Running Head: INFORMATION SYSTEMS ARCHITECTURE FRAMEWORK

MetaTech Consulting, Inc.

White Paper

An Assessment of an Enterprise Information Systems Architecture Framework

Jim Thomas (mailto:jim.thomas@ieee.org)

Chief Technology Officer

MetaTech Consulting, Inc

http://metatechconsulting.com

.

March 7, 2004

Abstract

It is the purpose of this paper to provide an assessment of a framework for architecting a knowledge-based information system. It first distinguishes *architecture* from *engineering* for the purpose of distinguishing architectural artifacts, it then introduces information systems in general and knowledge-based information systems in particular, and it finally it presents a framework for assessment. The framework presented was assessed to be suitable and appropriate when architecting KBS systems.

Table of Contents

An Assessment of an Enterprise Information Systems Architecture Framework

It is the purpose of this paper to provide an assessment of framework for information systems architecture for architecting a knowledge-based information system. This is completed by first presenting the substantiating information on architecture discipline and its artifacts. It begins with an introduction to the Software Architecture discipline to support the necessary assertion that the artifacts generated through the architecture definition process are substantially different than through produced through Systems Engineering. This is followed by a relationship and interdependence of the roles of the practitioners of these disciplines. The discussion of architecture concludes with formal descriptions of architectural artifacts as supported by the Software Engineering Institute and the Institute of Electrical and Electronics Engineers (IEEE). Having established a sufficient context for the architectural processes, the paper continues by establishing the bounds of the remaining discussions by providing a brief foundation of Enterprise Information Systems. Specifically addressed within that discussion is the classification of different information systems as well as the classification of content within those systems by relative maturity and value. With all prerequisite discussions complete, the paper then introduces a commercial enterprise architecture framework. In the interest of brevity, this paper will not provide an extensive treatment of the framework - those interested in a deeper understanding of the subject are encouraged to read the referenced documentation. This paper will, however, substantiate the strengths and weaknesses of the framework as a construct for architecting modern information systems – specifically *knowledge-based system*. The paper concludes with an assessment of the suitability and appropriateness of the framework when architecting such systems.

Architecture Versus Engineering

The software industry undertakes efforts of striking magnitude and complexity with the best of intentions only to achieve total success on a fraction of the starts. The rate that technology continues to evolve easily surpasses any software development team’s ability to assimilate it. Additionally, the complexity of the systems being built is such that it is simply not possible to manage all details of its creation in one’s mind. Nearly a half-century ago it was realized that building software required the same discipline and rigor as had been applied in more traditional engineering fields for centuries. Simply adopting a metaphor does not resolve all the woes of an industry. The not-quite-perfect fit of the civil engineering metaphor has left the emerging software disciplines with not-quite-trivial issues that continue to plague them. Research on the discipline of Software Engineering and Software Architecture is ongoing and has been well documented by practitioners and researchers such as W. Maier, M. Shaw, F. Brooks, A. Bryant, and D. Garlan. Individually, and in combination, works by these authors have profoundly affected the maturation of Software Engineering and Software Architecture.

Differentiated from Engineering

The foundation metaphor on which the software industry has evolved is based on construction. *Engineering,* a term lifted directly from the civil construction, has provided the discipline classifiers within the software industry. References to applications of this metaphor to the infant software discipline are present as far back as 1958 (Brooks, 1987). The language of the discipline has clearly accommodated the metaphor as typified by terms such as *requirements, specification, independent verification and validation, interfaces,* etc.

Architecture¸ a natural extension to *Engineering,* was first suggested for use as a classifier within the software industry in the early 1960s though it did not achieve general use until it was introduced at a 1969 NATO conference (Bryant, 2000). Though the term entered general use, only through its roots from the like-named civil discipline did it have general meaning. It was not until 2002 that consensus was achieved of the meaning of the term in the domain of software (Maier, Emery, & Hillard, 2001).

To support further discussion on the topic, it is necessary to make explicit the formal definitions of the relevant terms. Appreciation for the following terms will ensure a consistent context for subsequent analysis.

Software Engineering. The IEEE Standards Collection: Software Engineering (as cited by Pressman, 2001, p. 20) defines Software Engineering as "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance or software: that is, the application of engineering to software. (2) The study of approaches as in (1)."

Software Architecture. In describing architecture as "a representation of overall structure, or a representation that captures the essential features of a system while masking the unimportant details", Maier (1996) likens systems architecture to civil architecture in that it communicates style, decoration, and patterns of organization. The IEEE Standards Collection: Software Engineering (as cited by Maier, 2001, p. 108) defines architecture as "the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution."

