SYSTEM ARCHITECTURES FOR LIFE SUPPORT SYSTEMS

Alexander H. Levis and **Lee W. Wagenhals**
George Mason University, Fairfax, VA, USA

Keywords: operational architectures, technical architectures, functional architectures, structured analysis, object orientation, executable model

Contents

1. Introduction
2. On Architectures
2.1. Architects and Architectures
2.2. Architecting in Systems Engineering
3. Architecture Development Process
3.1. Structured Analysis Approach
3.1.1. Functional Decomposition and Activity Model
3.1.2. Data Model
3.1.3. Rule Model
3.1.4. Dynamics Model
3.1.5. Integrated Dictionary and Model Concordance
3.2. Object Oriented Approach
3.2.1. UML Elements and Diagrams
3.2.2. An Object Oriented Process
3.3. The Physical Architecture View
4. Conversion to the Executable Model
5. Conclusion
Glossary
Bibliography
Biographical Sketches

Summary

The rapid changes in information technology have precipitated changes in the structure and operations of organizations and increased uncertainty about system requirements that support or enable these operations. A way to deal with these uncertainties is to require an unprecedented level of interoperability among systems.

One way to achieve this is to use architectures that can provide current or future descriptions of a “domain” composed of components and their interconnections, actions or activities those components perform, and rules or constraints for those activities. These architectures, while they will change over time, will change at a much slower rate than the actual systems used to implement them.

Because of their stability, they can act as important guides to system acquisition decisions as well as defining operational concepts and business rules. The goal is to describe architectures using multiple views that answer questions regarding the operational capability that systems built conformant to the architecture can provide.

1. Introduction

An architecture framework provides common definitions, data, and references, and describes a set of products that comprise the multiple views of an architecture. Several frameworks have been proposed. A recent one is the C4ISR architecture framework of the US Department of Defense. There is also a traditional systems engineering framework based on the structured analysis approach and there are others based on software engineering approaches and expressed in terms of object orientation. These frameworks sometimes provide guidance for the design of the architecture, while in other instances they only define the architecture products while leaving unspecified the process that is to be used.

This article describes the role of the architect, characterizes the multiple views of an architecture and the products used to describe them, and establishes the criteria for determining the data needed in an architecture to support the role of the architect. To address these issues, the remainder of this article is divided into three sections. Section 2 describes the fundamentals of architectures and the role of the architect. Section 3 presents a generic process for creating an architecture that is capable of describing to the customer the behavioral and performance aspects of the systems that implement the architecture. Section 4 addresses the creation of executable models for the evaluation of the properties of an architecture.

2. On Architectures

2.1. Architects and Architectures

The Greek word αρχιτεκτων (*architecton*) means master builder or master mason. The term describes one who designs and builds structures whose form and function are both appealing and useful. The main contribution of the architect is the conceptualization and design of a unique structure to meet the client's needs. The architect has the special role of eliciting and converting the needs and desires of the customer who commissions him into a design that will be especially satisfying to that customer.

It is possible to derive several characteristics of system architects and the architectures they produce. The first is that an architect is needed only if the system is unprecedented and complex. If satisfactory working systems have already been built and the designs exist, there is no need for an architect. An architect is not needed if the system is very simple and can be constructed directly by a contractor. The architect's responsibility is different from the general contractor's. The architect is driven by the special needs of the customer and tends to develop the architecture in a top-down manner. Indeed, the task of the architect is to elicit those needs and to produce a description that can demonstrate to the customer that the system to be produced in conformance with the architecture will satisfy the customer's wants and needs. This means that the architecture does not include the details of the final system designs.

A key issue in the above description is where the architect's work in describing the architecture ends and the system design begins. The problem of demarcation between architecture and design—never an easy one to determine—is especially daunting in the

case where any future architecture will be populated in the design phase by existing systems, the so-called legacy systems. So, while the architecture definition process is thought of as being top-down, the presence of legacy systems introduces constraints in the design of the architecture that are best addressed in a bottom-up approach.

