Architectural Principles for Enterprise Frameworks

Richard A. Martin[1], Edward L. Robertson[2], and John A. Springer[2]

[1] Tinwisle Corp., 205 N College Ave., Bloomington IN 47404
[2] Computer Science Dept., Indiana University, Bloomington IN 47405
Supported by NSF grant ISS-82407.

Abstract. This paper continues our work on the analysis and formalization of enterprise architecture frameworks, proposing a number of principles related to the construction and use of these frameworks. These principles are intended to guide the development of a formal foundation for frameworks. Enterprise architecture frameworks organize, manage, and interrelate the wide variety of models used to structure and operate an enterprise. The principles are drawn from analyses of a number of existing frameworks and from observation of and participation in framework development. Since these frameworks involve modeling, some of the principles apply to broader aspects of modeling; other principles apply only to frameworks. As the goal of this work is a requirements specification for formalization of frameworks, the paper ends with a sketch of how the identified principles might guide this formalization.

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*[3] 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[4] – 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. In the following, "framework" will always be a shorthand for "enterprise architecture framework".

[3] 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.
[4] 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 US Government's Federal Enterprise Architecture Framework (FEAF), which "describes an approach, including models and definitions, for developing and documenting architecture descriptions"[14]. It is being deployed in all non-military agencies of the US Government. The annual ZIFA Forums[19, register as a "friend"] have included nearly 100 case studies highlighting the benefits of frameworks. Bernus *et al.*[2] 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 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.[5] This paper works toward correcting that deficiency, as part of a larger project which seeks to characterize and formalize frameworks.

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[18,19], we examined and incorporated several other frameworks, which are itemized in Sect. 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 develop a set of principles to guide any effort to understand and formalize the use of frameworks to support organization and interaction of the many models associated with an enterprise. This work therefore continues our effort to formalize the ways in which these particular frameworks manifest the architecture of an enterprise[10], 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 goal in developing these principles is to use them to guide our formalization activities, we hope that many are directly useful in the development of individual frameworks.

[5] 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.

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 modeling and analysis endeavor,[6] while Sect. 4 discusses principles especially pertinent to frameworks. We then conclude this document by considering how these principles guide the formalization of frameworks.

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[19], a revision to the European pre-standard ENV 40003:1990 *Computer Integrated Manufacturing: Systems Architecture Framework for Modeling*[16], ISO Standard 15288 *Systems engineering – System life cycle processes*[7], and the US Defense Department's C4ISR Architecture Framework[4], an analysis which we reported in [11].[7]

Principles are also based on professional observation and participation – often experience of the difficulties which arise when these principles are not followed. Draft working notes 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" [9], which reflects principle 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.[8]

We do not claim to have originated all these principles. Several are simply our statements of well-established suggestions (*e.g.* 6, "Do not hide architecture in methodology", which is a rephrasing of the data independence principle[3]) Principles reflecting some of the same concerns as ours have been identified elsewhere [5,8,17], although these other principles are largely directed at insuring the fidelity of the modeling process; intersections with our principles will be mentioned as they occur. We include them all because we intend this compendium as a basis for the formalization that we will briefly sketch in Sect. 5.

Occasionally specific facts are given in evidence. Only a few principles can be supported so concisely. One such principle (11), that states the independence of three commonly correlated scales, is supported by examples high in one scale

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

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

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

but low in another. Unfortunately, principles that describe general behavior 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.

Perhaps the most insightful principle is principle 10, which recognizes that decomposition uses both grids and trees. We first observed this duality in the context of adding detail within a Zachman framework[6], necessitating the use of recursion within a frame. This principle has been validated by its use in comparing frameworks[11] and its value in the development of international standards, particularly ISO 15704:2000 [7].

Many principles focus on highlighting and refining distinctions (such as principle 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 Modeling

Modeling as we mean it is a conceptual exercise, only analogously related to physical modeling as in, say, model railroads.[9] 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.

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 [17] and related psychological factors are discussed in [15].

2 Complexity tradeoff. There is typically a tradeoff between complexity in the modeling medium and complexity in model instances constructed using that medium. That is, if the underlying mechanism is too simple, then instances become complex to compensate; if the mechanism is too complex, it becomes the plaything of a very few specialists. Modeling mechanisms

[9] 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.

therefore should be defined with an attempt to find a "sweet spot" where these complexities are in balance. The success of Entity-Relationship (ER) models is attributable to this balance.[1] Of course we must remember that different modeling efforts have different sweet spots. The Unified Modeling Language[13] is an aggregation of several modeling mechanisms, each of which seeks to establish a "sweet spot" with respect to the representational needs of the content being modeled.

