Architectural Principles for Enterprise Frameworks: Guidance for Interoperability

Richard A. Martin[1], Edward L. Robertson[2], John A. Springer[3]
1 Tinwisle Corporation Email: tinwisle@bloomington.in.us
2 Computer Science Department, Indiana University Email: rbstn@cs.indiana.edu
3 Computer Science Department, Indiana University Email: jospring@cs.indiana.edu

Abstract. This paper presents a number of principles related to the construction and use of enterprise architecture frameworks. These principles are intended to guide the development of a formal foundation for frameworks but also serve as guidance for efforts to enable the interoperability of enterprise models and model components. The principles are drawn from analyses of a number of existing frameworks and from observation of and participation in framework development.

1. INTRODUCTION

An *enterprise architecture framework* is a means to understand an enterprise or class of enterprises by organizing and presenting artifacts that conceptualize and describe the enterprise. An *enterprise*[1] is a collective activity in a particular domain, with actors sharing a common purpose; an enterprise can be a business, a collection of businesses with a common market, a government agency, etc. *Architecture* is a metaphor to the realm of office towers and bridges, intended to capture the use-oriented, as opposed to construction-oriented, aspects of the design of those structures. A *framework* is a structured container for holding and interconnecting things[2] – in the remainder of this document those things are *artifacts* that comprise the enterprise architecture. In framework contexts, artifacts are almost always models of some kind, which we sometimes call "components" to indicate that they are pieces of the entire framework. These artifacts are conceptual, logical, and physical representations at all levels of the enterprise and range from simple lists through elaborate data models, tools supporting methodologies, and operating procedures. In the following, "framework" will always be shorthand for "enterprise architecture framework".

[1] The word "organization" is a common synonym for enterprise, but we must often use "organization" to denote the way things are organized and thus restrict it to that use.
[2] As another metaphor, think of a framework for electronic components which both holds circuit boards and provides for wiring between those boards.

Frameworks have been widely used. The Information Technology Management Reform Act of 1997 led to the U.S. Government's Federal Enterprise Architecture Framework (FEAF), which "describes an approach, including models and definitions, for developing and documenting architecture descriptions" [U.S. GAO, 2003]. It is being deployed in all non-military agencies of the U.S. Government. The annual ZIFA Forums [ZIFA, 2004] have included nearly 100 case studies highlighting the benefits of frameworks. Bernus et al. [Bernus, 2003] give several thorough case studies (along with an extensive discussion of enterprise architecture issues). Whether the frameworks address manufacturing operations, process control, information systems, or government bureaucracy, the artifacts produced to describe the enterprise comprise a valuable asset requiring its own distinct management. Managing and gaining full value from that asset is the reason enterprise architecture frameworks are conceived, built, and used.

Professional practice has taught us about the fragility of isolated application silos on islands of automation and about the difficulty in achieving interoperability under such circumstances. While these are typically called "data silos," the significant problem is that they are in fact model silos. That is, the mismatch of underlying models is the greatest impediment to integration and interoperability.

In spite of their wide use and importance, frameworks have all been defined only descriptively. This means that it is currently impossible to formally relate different frameworks, to say nothing of implementing tools that properly support these frameworks.[3]

This work is about frameworks in general and not about any one particular framework. Although our original motivation was the Zachman Framework for Enterprise Architecture [Zachman, 1987, ZIFA, 2004], we examined and incorporated several other frameworks, which are itemized in Section 2. Moreover, this work is about structure and not about contents. Thus "framework" by itself indicates a collection of descriptions and principles for organizing framework contents while "framework instance" indicates the use of a framework describing one particular enterprise.

The primary goal of this paper is to identify the guidance for interoperability that the principles elicit. Such guidance follows from the understanding of frameworks and framework formalization that led us to the use of frameworks to support organization and interaction of the many models associated with an enterprise. This work continues our effort to formalize the ways in which these particular frameworks manifest the architecture of an enterprise [Martin, 1999], with an eye toward (i) connecting a framework instance's contents, (ii) manipulating those contents and connections, and hence (iii) relating different frameworks and recasting instances from one framework standard to another. While our primary motivation for developing these principles is to use them to guide our formalization activities, we

[3] There are software packages that purport to implement various frameworks, but these packages only implement the "holding" aspect of frameworks. That is, they are tools for editing and managing representations which populate a framework instance, without respect to the semantics that the framework provides.

believe that many are directly useful in the development of individual frameworks and for enabling interoperability among framework instances.