The architect develops and presents the architecture as a set of abstract views or models. The abstraction occurs in two dimensions: generalization and composition. The architect begins with a very general description of the system. Based on discussions with the customer, the architect arranges and specializes the components to suit the customer's needs and desires. Note that the number of components in the architecture and, thus, its complexity do not need to increase substantially in this specialization process. The number of components does increase as more detail is incorporated in the architecture. Adding detail is generally accomplished by decomposing the basic components of the architecture into their constituent parts. Decomposition can significantly increase the number of components of the architecture and thus its complexity. Because of this, the architect should use decomposition only as necessary to address the questions and concerns of the customer. The customer and the architect assume that these components will work properly because they will be constructed and installed in accordance with established codes and guidelines. The actual system design and implementation will involve the specialization of the architecture and the addition of all of the details of the design so that the system can be manufactured. This specialization process is the task of the general contractor or system engineer. The actual system will be the most specialized version of the architecture. There can be many ways of specializing a single architecture into actual systems. The selection of the actual design can be determined by cost and technology factors. Thus, an architecture can be a valid description of a way of satisfying a customer's need over a long period, even as the specific techniques for implementing the architecture change.

One thinks of an architecture as being implemented by many diverse interacting systems. The architect needs to be knowledgeable not only about the individual systems, but also about the interrelationships among them. Furthermore, the architect must use creativity and vision because of the unprecedented and complex nature of the design and the lack of an initial clear definition of the needs and requirements for the component systems. It is important that the architect be able to show and discuss with the customer what properties the architecture will have. This means that the architectural models must be capable of providing insight into the logical and behavioral aspects of the architecture and the performance aspects of systems that are conformant to the architecture. This important criterion influences the process and the techniques for developing a description of an architecture of a system.

Systems architecting is part of the system engineering process and relies on many of the methodologies that have been developed. The architect has many tools and techniques available to describe the architecture. Two major paradigms that are appropriate are the traditional structured analysis and design technique (SADT) and the object oriented approach that originated with software systems. Both offer advantages and both can fall short of the requirement of being able to convey the logical, behavioral, and performance properties of the architecture. This is because both approaches rely on static pictures, diagrams, and textual descriptions to define the architecture. However,

an architecture is instantiated with dynamic systems that interact with their environment over time. To describe and understand fully the dynamic aspects of the system requires an executable model.

2.2. Architecting in Systems Engineering

The complete representation of a single architecture requires different views. These views can be categorized by their perspective. In defining an architecture, several perspectives need to be described. To begin with, since the architecture will be created so that the systems that populate it will perform some useful function, the process or activities that need to take place in order for the systems to accomplish their purpose must be described. In an information system, the processes receive and transform data that "flow" between them. These processes or activities follow rules that determine the conditions under which they occur and the type of outputs they produce. In addition, the processes should occur in some order based on the initial conditions of the system, and several processes may occur concurrently and asynchronously. In addition to the processes or activities, it is necessary to describe the components that will implement the design: the hardware, software, personnel, and facilities that will comprise the system and perform the processes.

This fundamental notion leads to the definition of the two basic architectural constructs in structured analysis. A functional architecture is a set of activities or functions, arranged in a specified partial order that, when activated, achieves a set of requirements. Similarly, a physical architecture is a representation of the physical resources, expressed as nodes, that constitute the system and their connectivity, expressed in the form of links. (In the systems engineering literature, the term system architecture has often been used to denote the physical architecture. This has been a source of confusion because the term system architecture denotes different constructs in different frameworks.) Both definitions should be interpreted broadly to cover a wide range of applications; furthermore, each may require multiple representations or views to describe all aspects.