3 Naming matters. Naming, *i.e.* the assignment of a string[10] 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.
Because names serve a role in human communication as well as one related to formal structure, naming must be done with great care. Of course the formalism works equally well whether the names used are well understood by human participants or are merely nonsense terms, as long as meaning is unambiguous (This fact is quite beneficial, since it allows the focus to be on the human/ontological aspects of name choice). Naming has important (and sometimes unexpected) consequences because that name typically has other associations. Even professionals often use the same names with different meanings. Thus, the development of ontologies and ontological methods to manage naming is complementary to the study of formalisms for framework expression.

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.
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 $\mathcal{X}$ when used in the phrase "instance of $\mathcal{X}$". Also, "meta" is roughly the inverse of "instance of", in that the meta of an instance of $\mathcal{X}$ is in fact $\mathcal{X}$. 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.

5 Dependency is not chronology. That is, just because $\mathcal{B}$ depends upon $\mathcal{A}$, it is not necessary that $\mathcal{B}$ follows $\mathcal{A}$ in time. For example, there is a dependency

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

in the general activities of SALES ⇝ SHIPPING ⇝ PRODUCTION, in that the purpose of SHIPPING is to fulfill SALES and the purpose of PRODUCTION is to enable SHIPPING. While an individual sale (an operation deriving from SALES) is followed by a shipment, in an ongoing enterprise (where sales occur over a long time) it is not the case that SALES as a unit precedes SHIPPING. Moreover, an operation of PRODUCTION must occur before the corresponding SHIPPING and might even occur before the corresponding SALES event, even though the purpose of the first followed from the second. Indeed, this aspect of anticipation - separating timing from dependency - is a central reason for enterprise modeling (whether formal or informal).

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 (indeed it would be most foolish to claim so) but rather to observe that these constraints must be made explicit and external to the construction process. In particular, the architectural form should survive changes in method and technology.

4 Principles Specific to Frameworks

7 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 artifacts and arrange them for practical understanding. Frameworks help to simplify complex presentations which are composed of many inter-related artifacts. 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.

8 Distinguish structure from connectivity. Structure and connectivity are distinct aspects of frameworks[11] 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 (discussed below in terms of view definitions) can be tractably expressed when phrased in structural terms, whereas desired views involving connections may be difficult to specify and expensive to compute.

Modeling that confounds structure and relationship is prone to both inaccuracy and brittleness. Thus we strongly recommend that the models used

[11] 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.

within a framework exhibit the same distinction; fortunately many common models do, including entity–relationship, process–flow, personnel–reporting line in an organization chart.

9 Separate policy from mechanism. That is, policy should be found in framework contents and not framework structure. As their goal is to facilitate understanding, frameworks provide (structural and connective) mechanism rather than delineate policy concerning enterprise management. Within an instance of a framework, representation of such understanding may constrain the operation of a *particular* enterprise and that framework may of course define policies. This parallels a distinction between mechanism and policy that was popularized in the conceptualization of computer operating systems.

10 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[12] 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.

Individual artifacts, in turn, are identified within a framework by name. A name can indicate a coordinate position along a particular ordinant dimension or indicate one member of a collection. When a string is used as a name in one of these contexts, we will refer to it as a *label*. Such a label has meaning fixed by formalism within a formal context; but when viewed in isolation that fixed meaning may be lost.

11 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 10, abstractness, and scope are ordinant-ordered and refinement is decompositional.[13]

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

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

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

finely refined), these three dimensions are often confused. But they are in fact independent. Examples validating this independence occur in: (*i*) ER models (fully developed, with all attributes, relationship constraints, *etc*), which are both abstract and finely refined; (*ii*) ISO 19439, where the "Generic" plane includes items across the range from abstract to concrete; and (*iii*) C4ISR, where technically detailed products span a range of operational abstraction. Understanding (and distinguishing) conceptual scales is essential because they govern the ways in which framework dimensions are conceived, ordered, populated, and constrained.

The fourth principle of Greenspan *et al.*[5, §2] focuses on abstraction and refinement, although without a firm distinction between the two.

The following principles seem less likely to guide practice than those itemized above. That is, they are more purely specifications for our intended formalism development.

12 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 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). Examples of such purposive dimension are *Role* in the Zachman framework, *Model Phase* in ISO 19439, *Process Group* in ISO 15288, and *Guidance* in C4ISR. Derived dimensions, produced through views (see principle 16 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 5.