The preceding paragraph may be interpreted to say that architecture is design. Clements (n.d.) supports this generalization though he adds a caveat that aims to make clear that not all design is architecture. He distinguishes the two as follows:

> The architect draws the boundary between architectural and nonarchitectural design by making those decisions that need to be bound in order for the system to meet its development, behavioral, and quality goals. (Decreeing what the modules are achieves modifiability, for example.) All other decisions can be left to downstream designers and implementers.

Roles

There is a tendency to attempt to partition the roles of architecture and engineering in such a way that there is clear distinction between them. While doing so makes it far easier to argue the need for the coexistence of the disciplines, it is not a desirable model. Rather, the roles of system engineer and system architect coexist on a single continuum that is on one end is quantitative and the other end is qualitative (Maier, 1996). An architect, being the trusted agent for the client, generally follows an inductive path to help articulate the objective capabilities of what is to be built. The architect produces one or more views that of the objective system, *architectural views*, that communicate to the engineer what is to be built and what is necessary to ensure customer satisfaction. The engineer, being concerned with quantifiable costs and then follows a deductive path by applying mathematics and hard science to achieve technical optimization in designing the objective system.

The architect is active in the conceptualization, scoping, partitioning, specification, and certification of a system. Clients and engineers generally do not share a common vocabulary. Clients state requirements in terms of their business needs. Engineers require far more specificity before they can perform detailed design with the rigor mandated by their discipline. The architect must be able to interpret the client's business needs and translate them into high level specifications. She will capture the relevant details and represent them in architectural views that address each of the clients concerns. Multiple iterations will likely be needed to ensure that all of the scenarios that characterize the business needs are captured. The architect will partition the architectural components in what is believed to be the best configuration – this will generally be in accordance with well established architectural styles that have performance characteristics appropriate for the clients needs. Once complete, the architectural views are conveyed to the engineers such that they can translate them into specifications suitable for implementation. The architect continues her involvement to ensure the intent of the client's needs are given deference when any ambiguity in the specifications exist or when latitude exists for engineering or implementation trades. It is the architect, on behalf of the client, that is responsible certifying the design and resulting system satisfies the client's needs. Having done so, the architect will greatly increases the probability of ultimate acceptance by the client.

Furthermore, on the subject of increasing probability of success, Maier and Rechtin (2002) emphasize the necessity to have both absolute consistency and completeness of interface descriptions and a disciplined methodology when undertaking a large, complex effort. Again, the architect is active throughout to lifecycle to provide the necessary consistency.

Architecture Artifacts

The process of architecture exists for a number of purposes. One readily identifiable purpose is to generate some set of documentation that is meaningful to stakeholders. Clements et al. (2003, p. 301) summarize the relationship between the stakeholders and a number of architectural products, referred to as *architectural views*, in the chart represented in Figure 1. Though it is beyond the cope of this paper to characterize each of the architectural artifacts individually, a characterization of the artifacts as a whole is provided presently.

Architectural Views

The IEEE Standards Collection: Software Engineering (as cited by Maier, 2001, p. 108) defines architectural views as “a collection of models that represent one aspect of an entire system.” Such a model is relevant only to the system being defined. It does not provide a generalization suitable across multiple systems.

Architectural Description

Maier (2001, p. 108) describes an architectural description as “a concrete artifact” that communicates the architecture. The IEEE Standard 1471 places normative requirements of these artifacts. These descriptions, contained in one or more *architectural views*, must specify the systems stakeholders and address their architectural concerns including: (a) functionality, (b) performance, (c) security, and (d) feasibility.

Additionally, an architectural description must make explicit the rational for making key architectural decisions.

Information and the Enterprise

The term *Enterprise Information System* has emerged only within the last decade and still has not achieved general use within the system engineering vernacular as evident in its continued absence from most prominent texts on the subject. It has, however, become well enough established to warrant a international conference with sponsorship from such organizations as the *Association for Computing Machinery*, the *American Association of Artificial Intelligence*, and the *Institute of Electronics, Information and Communication Engineers*. These organizations will, with others, host the 6th Annual Conference on Enterprise Information Systems in April of this year. The convergence of energies in the field has furthered the drive toward more powerful and capable information systems with ever increasing value to the host enterprises. The following paragraphs introduce several classes of information systems as they have emerged in the past few decades. These classes of systems have matured at differing rates an in a non-sequential fashion.