Before even attempting to develop these representations, the operational concept must be defined. This is the first step in the architecture development process. An operational concept is a concise statement that describes how the goal will be met. There are no analytical procedures for deriving an operational concept for complex, unprecedented systems. On the contrary, given a set of goals, experience, and expertise, humans invent operational concepts. The development of an architecture is both an art and a science. The conceptualization of an operational concept falls clearly on the art side. A good operational concept is based on a simple idea of how the overriding goal is to be met. For example, centralized decision making and distributed execution represents a very abstract operational concept that lends itself to many possible implementations. As the architecture development process unfolds, it becomes necessary to elaborate on the operational concept and make it more specific. The clear definition and understanding of the operational concept is central to the development of compatible functional and physical architecture views.

Analogous to the close relationship between the operational concept and the functional architecture view—to the extent that often a graphical description of the operational

concept is improperly presented as the functional architecture—is the relationship between the technical architecture view and the physical one.

A technical architecture (TA) view is a minimal set of rules governing the arrangement, interaction, and interdependence of the parts or elements whose purpose is to ensure that a conformant system satisfies a specified set of requirements.

It provides the basis from which engineering specifications can be derived, guiding the implementation of the systems. The technical architecture view corresponds, in broad terms, to the "building code," the set of standards, guidelines, and best practices, that any architect must take into consideration.

It is the technical architecture view that provides the connection between the abstract descriptions of the physical and functional architecture views and the implementation of the detailed system design.

When architects define the technical architecture view, they are providing guidance on the further specialization and decomposition of the components of the physical and functional architecture that will be accomplished in the detailed engineering design of the system.

All of these representations of the architecture, even when they describe the dynamic behavior of the architecture, are static. They are inadequate for analysis of the behavior and performance aspects of the architecture. In Section 3, details of the models used in these representations are described.

They contain a great deal of information but, in general, they are not well suited to answering the main concerns of the customer. In order to analyze the behavior and performance of the architecture and address the concerns of the customer, an executable model is derived from them. After all, the systems to be designed are dynamic ones. An executable model is a dynamic model; it can be used to analyze the properties of the architecture and it can be used to carry out simulations. However, it also serves in a subtler, but very important role.

It becomes the litmus test by which one can determine whether the description of the system architecture—as given by a set of static representations or models—is sufficient with respect to the set of questions to be asked of the architecture about the dynamic behavior of the systems.

Indeed, the methodologies, whether structured analysis based or object oriented based, become rigorous when an executable model is derived and the condition is imposed that all information contained in the executable model must be traced back to one or more of the static views.

A key use of the executable model is to allow the architect to shift the locus of discourse with the customer or user from the architecture views to the behavior and performance that the architecture enables. This concept is shown in Figure 1. The figure points out that the static descriptions of the architecture are a means to an end; the end is to

provide the system users with capabilities that support the organization's goals.