Section 2 begins this paper with a discussion of the origin and (to the extent possible) validation of the principles. Section 3 introduces a few principles that are general in nature, applicable to any modelling and analysis endeavour,[4] while Section 4 discusses principles especially pertinent to frameworks. We then conclude this document by considering how these principles guide the formalization of frameworks and efforts to enable interoperability.

2. Origins of the Principles

The principles described below come from (i) evaluation and comparison of different frameworks, (ii) observation of the process of defining frameworks, and (iii) participation in this same process.

Principles are largely based on analysis of the framework architectures: Zachman [Zachman, 1999], an ISO draft standard titled Enterprise Integration - Framework for Enterprise Modelling [ISO 19439, 2004], ISO Standard 15288 Information Technology – Life Cycle Management - System Life Cycle Processes [ISO/IEC 15288, 2002], and the U.S. Department of Defense C4ISR Architecture Framework [US DoD, 1997], an analysis which we reported in [Martin, 2002, Martin, 2003].[5]

Principles are also based on professional observation and participation -- often experience of the difficulties which arise when these principles are not followed. Meeting minutes from ISO efforts illustrate such difficulties, as in the statement "Something is not very clear the distinction between the interoperability of process models and the interoperability of processes" [WG1, 2003], which reflects principle 3.4 about meta-levels. Our own professional experience includes constructing and analyzing models in an enterprise context, teaching modeling, and participating in the development of international standards for enterprise architectures.[6]

We do not claim to have originated all these principles. Several are simply our statements of well-established suggestions (e.g. 3.6, "Do not hide architecture in methodology", which is a rephrasing of the data independence principle [Date, 1981]). Principles reflecting some of the same concerns as ours have been identified elsewhere [Greenspan, 1994, ISO TR 9007, 1987, Totland, 1997], although these other principles are largely directed at ensuring the fidelity of the modelling process.

Occasionally specific facts are given in evidence. Only a few principles can be supported so concisely. One such principle 4.6, that states the independence of three commonly correlated scales, is supported by examples high in one scale but low in another. Unfortunately, principles that describe general behaviour do not admit such concise support. This is very loosely similar to the difference between existential and universal propositions, in that one instance proves the former.

---

[4] We are still using "framework" as shorthand for "enterprise architecture framework", but it would be a valuable exercise to see which of these principles hold for other classes of frameworks.

[5] Space limitations make it impossible to repeat that analysis here.

[6] Richard Martin is convener of TC 184/SC 5/WG 1, "Modeling and architecture", of the International Standards Organization.

Perhaps the most insightful principle is principle 4.4, which recognizes that analytical partitioning uses both grids and trees. We first observed this duality in the context of adding detail within a Zachman framework [Inmon, 1997], necessitating the use of recursion within a frame. This principle has been validated by its use in comparing frameworks [Martin, 2003] and its value in the development of international standards [Bernus, 1996], particularly ISO 15704:2000 Industrial Automation Systems Requirements for Enterprise Reference Architecture and Methodology[ISO 15704, 2000].

Many principles focus on highlighting and refining distinctions (such as principle 3.5, which distinguishes dependency and temporal order). They arise from observation of the ways in which people model, and the successes and the difficulties encountered therein.

Principles may be descriptive, describing the way that model artifacts are constructed and organized, or prescriptive, recommending how they should be. However, prescriptive principles all began as observations of the form "People have trouble with …". Prescriptive principles of course guide practice; but they also guide the formalization effort, indicating what should be facilitated or discouraged.

3. General Principles of Modelling

Modeling as we mean it is a conceptual exercise, only analogously related to physical modeling as in, say, model railroads.[7] Conceptual modeling does yield representations in a particular medium, not necessarily a medium with physical manifestations, but these are representations of the modeled concepts. Thus principles apply to both concepts and representations.

Each of the following principles begins with a short phrase (indicated in that manner) which identifies and hopefully summarizes the principle. More extensive discussion of the respective principles, including evidence for them, is given in our EMMSAD04 paper or technical report [Martin, 2004a, Martin, 2004b]. Much of this paper originates in those works as well.

3.1 Communication is a goal of modeling.

Models (including frameworks) are formal artifacts but they are developed and used by people. Therefore any modeling formalism must be robust and tractable in interaction with non-formal components – people. This principle is discussed at great length in [Totland, 1997] and related psychological factors are discussed in [Siau, 1999].