Information Systems

Through the last decade of the 20th century, six types of information systems were well understood (Lauden & Lauden, 1998). These system classes were a) *Transaction Processing Systems (TPS)*, b) *Office Automation Systems (OAS)*, c) *Knowledge Work Systems (KWS)*, d) *Decision Support Systems(DSS)*, e) *Management Information Systems (MIS)*, and f) *Executive Support Systems (ESS).* A graphical representation of the application of these systems is provided in Figure 2 and a chart of each system type is represented in Figure 3. *Knowledge-based systems (KBS)* systems, also called *expert systems*, began emerging in the mid 1980's to address a desire to produce intelligent systems that leveraged rules that embody corporate knowledge together with inference engines to automate decision making activities. Challenges, such as those outlined by Swartout (1996), impeded development of these systems and many became disillusioned with the potential viability of the system. Efforts to build effective KBSs continue today with activities to establish development methodologies, produce ontologies and other constructs necessary for the operation of inference engines, and to mature the tools and techniques for capturing the knowledge and expertise of domain experts.

Content Maturity

There exists a natural hierarchy of the maturity of *content* that is known by the moniker *DIKW Hierarchy*. The earliest modern characterization of the concept was provided by Harland (1982) and was embodied in a cartoon (Figure 4) that identifies a) *information*, b) *knowledge*, and c) *wisdom*. Harland attributes the foundation of this hierarchy to the T.S. Eliot titled *The Rock*. It is unclear at what point the model was extended to recognize *data* as an identifiable layer of the hierarchy, but Ackoff (1989) extends the model further to also include *understanding.*

Information systems have matured over the years to satisfy requirements within the topmost levels of the DIKW hierarchy (e.g. knowledge and wisdom). *Transaction Processing Systems* principally concern themselves with the *data* layer while *Office Automation Systems* deal with the *information* layer. *Knowledge Work Systems* are associated with the *knowledge* layer of the model and *Decision Support Systems, Management Information Systems*, and *Executive Support Systems* are all related with the *understanding* layer. It is the objective of those designing the *Knowledge-based Systems* and *Expert Systems* to address the *knowledge* layer of the model.

Application of an Architecture Framework

Zachman (1987) presented a framework for Information Systems Architecture that proposed three categories of concerns and six levels of detail for each. This, he represented in a 3x5 matrix that provided a high level model to guide development. His framework was *descriptive* rather than *prescriptive*. That is, each cell of the matrix describe the products that might be developed rather than prescribing precisely what those products must be. It was his expectation (Sowa & Zachman, 1992) that his framework would be used with, rather than replace, the other programming tools, techniques, and methodologies common to the development of software systems. By 1992, the model had been extended from three to six columns resulting in the 30 cell matrix commonly used today. Figure 5 depicts the framework matrix as it exists today. The following brief paragraphs will describe the columns and rows of the framework as characterized by Sowa and Zachman (1992).

Columns

Each column of the model represents a different type of abstraction, or different ways to describe the real world.

Data. This column, from the original model, shows what entities are involved. The content of this column is identified by questions asking “What”.

Function. This column, from the original model, shows what functions are performed on the associated data. The content of this column is identified by questions asking “How”.

Network. This column, from the original model, shows the location and interconnections between the elements of the system. The content of this column is identified by questions asking “Where”.

People. This column, in the extended model, is used to represent the real-world people or organizational entities concerned with functions. The content of this column is identified by questions asking “Who”.

Time. This column, in the extended model, is concerned with the temporal issues of the functions of the system. The content of this column is identified by questions asking “When”.

Motivation. This column, in the extended model, captures the rational for why functions occur. The content of this column is identified by questions asking “Why”.

Rows

Each row of the model represents a specific perspective, set of constraints, or level of abstraction to be applied to each column.

Scope. This row represents the perspective of the planners view and represents external constrains (e.g. financial). The cells of this row make explicit the context for the information system.

Enterprise model or *Business model.* This row represents the owners view and corresponds to the policy and usage constraints meaningful to the system. The cells of this row make explicit the conceptual model meaningful to the information system.

System model. This row represents the designers view and corresponds to the operational and structural constraints meaningful to the system. The cells of this row make explicit the logical model meaningful to the information system.

Technology model. This row represents the builders view and corresponds to the technological constraints meaningful to the system. The cells of this row make explicit the physical model meaningful to the information system.

Detailed representations or *Out-of-context views.* This row represents the subcontractors view and corresponds to the constraints meaningful to the system though unrelated to the system itself. The cells of this row make explicit the physical model meaningful to the information system.

Application to Knowledge-based Systems

With respect to the Zachman Framework, Knowledge-based Systems (KBS) differ from other information systems mostly in the *Business Model* and *System Model* rows and also in the *Data, Function, and Motivation* columns. Though definition of the six cells that fall in the intersections of these rows and columns are important for all types of information systems, KBS required strong conceptual and logical models of the entities and processes as well as the relationships between each. While models of literal entities and generalizations of those entities will generally suffice for other systems, greater abstraction is necessary for KBS. Also, the most meaningful abstractions in a KBS will include the motivating factors that are frequently unnecessary for other information systems. The logical and conceptual models are constructed, in part, of ontological information – criteria for distinguishing types of things and their relationship. While this degree of ontological information is certainly necessary for KBS, it is not wholly sufficient. Deeper ontological information, referred to as an *Upper Ontology* is necessary to bridge content for separate domains (e.g. weather information and handling characteristics of an automobile). While our minds are capable of relating this disparate information within a few fractions of a second, few manmade information systems are capable of synthesizing those relationships given any amount of time without an explicit ontology. An interesting collection of KBS projects, many of which indicate the importance of the ontology by including the term in the projects name, can be found at http://www.cs.utexas.edu/users/mfkb/related.html.