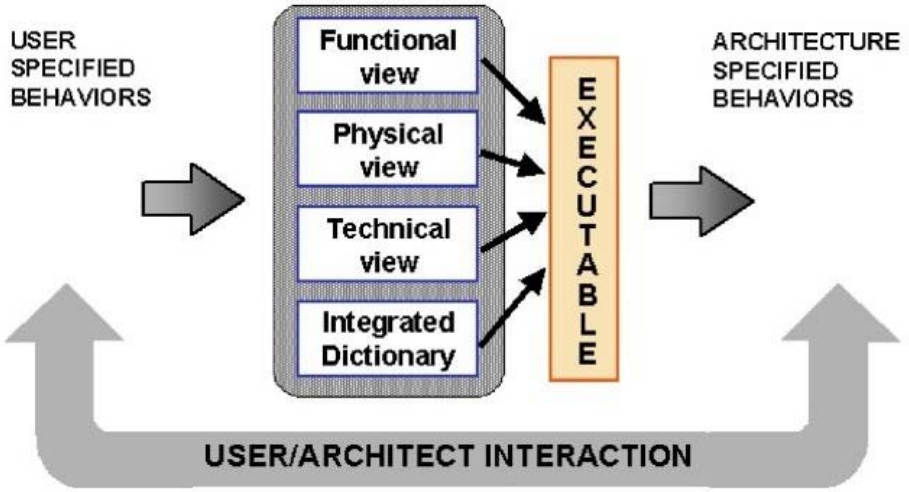

Figure 1. Behavior as the locus of discourse

Section 3 describes the two paradigms for the architecture development process. While structured analysis and object orientation are conceptually different, many of the same tools can be used to construct the various representations that comprise each approach. Furthermore, once the executable model is obtained, whether through structured analysis or object orientation, the evaluation phase is the same.

3. Architecture Development Process

These are six fundamental steps in developing an architecture (as shown in Figure 2); it is the responsibility of the (chief) architect that they are followed in every architecture development effort. The first step reinforces the idea that architectures must be designed for a purpose—to address a particular set of problems. This purpose and the associated problems must be articulated before the development of the architecture and the production of the diagrams and tables—sometimes referred to as products—that describe the views begins.

The second step is the determination of the scope of the architecture. In systems engineering terms, this corresponds to determining the system boundary; in other words, determining which elements are going to be considered within the architecture and which will be considered as part of the environment. The third step is closely related to the first one: the choice of attributes that are to be included is directly dependent on the questions to be answered.

At this point, the architect is ready to determine which specific products are needed. Note that the first four stages require the involvement of few persons: the architect, the customer, and a small staff that supports the architect. The fifth step is the labor intensive one in which the architecture is designed and the products produced. This step is guided by the previous four steps; failure to do so may easily result in a set of diagrams and tables unable to address the last—and most critical—step: the use of the

architecture for the intended purpose; that is, to provide answers to the original questions. The development of the executable model could be included either in step 5 or 6.

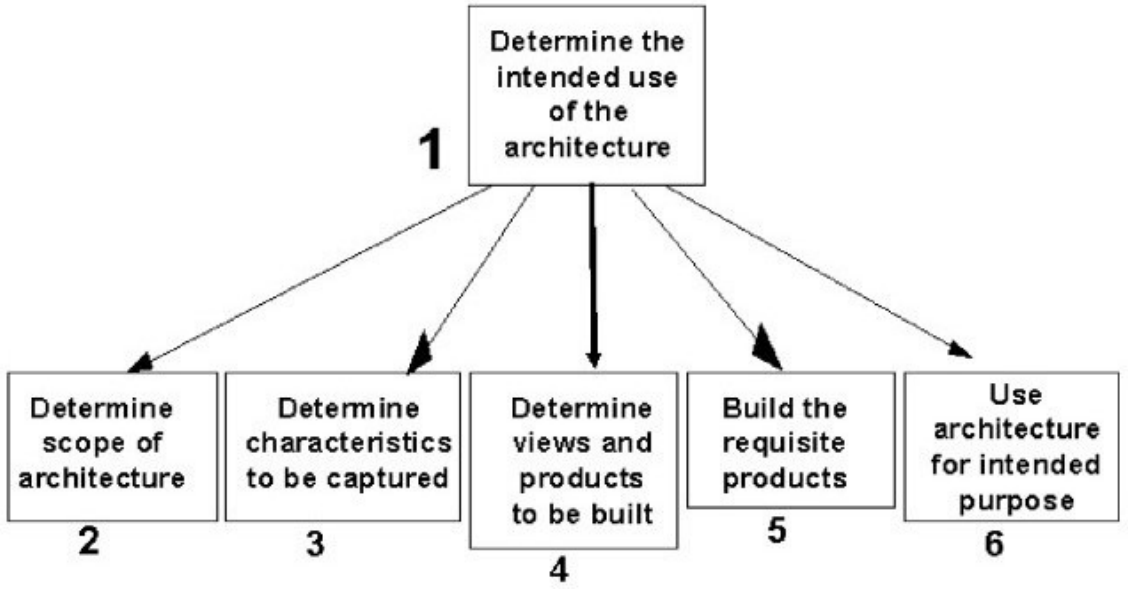

Figure 2. Universal guidance for architecture development

Given this basic design process, the next step is to develop specific procedures that realize the functions indicated in each one of the boxes. Procedures based on either structured analysis or object orientation can be used.