3.2 Complexity tradeoff.

There is typically a tradeoff between complexity in the modeling medium and complexity in model instances constructed using that medium. Modeling

[7] We draw this distinction because, for most people, the first connotation of "make a model" is to construct a model railroad or something similar. Model railroads diminish function but primarily reduce physical scale; indeed, the first descriptor applied to a model railroad is its "gauge", or physical scale.

mechanisms therefore should be defined with an attempt to find a "sweet spot" where these complexities are in balance.

3.3 Naming matters.

Naming, i.e. the assignment of a string[8] to a concept or artifact, serves as the bridge between formal artifacts and human interpretation. That is, there are two sides to naming: "external" (relating to the real world) and "internal" (relating to the mechanism and models of a framework). Said another way, internal naming involves formal meaning while external naming involves human understanding of that meaning.

Both sides of this principle impact interoperability. Internally, interoperable components must interpret names consistently across the interaction, hence the emerging emphasis on formal ontological methods to resolve semantic consistency. Externally, human mediated interoperability depends upon the correct assignment of actions to messages received and the creation of messages that convey the intended semantics to the receiver. Whereas the internal context should be well defined, the external context is often ambiguous.

3.4 Use "meta" with great care, because the term is seriously overloaded.

This particularly applies when discussing meta-levels. This is particularly true because "meta" is a relative term, not an absolute.

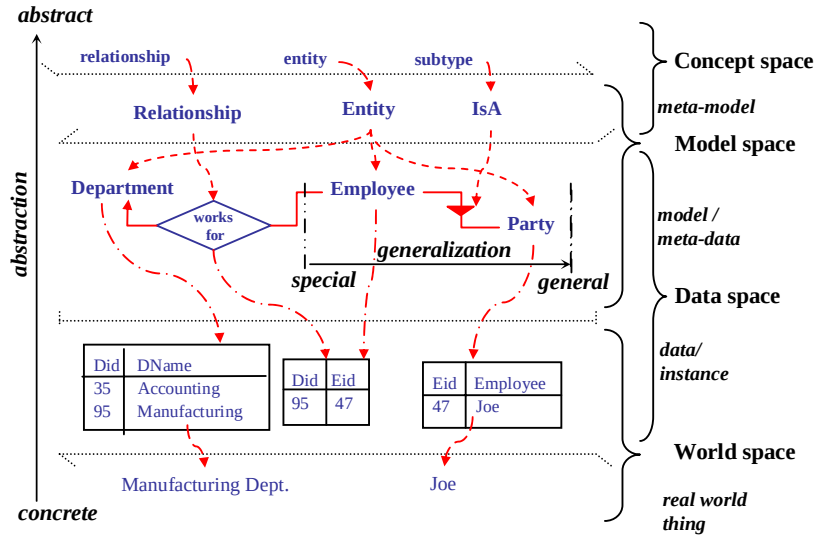

Figure 1 Relative meta-

[8] We do not use "label" because we want to restrict that term to a specific use.

One obvious example of the relativeness of "meta" is observable in the realm of ER modeling. There, the meta-model level decomposes all models into Entities and Relationships; the model level may decompose a particular model for corporations into Department, Employee, Project (instances of Entity), Works For (instance of Relationship), etc.; the model population level (for a fixed corporation) into Sales, Human Resources, Accounting, etc. (instances of Department). Thus the model level is meta with respect to the model population and ER notation is the meta-meta level for the model population. Notice that "instance" is also a relative term, in that it does not indicate an absolute level but only the level below X when used in the phrase "instance of X". Also, "meta" is roughly the inverse of "instance of", in that the meta of an instance of X is in fact X . However, since our interest focuses on models and meta-models, henceforth "instance" shall denote artifacts at the model level; that is, Department, Employee, Works for, etc in the above example.

3.5 Dependency is not chronology.

That is, just because B depends upon A, it is not necessary that B follows A in time. While much of the evidence for this principle comes out of difficulties arising when it is not followed, ISO 14258 Industrial automation systems – Concepts and rules for enterprise models, makes this distinction explicit [ISO 14258, 1998].

3.6 Do not hide architecture in methodology.

It is wrong to bury characterizations of things in methods that are used to construct them. This is not to claim that methods do not constrain results (to claim so would be most foolish) but rather to observe that such constraints must be made explicit and external to the construction process. In particular, the architectural form should survive changes in method and technology. Thus the link between architectural form and interoperability is very strong. Robust interoperability should also survive changes in method and technology.