Summary

This paper has presented a discussion of a candidate architecture framework for information systems. It began by distinguishing *architecture* from *engineering* for the purpose of distinguishing architectural artifacts. An introduction to information systems was then provided to allow the author to distinguish between the different types of information systems. This was necessary as only the KBS was of concern for the assessment. The Zachman Framework for Information Systems Architecture was presented with sufficient detail to support an assessment of the Framework in terms of its suitability and appropriateness when architecting KBS systems.

Conclusions

The Zachman Framework for Information Systems Architecture captures well the concepts of DIKW Hierarchy and, thus, it does accommodate the entities are the level necessary to produce a robust KBS. Rather than extending the model (e.g. adding additional rows or column) to satisfy the unique demands of KBS development, the artifacts described by the Framework must simply be developed to a greater degree of completeness. The Framework is well suited for use when architecting KBS systems though the practitioner must give deference to need for the ontology to be captured sufficiently well.

References

Ackoff, R. L., (1989). From Data to Wisdom. *Journal of Applies Systems Analysis*, Volume 16, 1989 p 3-9.

Brooks, F. P. (1987, April). No silver bullet: Essences and accidents of software engineering. *IEEE Computer*, 10-19.

Bryant, A. (2000). *It's engineering Jim….but not as we know it: Software engineering – solution to the software crisis, or part of the problem?* Proceedings, 22nd international conference on Software engineering, 77-86. ACM Press New York, NY, USA.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., & Stafford, J. (2003). *Documenting software architecture: Views and beyond.* Addison-Wesley, New York.

Clements, S. (n.d.). What's the difference between architecture and design? Essays on Software Architecture. Retrieved July 26, 2003 from http://www.sei.cmu.edu/architecture/essays.htm

Harland, C., (1982). Information as Resource. *The Futuris*t, December 1982 p 34-39.

Lauden, K. C. & Lauden, J. P. (1998). *Management Information Systems: New approaches to organization & technology (5th ed.).* Prentice-Hall, Upper Saddle River, New Jersey 07458.

Maier, M. W. (1996, October). Developments in System Architecting. *2nd IEEE International Conference on Engineering of Complex Computer Systems,* 139 – 142.

Maier, M. W., (2001, April) Software architecture: Introducing IEEE standard 1471. *Computer*, 107-109.

Maier, W.M. & Rechtin, E. (2002). *The art of systems architecting (2nd ed.*). CRC Press LLC., Boca Raton, FL, 33431.

Pressman, R. S. (2001). *Software engineering: A practitioner's approach (5th ed*.). McGraw-Hill, New York, NY, 10020.