3.1. Structured Analysis Approach

The structured analysis approach has its roots in SADT, which originated in the 1950s and now encompasses structured design, structured development, structured systems analysis, and the many variants that have appeared since then, often embodied in software packages for computer-aided requirements generation and analysis. This approach can be characterized as a process oriented one, in that it considers as the starting point the functions or activities that the system must perform. A second characterizing feature is the use of functional decompositions and the resulting hierarchically structured diagrams. However, to obtain a full specification of the architecture that allows the derivation of the executable model, in addition to the process or activity model, a data model, a rule model, and a dynamics model are required. Each one of these models contains interrelated aspects of the architecture description. For example, in the case of an information system, the activities or processes receive data as input, transform it, and produce data as output. The associated data model describes the relationships between these same data elements. The activities take place when some conditions are satisfied. These conditions are expressed as rules associated with the activities. But for the rules to be evaluated, they require data that must be available at that particular activity with which the rule is associated; the output of the rule also consists of data that control the execution of the process. Furthermore, given that the architecture is for a dynamic system, the states of the system need to be

defined and the transitions between states identified to describe its dynamic behavior. State transition diagrams are just one way of representing this information. Underlying these four models is a data dictionary or, more properly, an integrated system dictionary, in which all data elements, activities, rules, and flows are defined. The construct that emerges from this description is that a set of interrelated views, or models, is needed to describe an architecture using the structured analysis approach.

The activity model, the data model, the rule model, and the supporting integrated system dictionary, taken together, constitute the functional architecture view of the system. The term functional architecture has been used to describe a range of representations: from a simple activity model to the set of models defined here. What a functional architecture does not contain is the specification of the physical resources that will be used to implement the functions or the structure of the human organization that is supported by the information system. These descriptions are contained in the physical architecture.

-
-
-

Bibliography

C4ISR Architecture Working Group, US Department of Defense. (1997). *C4ISR Architecture Framework Version 2.0.* [An architecture framework that has been mandated for US defense systems is described in this document.]

DeMarco T. (1979). *Structured Analysis and Systems Specification*. Englewood Cliffs, NJ: Prentice Hall. [This is a classic work on the subject.]

National Institute for Standards and Technology (NIST). (1993). *Federal Information Processing Standard No. 183: Integration Definition for Function Modeling* (FIPS 183). Gaithersburg, MD: NIST. [Contains important standards relating to architectures.]

Fowler M. and Scott K. (2000). *UML Distilled, Second Edition*. Reading, MA: Addison-Wesley. [This is an excellent work on UML.]

Gane C. and Sarson T. (1978). *Structured Systems Analysis: Tools and Techniques.* Englewood Cliffs, NJ: Prentice Hall. [This is a classic work on structured analysis techniques.]

Levis A.H. (1999). Systems architectures. *Handbook of Systems Engineering and Management* (ed. A.P. Sage and W.B. Rouse), pp. 427–456. New York: John Wiley. [This is a complimentary overview of system architectures and is presented in a comprehensive handbook.]

Marca D.A. and McGowan C.L. (1987). *Structured Analysis and Design Technique.* New York: McGraw-Hill. [This is a classic work on structured analysis techniques.]

Rechtin E. (1991). *Systems Architecting: Creating and Building Complex Systems*. Englewood Cliffs, NJ:

Prentice Hall. [This is one of the first works to discuss systems architecting in systems engineering.]

Rechtin E. (1992). The art of systems architecting. *IEEE Spectrum*, **29**(10), pp. 66–69. [This is an overview of systems architecting in systems engineering.]