4. Principles Specific to Frameworks

4.1 Frameworks organize artifacts.

A framework is a means to facilitate understanding of enterprises and to communicate that understanding, principally by organizing and connecting artifacts used to represent a particular enterprise. Frameworks help us to take very richly textured descriptive and prescriptive artifacts and arrange them for practical understanding. Frameworks help to simplify complex artifact collections that are composed of many inter-related components. The organizational mechanism of a framework is primarily a collection of *dimensions* along which the artifacts are placed and hence classified. It is in the number and different natures of these dimensions that frameworks vary. Many further principles relate to the characterization of these dimensions.

4.2 Distinguish structure from connectivity.

Structure and connectivity are distinct aspects of frameworks[9] and a framework formalization (or standard) should distinguish them. The clarity of this distinction directly impacts the quality of a framework; unfortunately many frameworks do not achieve their intended impacts because they do not exhibit this distinction with sufficient clarity. Furthermore, useful reorganizations of a framework, one of many viewing mechanisms, can be tractably expressed when phrased in structural terms, whereas desired views involving connections may be difficult to specify and expensive to compute.

4.3 Separate policy from mechanism.

That is, policy should be found in framework contents and not framework structure.

4.4 Two aspects of organization.

There are two general ways in which items within a framework are (typically) arranged: (i) in an *ordinant* structure (that is, a table, grid, or matrix) or (ii) in a *decompositional* structure (that is, a tree). We call either of these dimensions of the arrangement. Dimensions of either kind are discrete[10] and ordinant dimensions typically have only a few coordinate positions. The coordinate positions of an ordinant dimension may be ordered (e.g. rank) or unordered (e.g. gender), while a decompositional dimension is always ordered only by its containment relation.

An important step in organizing artifacts is to identify and characterize (as ordinant or decompositional) the dimensions that define the structure. The definition of an ordinant dimension is the identification of its coordinates and, where relevant, the order of those coordinates. Recall that dimensions only describe the placement of items (in a real or conceptual space) and not the interconnection of these items, which is typically much richer and more complex.

Given this distinction in structural arrangement and the two principles that follow, it seems critical that structural alignment be essential for interoperability. Context is a structural characteristic of frameworks and the semantic interpretation of content is highly dependent upon context.

[9] We find it helpful to visualize a computer room where frames both hold devices (servers, disk drives, communications interfaces, etc.) and provide channels for wiring these devices together. A second metaphor is between bone (structure) and muscle (connection); this emphasizes that operation largely occurs through the connections.

[10] This statement necessarily holds for decompositional dimensions but is sometimes relevant to distinguish meta-coordinates from instance coordinates where ordinant dimensions are involved.

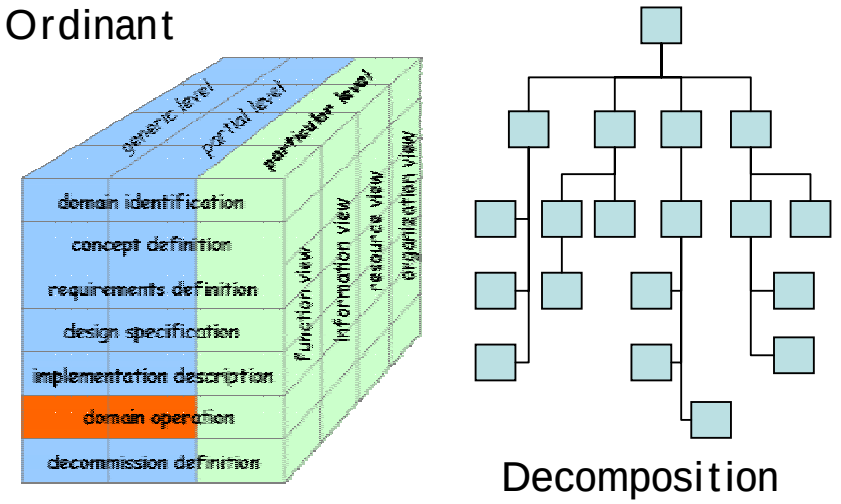

Figure 2 Two aspects of organization

4.5 Decomposition may occur at many meta-levels.

That is, it is natural and expected that there be meta-level and model-level decompositions (from whatever perspective "meta" is considered). For example, saying that the <conceptual; what> cell of a Zachman frame contains Entities and Relationships is a meta-level decomposition of that cell, while saying that Employee and Department are Entities is a model-level decomposition.