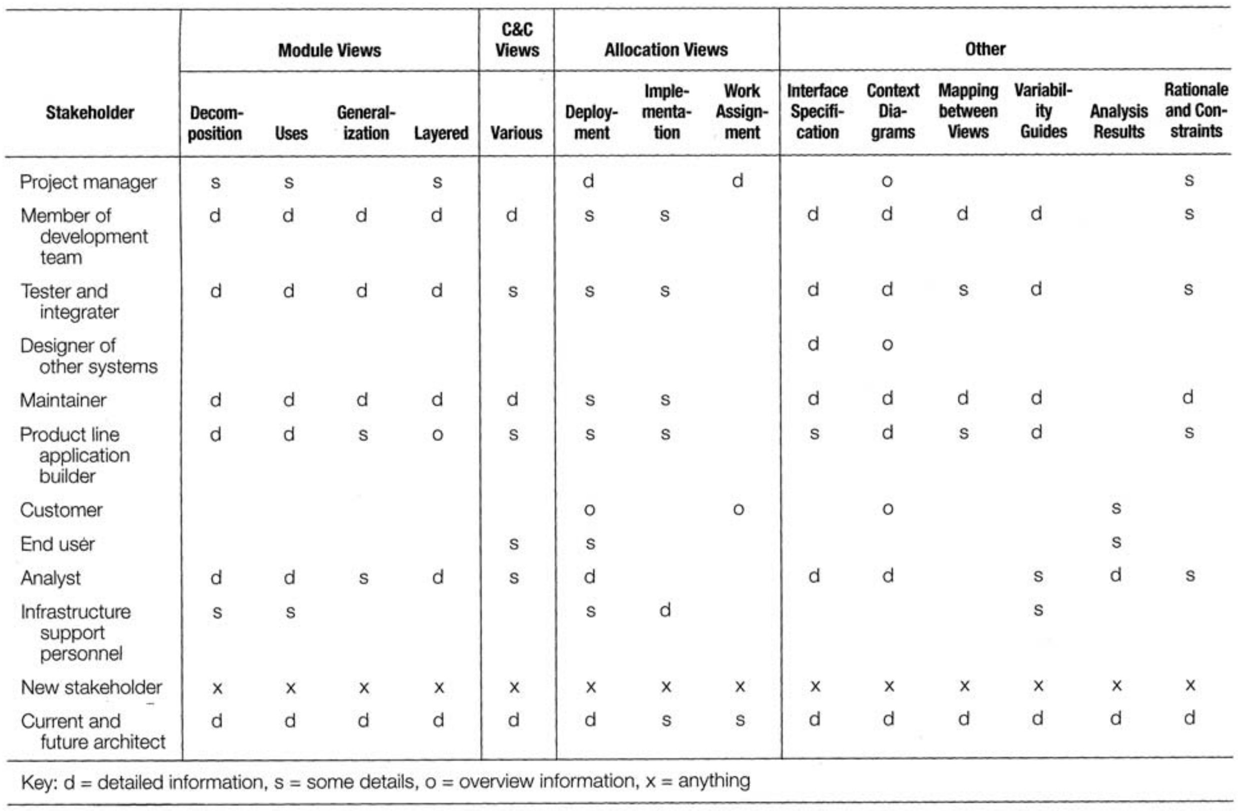
Sowa, J. F., (1992). Extending and formalizing the framework for information systems architecture. *IBM Systems Journal* 26 (3), 590 - 616.

Swartout, B. (1996). Future Directions in Knowledge Based Systems. *ACM Computing Surveys*. Retrieved 7 March, 2004 from http://80-

portal.acm.org.ezproxy.umuc.edu/ft_gateway.cfm?id=63340&type=pdf&coll=portal&dl=ACM&CFID=18780400&CFTOKEN=57173684 .

Zachman, J. A., (1987). A framework for information systems architecture. *IBM Systems Journal* 31 (3), 554 – 470.

Figure 1



Stakeholder	Module Views				C&C Views	Allocation Views			Other					
	Decom-position	Uses	General-ization	Layered	Various	Deploy-ment	Imple-menta-tion	Work Assign-ment	Interface Specifi-cation	Context Dia-grams	Mapping between Views	Variabil-ity Guides	Analysis Results	Rationale and Con-straints
Project manager	s	s		s		d		d		o				s
Member of development team	d	d	d	d	d	s	s		d	d	d	d		s
Tester and integrater	d	d	d	d	s	s	s		d	d	s	d		s
Designer of other systems									d	o				
Maintainer	d	d	d	d	d	s	s		d	d	d	d		d
Product line application builder	d	d	s	o	s	s	s		s	d	s	d		s
Customer						o		o		o			s	
End user					s	s							s	
Analyst	d	d	s	d	s	d			d	d		s	d	s
Infrastructure support personnel	s	s				s	d					s		
New stakeholder	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Current and future architect	d	d	d	d	d	d	s	s	d	d	d	d	d	d

Key: d = detailed information, s = some details, o = overview information, x = anything

Figure 1. Summary of Documentation Needs (Clements et al., 2003, p. 301)