Rechtin E. and Maier M. (1996). *The Art of Systems Architecting*. Boca Raton, FL: CRC Press. [This is a generally excellent work discussing systems architecting in systems engineering.]

Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W. (1991). *Object-oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall. [This is a classic work on object oriented design.]

Sage A.P. (1993). Object oriented methodologies in decision and information technologies. *Information and Decision Technologies* **19**, pp. 31–53. [This is a review of the basic concepts of object oriented design and formulation of a methodology for the design of decision support systems.]

Solvberg A. and Kung D.C. (1993). *Information Systems Engineering.* Berlin: Springer-Verlag. [This contains an excellent discussion of the systems engineering of information systems.]

Wagenhals L.W., Shin I., Kim D., and Levis A.H. (2000). C4ISR architectures II: a structured analysis approach to architecture design. *Systems Engineering* **3**(4), pp. 248–287. [This is an earlier, more detailed presentation of material contained in the EOLSS article.]

Ward P. and Mellor S. (1986). *Structured Development of Real-time Systems.* New York: Yourdon Press. [This is a classic work on structured analysis.]

Yourdon E. (1989). *Modern Structured Analysis.* Englewood Cliffs, NJ: Yourdon Press. [This is also a classic work on structured analysis.]

Yourdon E. and Constantine L. (1975). *Structured Design.* New York: Yourdon Press. [This is another classic work on structured analysis.]

Biographical Sketches

Dr. **Alexander H. Levis** is Chief Scientist of the USAF, on leave from George Mason University, Fairfax, VA, where he is University Professor of Electrical, Computer, and Systems Engineering. He is associated with the C3I Center, where he heads the Systems Architectures Laboratory. He was educated at MIT, where he received B.Sc. (1965), M.Sc. (1965), M.E. (1967), and Sc.D. (1968) degrees in Mechanical Engineering, with control systems as his area of specialization. He also attended Ripon College, where he received an AB degree (1963) in Mathematics and Physics. He taught systems and control theory at the Polytechnic Institute of Brooklyn from 1968 to 1973 and, for the following six years, he was with Systems Control, Inc. of Palo Alto, CA, where he was manager of the systems research department. From 1979 to 1990 he was a Senior Research Scientist at the MIT Laboratory for Information and Decision Systems. He joined George Mason University and the C3I Center in 1990. Dr. Levis is a Fellow of the Institute of Electrical and Electronic Engineers (IEEE) and past president of the IEEE Control Systems Society, from which he received in 1987 the Distinguished Member award. He is also a Fellow of the American Association for the Advancement of Science (AAAS), a senior member of the American Institute of Aeronautics and Astronautics (AIAA), and a member of INCOSE and AFCEA. He has received the Third Millennium medal from IEEE and twice the Exceptional Civilian Service medal from the USAF.

Lee W. Wagenhals is a Research Associate Professor with the Center of Excellence Command, Control, Communications, and Intelligence at George Mason University in Fairfax, VA. He is currently the acting director of the System Architectures Laboratory. His degrees include a B.Sc. in Electrical Engineering from Lehigh University (1965), an M.Sc. from the Air Force Institute of Technology (1971), and a Ph.D. in Information Technology from George Mason University (2000). He joined George Mason University

in 1992, where he has worked on a variety of research tasks related to the design, analysis, and evaluation of architectures of Department of Defense information systems. He teaches graduate and undergraduate system engineering courses at George Mason University and short courses on C4ISR architectures, sponsored by the Armed Forces Communications and Electronics Association (AFCEA). Dr. Wagenhals has over 20 years of experience with the USAF, where he served on several assignments in both the operations and the research and development of C3 systems. Dr. Wagenhals's research interests include the design and evaluation of C3 operational and system architectures and the integration of Bayesian network and discrete event system modeling technologies to support the temporal evaluation of courses of action in complex geopolitical/military situations.