4.6 Three aspects of scale.

There are (at least) three distinct dimensions that reflect conceptual (as opposed to physical) scale: (i) abstractness, ranging from abstract to concrete, (ii) scope, from general (generic) to special (specific), and (iii) refinement, from coarse to fine. Using the terminology of principle 4.4, abstractness, and scope are ordinant-ordered and refinement is decompositional.[11]

Because it is common to have co-occurrence of the origin or extreme endpoints in all three dimensions (as a module that is concrete, specific, and finely refined), these three dimensions are often confused. Understanding (and distinguishing) conceptual scales is essential because they govern the ways in which framework dimensions are conceived, ordered, populated, and constrained.

4.7 One dimension manifests purpose within a framework.

One, and typically only one, of a framework's ordinant-ordered dimensions reflects the purposive nature expressed within a framework. Note that such a "purposive

[11] In fact, refinement is often the canonical hierarchy.

dimension" does not represent the purpose of the framework but instead represents the fact that artifacts derive their purpose from artifacts earlier in the dimension's order (most often through elaboration). Derived dimensions, produced through views (see principle 4.11 below), may also exhibit a purposive order; the C4ISR's "Force Integration" dimension, derived from a command-structure hierarchy, exhibits the purpose inherent in any chain of command.

The ordering of a purposive dimension often manifests itself as causality, dependency, or chronology. However, it is not merely a time dimension, even though purpose in a framework often leads to temporal ordering in the operations of the enterprise. This indeed follows from general principle 3.5.

4.8 Refinement is recursive.

The decompositional scale dimension, refinement, is fundamentally different in that it works (or at least works best) through decomposition and successive refinement. Thus frameworks should be recursive in their application. Unfortunately, practice often foreshortens the recursion, forcing a fixed (albeit hierarchical) or flattened structure.

Recursion also has an impact on contextual alignment for interoperability. Erroneous assumption of recursive level during interactions is as destructive to automation outcomes as it is to human mediated activities.

4.9 All context is relevant.

It seems necessary, as one moves through a framework along its purposive dimension, from row to row in a Zachman framework for example, that the entire framework structure at one row is *potentially* relevant when describing a component at the next. This is not to claim that an entire row is in fact materially relevant for each component in the next; it is merely recognition that all of the models from prior coordinates can be useful in understanding and constructing the next. Moreover, it is sometimes as important to know which concerns are not needed as it is to know which are. Perhaps this principle just reflects the fact that frameworks, and the enterprise domains with which they are concerned, are not suitable for minimally descriptive artifacts.

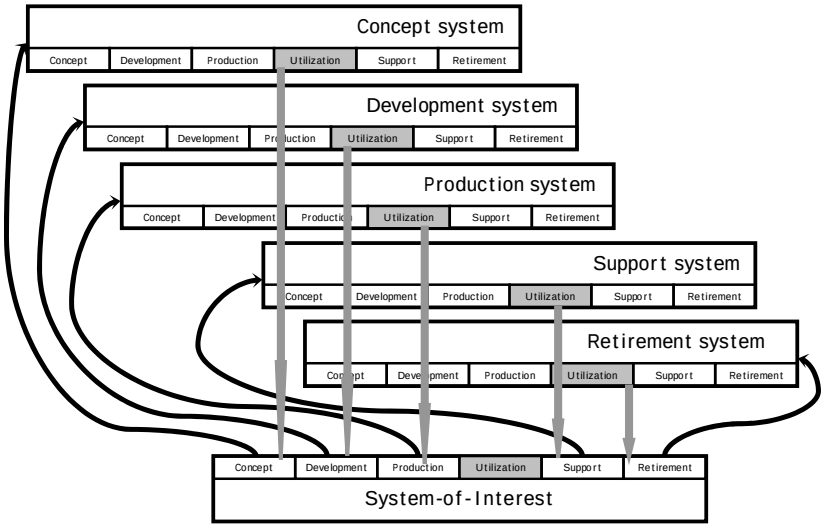

Figure 3 Recursion in ISO 15288 cf.