Figure 2

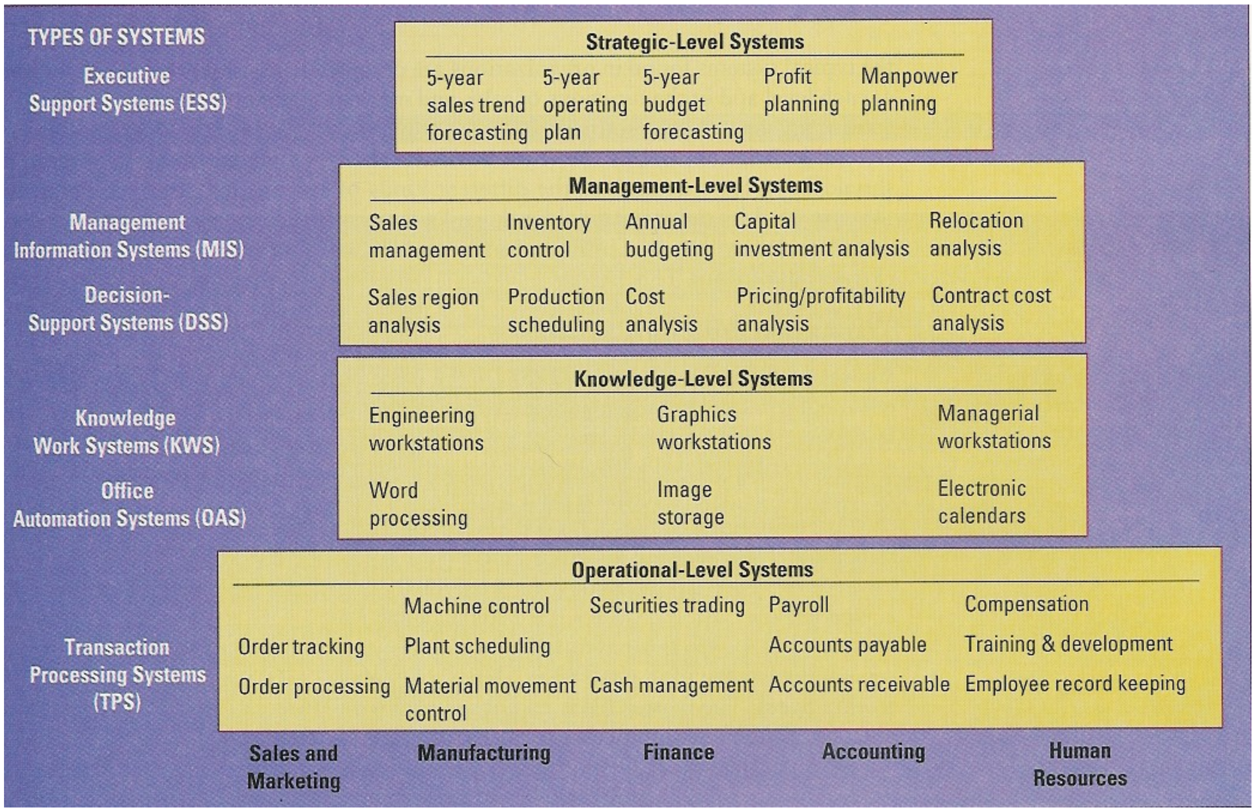
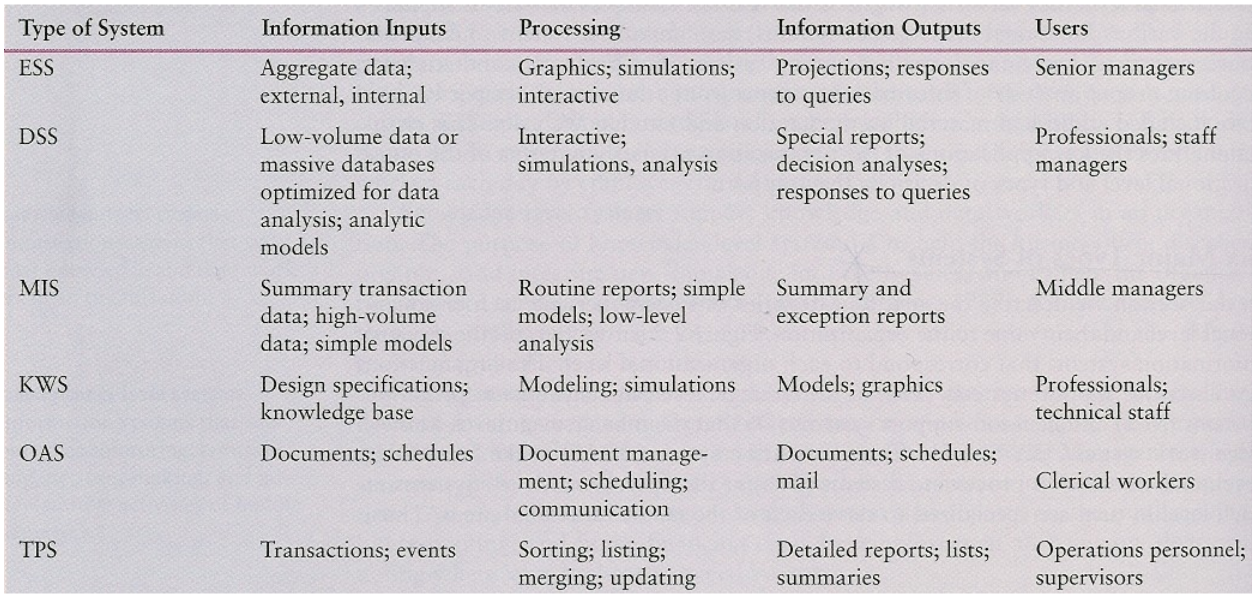


Figure 2. Types of Information Systems (Lauden & Lauden, 1998, p. 39).

