1967. He joined the MIT faculty in 1967. From 1977 to 1979, Professor Sussman served as the Associate Dean of Engineering for Educational

Programs. From 1980 to 1985, he served as Head of the Department of Civil Engineering at MIT. From 1986 to 1991, he served as Director of the Center for Transportation Studies (CTS) at MIT. During his term, research volume grew by 400% to more than $4 million annually at that time, reflecting an important expansion of CTS' research agenda. Dr. Sussman is a member of the American Society of Civil Engineers, Transportation Research Forum, Transportation Research Board (Executive Committee chair in 1994; member, 1991–1998), ITS America (Board of Directors, 1995–2001) and ITS Massachusetts (Board of Directors, 1996–2001). He cofounded Multisystems of Cambridge, MA in 1966 (now part of Transystems).

<enote>AQ1: Please supply page numbers of quotations

<enote>AQ2: Please give page range

<enote>AQ3: Is this a reference to a specific page in the thesis or the total number of pages? If the latter, we should change to 243 pp. as opposed to p. 243.

<enote>AQ4: Is this a reference to a specific page in the item or the total number of pages? If the latter, we should change to 30 pp. as opposed to p. 30.

<enote>AQ5: Please cite or delete listing