4.10 Connections can be of arbitrary arity.

Connections between framework artifacts can be of arbitrary arity, although binary ones are most common. However, it is sufficient to provide for the construction of arbitrary connections using binary ones. For example, a Relationship in an ER model may be constructed to have any degree, but the basic connections are always between a single Entity and a single Relationship.

4.11 Views are important in standards and methodologies.

A framework formalism should provide a general mechanism for defining views. Views are used in enterprise modeling because the complexity of an enterprise makes it impossible for a single descriptive representation to be humanly comprehensible in its entirety. The notion of view is inherent in any large, complex structure observed and managed by many individuals who neither can nor should attempt to analyze, design, or implement the entire structure.

The view mechanism should be general and dynamic. It must be general because there is little commonality of particular views across frameworks. It must be dynamic both because new views arise as standards are extended and because ad hoc views are requested. Just as view mechanisms provide the content for integration, so too will systems and components become interoperable through view mechanisms. To act on behalf of another seems to require some mechanism for perception that goes beyond simple enumeration of content.

4.12 Construction through views.

Views are not merely used for viewing; they are often used for constructing and populating frameworks.

4.13 Constraint mechanisms are necessary.

Framework standardization, as currently practiced, augments the frameworks themselves with voluminous texts constraining how frameworks are to be constructed or applied. In spite of considerable effort, such texts are inconsistent, ambiguous, and difficult to apply. Framework formalization should provide a foundation upon which unambiguous, concise, and effectively computable constraint mechanisms can and should be built.

4.14 Constraints may occur at various meta-levels.

This is a natural partner to principle 4.5 and the same example applies. Within the <conceptual; what> cell, the constraint that Entities only connect to Relationships is "meta" with respect to "cardinality constraints", such as requiring that an individual Employee works in one Department.

The above principles characterize many of the frameworks that are concerned with domains at the enterprise level, although we have found no framework that exhibits all of these principles. Collectively, these principles constitute the foundation upon which useful enterprise frameworks are constructed.

5. Toward Framework Formalizations

While the previous sections discussed principles obtained from observation and analysis of existing frameworks, this section outlines how these principles guide formalizing enterprise frameworks. Although the individual framework instance is of course the formalized artifact, the following discussion is directed toward "architectural" standards that prescribe how a collection of frameworks is to be formalized.

There are four major aspects of a formalism that follow from the above principles. We itemize these four and justify why they should be treated distinctly. The long version of this paper then delves more deeply into these four aspects [Martin, 2004b].

- **structure**: the way that components and sub-components of an enterprise are placed within a framework. Principles 4.4 – 4.8 guide the elaboration of this aspect.
- **connections**: the manner in which components and sub-components of an enterprise are interconnected within a framework. It is through these connections that the operations of an enterprise are manifest.
- **views**: formal mechanisms for restructuring a framework to emphasize features from a particular conceptual or operational perspective.

- **constraints**: formal mechanisms by which the conformance of a particular instance to a standard or architecture may be evaluated.

The deliberate separation of structure and connections is a direct consequence of principle 4.2. A framework is thus a structure for holding artifacts and a mechanism for connecting them.

The needs for views and constraints are enunciated in principles 4.11 and 4.13 respectively. While it is necessary to draw distinctions between structure and connections, it is advantageous to do the opposite, drawing parallels between views and constraints. In particular, the ability to define views immediately enables constraints definable in terms of views, as in "view A is a subset of view B".

A formalism for framework structures provides the foundation upon which formalizable, and therefore precise and coherent, view mechanisms can be built; and, conversely, view mechanisms provide the formalism through which one single overarching structure is coherently and consistently created by these many individuals.

6. Conclusion

We have identified 20 principles about the ways in which enterprise frameworks are or should be constructed and used, but this is only one step on a longer path. These principles will guide the formalization of frameworks, as discussed in section 5, but we are early in the work of that formalization. It is evident that the structure of a framework is carried by a tree whose nodes have a tabular, dimensional form, but many details governing the expression of structure and the interaction of this expression with connections, views, and constraints are yet unknown. Because existing frameworks do not treat connections in a disciplined manner, there is less guidance concerning connections from existing practice.

Interoperability, that is the automatic operation of agents from one enterprise in the context of a second, can be facilitated through enterprise framework principles in two ways. The first, and by far the preferable, way is to enable the automated agent to "understand" the second enterprise's context. Unfortunately, the mere presence of frameworks does not guarantee this. The second, and always available, way is that the frameworks facilitate true human understand even if such understanding is not immediately automatable. That is, a variety of implications of the above principles, such as having model artifacts specifically identified through frameworks, knowing the dimensional structure of frameworks, and having constraints articulated, facilitates human specification of the integration that is a precursor to interoperability.