Figure 3



Type of System	Information Inputs	Processing	Information Outputs	Users
ESS	Aggregate data; external, internal	Graphics; simulations; interactive	Projections; responses to queries	Senior managers
DSS	Low-volume data or massive databases optimized for data analysis; analytic models	Interactive; simulations, analysis	Special reports; decision analyses; responses to queries	Professionals; staff managers
MIS	Summary transaction data; high-volume data; simple models	Routine reports; simple models; low-level analysis	Summary and exception reports	Middle managers
KWS	Design specifications; knowledge base	Modeling; simulations	Models; graphics	Professionals; technical staff
OAS	Documents; schedules	Document management; scheduling; communication	Documents; schedules; mail	Clerical workers
TPS	Transactions; events	Sorting; listing; merging; updating	Detailed reports; lists; summaries	Operations personnel; supervisors

Figure 3. Information Systems Characteristics (Lauden & Lauden, 1998, p. 40)

Figure 4

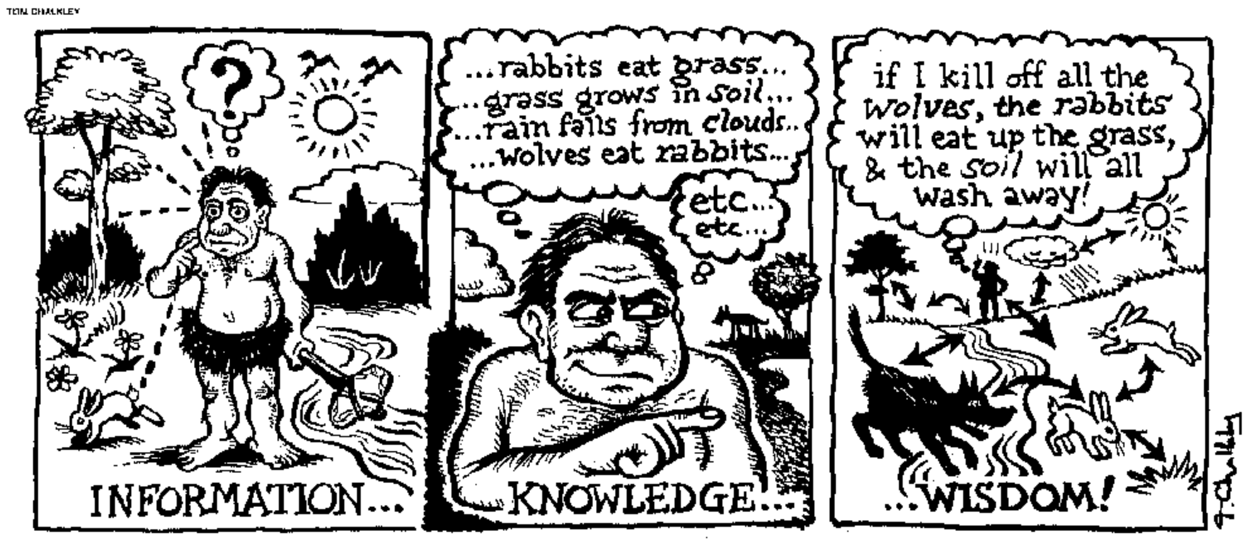
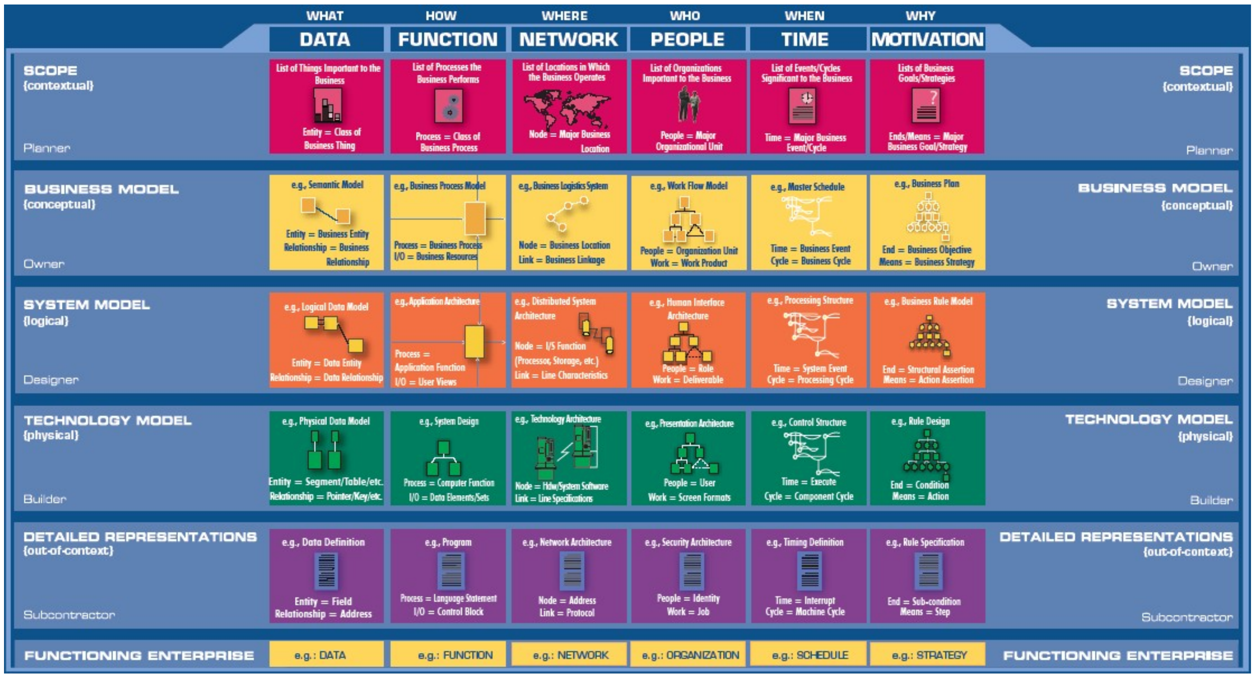