13 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.

A major benefit of recursion in framework structure is that it directly supports a "drill-down" approach to framework development and exploration. To manage and comprehend the richness present in a framework, it is necessary to separate the artifacts such that detail is hidden until revealed for consideration. Recursion is the mechanism for providing this layered approach. Furthermore, recursion greatly facilitates building one unified framework out of several here-to-fore independent ones.

14 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.

15 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.

16 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. Although particular framework standards have often been defined with fixed, predefined views intended to usefully "package" subsets of their complex space,[14] we have seen increasing awareness that views must be dynamically definable. Furthermore, views can be quite simple or very elaborate depending upon the intended use. The view mechanism should facilitate dynamic extraction and restructuring of an enterprise model from various conceptual perspectives.

17 Construction through views. Views are not merely used for viewing; they are often used for constructing and populating frameworks. For example, entities are placed in the ER model through the "information view" rather than into the complete framework. Thus the "view update" problem from the world of relational database reappears in the context of frameworks.

18 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. Such application is of course limited by the degree to which constraints fall wholly within a framework, since a constraint that is even partially outside of the framework is not enforceable within the framework. Framework formalization should provide a foundation upon which unambiguous, concise, and effectively computable constraint mechanisms can and should be built.

Beyond the simple observations that informal constraints exist and formal ones are highly desirable, at this point we can draw no further principles concerning constraints. Because current frameworks are largely structural,

[14] Such views are often described as if they comprise a distinct dimension, but such a collection of views is an artifact of the process rather than part of the underlying framework.

the constraints we observe are also structural. We do caution that constraints are also subject to considerations about meta-levels – in particular, model constraints must be distinguished from instance constraints.

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[12].

structure: the way that components and sub-components of an enterprise are placed within a framework. Principles 10 – 13 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 8. 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 16 and 18 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 18 principles about the ways in which 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.

Finally, it is important that the formalization attempts to reach "sweet spots", as discussed in principle 2.

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

References

1 C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design: An Entity-Relationship Approach.* Benjamin/Cummings, Redwood City, CA, 1992.

2 Peter Bernus, Laszlo Nemes, and Gunter Schmidt, editors. *Handbook on Enterprise Architecture.* Springer Verlag, 2003.

3 C. J. Date. *An Introduction to Database Systems.* Addison-Wesley, 1981 ff.

4 Department of Defense – Architecure Working Group. C4ISR Architecture Framework, Version 2.0, 1997.

5 Sol J. Greenspan, John Mylopoulos, and Alexander Borgida. On formal requirements modeling languages: RML revisited. In *International Conference on Software Engineering*, pages 135–147, 1994.

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

7 International Organization for Standardization. All ISO documents are available from `www.iso.ch`.

8 International Organization for Standardization. Concepts and terminology for the conceptual schema, 1987. No long available through [7].

9 International Organization for Standardization TC 184, SC 5, WG1. N450 Meeting Minutes, St. Denis, France, Nov. 2003. `forums.nema.org/~iso_tc184_sc5_wg1`.

10 Richard Martin and Edward Robertson. Formalization of multi-level Zachman frameworks. Technical report, Computer Science Dept., Indiana Univ., 1999. `www.cs.indiana.edu/ftp/techreports/TR522.html`.

11 Richard Martin and Edward Robertson. A comparison of frameworks for enterprise architecture modeling. In *ER2003 - 22nd Intl. Conf. on Conceptual Modeling*, pages 562–564, 2003.

12 Richard A. Martin, Edward L. Robertson, and John A. Springer. Architectural principles for enterprise frameworks. Technical report, Computer Science Dept., Indiana Univ., 2004. `www.cs.indiana.edu/ftp/techreports/TR594.html`.

13 Object Management Group. Unified Modeling Language. `www.uml.org`.

14 U. S. General Accounting Office. Information technology: A framework for assessing and improving enterprise architecture management, 2003.
15 Keng Siau. Information modeling and method engieering: A psychological perspective. *J. of Database Systems*, 10(4):44–50, 1999.
16 The European Committee for Standardization. CEN ENV 40 003, Computer Integrated Manufacturing: Systems Architecture Framework for Modeling , 1990.
17 Terje Totland. *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, August 1997.
18 J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.
19 Zachman Institute for Framework Advancement. *The Zachman Framework*. Various pages at `www.zifa.com`; the "Quickstart" is particularly relevant.

This article was processed by the author using the TEX macro package from Springer-Verlag.