Because these are principles, we expect situation specific exceptions. Models of every kind are most often incomplete and imprecise representations expressed using available tools and media. To the extent that these principles guide a better understanding of the structure, connections, views and constraints embodied in a modern enterprise, they can add precision and completeness to the expression of that

enterprise. And finally, it is important that the formalization attempts to reach "sweet spots", as discussed in principle 3.2.

In as much as the principles enunciated herein are the core of a "requirement specification" for analysis and formalization of enterprise frameworks, we welcome all suggestions and comments.

7. REFERENCES

Bernus P, Nemes L, Williams T J, editors (1996) Architectures for Enterprise Integration, Chapman and Hall, London

Bernus P., Nemes L., & Schmidt G., editors (2003) Handbook on Enterprise Architecture. Springer Verlag, Berlin

Date, C. J. (1981) *An Introduction to Database Systems*. Addison-Wesley

Greenspan S. J., Mylopoulos J., & Borgida A. On formal requirements modeling languages: RML revisited. In *International Conference on Software Engineering*, pages 135--147, 1994.

Inmon W., Zachman J., & Geiger J. (1997) *Data Stores, Data Warehousing, and the Zachman Framework*. McGraw-Hill

ISO 14258 (1998) Industrial Automation Systems – Concepts and rules for enterprise models, International Oragization for Standards, Geneva.

ISO 15704 (2000) Industrial Automation Systems – Requirements for Enterprise Reference Architecture and Methodology, International Organization for Standardization, Geneva

ISO 19439 (2004) FDIS Enterprise Integration – Framework for Enterprise Modelling, International Organization for Standardization, Geneva.

ISO/IEC 15288 (2002) Information Technology - Life Cycle Management - System Life Cycle Processes. International Organization for Standardization and International Electrotechnical Commission, Geneva

ISO TR 9007 (1987) Concepts and Terminology for the Conceptual Schema. International Organization for Standardization, Geneva, No long available through www.iso.ch.

Martin R. & Robertson E. (1999) Formalization of multi-level Zachman frameworks. Technical Report 522, Computer Science Dept., Indiana Univ., 1999. www.cs.indiana.edu/ftp/techreports/TR522.html.

Martin R. & Robertson E. (2002) Frameworks: Comparison and correspondence for three archetypes. In ZIFA 2002 Enterprise Architecture Forum, 2002.

Martin R. & Robertson E. (2003) A comparison of frameworks for enterprise architecture modeling. In ER2003 22nd Intl. Conf. on Conceptual Modeling, pages 562--564.

Martin R. A., Robertson E. L., & Springer, J. A. (2004a) Architectural principles for enterprise frameworks. In *CAiSE Workshops Knowledge and Model Driven Information Systems Engineering for Networked Organizations*, Janis Grundspenkis & Marite Kirikova (Eds.), Riga Technical University, Lativa

Martin R. A., Robertson E. L., & Springer J. A. (2004b) Architectural principles for enterprise frameworks. Technical report, Computer Science Dept., Indiana Univ., www.cs.indiana.edu/ftp/techreports/TR594.html.

Siau K. (1999) Information modeling and method engineering: A psychological perspective. *J. of Database Systems*, 10(4):44--50.

Totland T. (1997) *Enterprise Modeling as a Means to Support Human Sense-making and Communication in Organizations*. PhD thesis, Norwegian University of Science and Technology, Department of Computer and Information Science.

U.S. Department of Defense - Architecture Working Group (1997) *Command, Control, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework, Version 2.0*

U.S. General Accounting Office (2003) GAO-03-584g Information Technology: A framework for assessing and improving enterprise architecture management. Washington, D.C.

WG1 (2003) Meeting Minutes St. Denis. International Organization for Standardization TC 184, SC 5, WG1, available at forums.nema.org/~iso tc184 sc5 wg1.

Zachman J. A. (1987) A framework for information systems architecture. IBM Systems Journal, 26(3), 1987.
ZIFA (2004) The Zachman Framework. Zachman Institute for Framework Advancement. Various pages at www.zifa.com; the “Quickstart” is particularly relevant.