TOM CHALKLEY
?
...rabbits eat grass...
...grass grows in soil...
...rain falls from clouds..
...wolves eat rabbits...
etc... etc...
if I kill off all the wolves, the rabbits will eat up the grass, & the soil will all wash away!
INFORMATION...
KNOWLEDGE...
...WISDOM!

Figure 4. Information, Knowledge, Wisdom (Harland, 1982)

Figure 5



	WHAT DATA	HOW FUNCTION	WHERE NETWORK	WHO PEOPLE	WHEN TIME	WHY MOTIVATION	
SCOPE {contextual} Planner	List of Things Important to the Business Entity = Class of Business Thing	List of Processes the Business Performs Process = Class of Business Process	List of Locations in Which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Major Organizational Unit	List of Events/Cycles Significant to the Business Time = Major Business Event/Cycle	Lists of Business Goals/Strategies Ends/Means = Major Business Goal/Strategy	SCOPE {contextual} Planner
BUSINESS MODEL {conceptual} Owner	e.g., Semantic Model Entity = Business Entity Relationship = Business Relationship	e.g., Business Process Model Process = Business Process I/O = Business Resources	e.g., Business Logistics System Node = Business Location Link = Business Linkage	e.g., Work Flow Model People = Organization Unit Work = Work Product	e.g., Master Schedule Time = Business Event Cycle = Business Cycle	e.g., Business Plan End = Business Objective Means = Business Strategy	BUSINESS MODEL {conceptual} Owner
SYSTEM MODEL {logical} Designer	e.g., Logical Data Model Entity = Data Entity Relationship = Data Relationship	e.g., Application Architecture Process = Application Function I/O = User Views	e.g., Distributed System Architecture Node = I/S Function (Processor, Storage, etc.) Link = Line Characteristics	e.g., Human Interface Architecture People = Role Work = Deliverable	e.g., Processing Structure Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model End = Structural Assertion Means = Action Assertion	SYSTEM MODEL {logical} Designer
TECHNOLOGY MODEL {physical} Builder	e.g., Physical Data Model Entity = Segment/Table/etc. Relationship = Pointer/Key/etc.	e.g., System Design Process = Computer Function I/O = Data Elements/Sets	e.g., Technology Architecture Node = Hdw/System Software Link = Line Specifications	e.g., Presentation Architecture People = User Work = Screen Formats	e.g., Control Structure Time = Execute Cycle = Component Cycle	e.g., Rule Design End = Condition Means = Action	TECHNOLOGY MODEL {physical} Builder
DETAILED REPRESENTATIONS {out-of-context} Subcontractor	e.g., Data Definition Entity = Field Relationship = Address	e.g., Program Process = Language Statement I/O = Control Block	e.g., Network Architecture Node = Address Link = Protocol	e.g., Security Architecture People = Identity Work = Job	e.g., Timing Definition Time = Interrupt Cycle = Machine Cycle	e.g., Rule Specification End = Sub-condition Means = Step	DETAILED REPRESENTATIONS {out-of-context} Subcontractor
FUNCTIONING ENTERPRISE	e.g.: DATA	e.g.: FUNCTION	e.g.: NETWORK	e.g.: ORGANIZATION	e.g.: SCHEDULE	e.g.: STRATEGY	FUNCTIONING ENTERPRISE

Figure 5. Zachman Framework for Enterprise Architecture (http://www.zifa.com, 2004)