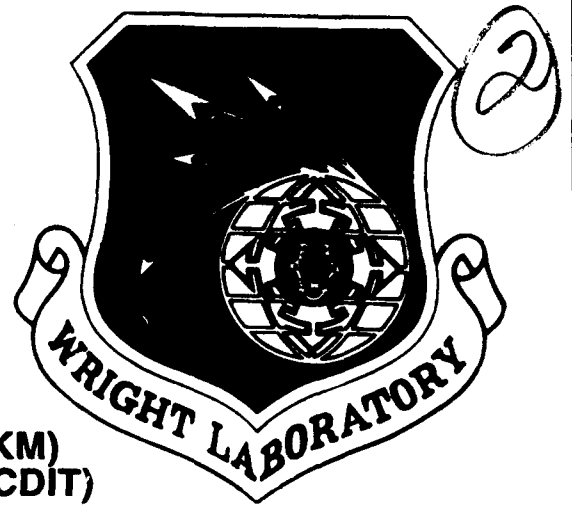


WL-TR-92- 8048

AD-A255 547



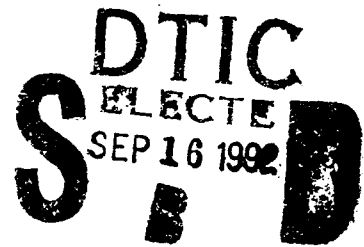
**INTEGRATION TOOLKIT AND METHODS (ITKM)  
CORPORATE DATA INTEGRATION TOOLS (CDIT)**

Review of the State-of-the-Art with Respect to Integration  
Toolkits and Methods (ITKM)

Anthony K. Sarris

Ontek Corporation  
22951 Mill Creek Drive  
Laguna Hills, CA 92653

June 1992



Final Report for Period June 1991 May 1992

Approved for public release; Distribution is unlimited.

92 9 15 042

416327

92-25264



132px

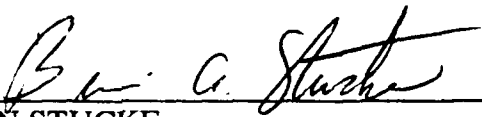
Manufacturing Technology Directorate  
Wright Laboratory  
Air Force Systems Command  
Wright-Patterson Air Force Base, Ohio 45433-6533

## NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



BRIAN STUCKE  
Project Manager



BRUCE A. RASMUSSEN, Chief  
Integration Technology Division  
Manufacturing Technology Directorate

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/MTIA, W-PAFB, OH 45433-6533 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

# REPORT DOCUMENTATION PAGE

FORM APPROVED  
OMB NO. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

<b>1. AGENCY USE ONLY (Leave Blank)</b>		<b>2. REPORT DATE</b> June 1992	<b>3. REPORT TYPE AND DATES COVERED</b> Final Report June 1991 - May 1992	
<b>4. TITLE AND SUBTITLE</b> Integration Toolkit and Methods (ITKM) Corporate Data Integration Tools (CDIT); Review of the State-of-the-Art with Respect to Integration Toolkits and Methods (ITKM)			<b>5. FUNDING NUMBERS</b> PE: 78011F C: F33615-91-C-5722 PR: 3095 TA: 06 WU: 33	
<b>6. AUTHOR(S)</b> Anthony K. Sarris			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b> ONTEK/SR-92/003	
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Ontek Corporation 22951 Mil' Creek Drive Laguna Hills, CA 92653			<b>10. SPONSORING/MONITORING AGENCY REP NUMBER</b> WL-TR-92-8048	
<b>9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Brian Stucke (513-255-7371) Manufacturing Technology Directorate (WLM/TIA) Wright Laboratory Wright-Patterson AFB, OH 45433-6533			<b>11. SUPPLEMENTARY NOTES</b> This is a Small Business Innovative Research Program, Phase I report.	
<b>12a. DISTRIBUTION/AVAILABILITY STATEMENT</b> Approved for Public Release; Distribution is unlimited.			<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT</b> Documented are the results of a study performed of the state-of-the-art with respect to systems analysis and modeling methods and tools which aid in the integration of distributed corporate resources. One major source of data regarding the state-of-the-art was a survey conducted of industry, tool developer/vendors and academia. The survey and other data sources are described in this document. The scope of state-of-the-art assessment activities under this ITKM project was focused on tools and methods to support enterprise analysis and modeling, particularly in the context of enterprise information integration. Given that breadth of context, as well as the tendency towards increasingly blurred boundaries among the various information technologies utilized for enterprise integration, some of the data in this document relates to areas outside the traditional realm of enterprise analysis and modeling methods and tools. For example, broader methods and tools such as: information frameworks and architectures; repository/data dictionary technologies; Computer-Aided Software Engineering (CASE) tools; "re-engineering" methods and tools, including those for integration, access and migration of legacy databases; and systems development life-cycle methodologies were also included in the study, at least to some degree.				
<b>14. SUBJECT TERMS</b> Integration, Database, Representation Language, Methodology, State-of-the-Art, Simulation, Enterprise Analysis and Enterprise Modeling, Framework, Enterprise, Toolkit, Modeling, IDEF.			<b>15. NUMBER OF PAGES</b> 132	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASS OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASS OF ABSTRACT</b> Unclassified	<b>20. LIMITATION ABSTRACT</b> SAR	

Standard Form 298 (Rev 2-89)  
Prescribed by ANSI Std Z39-18  
298-102

# Table Of Contents

	Page
<b>1. Introduction</b> .....	<b>1</b>
1.1. Description Of ITKM State-of-the-Art Review Document .....	1
1.2. Scope and Objectives Of ITKM.....	1
1.3. Background To ITKM .....	2
1.4. Ontek Approach To ITKM.....	2
<b>2. Summary Of Industry Survey Results.....</b>	<b>4</b>
2.1. Respondents .....	4
2.2. Observations On the State-of-the-Art and Critical Requirements.....	5
2.3. Summary Of Responses and Observations By Survey Area .....	7
2.3.1. Nature Of Existing Modeling.....	7
2.3.2. Current Enterprise Modeling Environment.....	8
2.3.3. Frameworks and Architectures.....	9
2.3.4. Enterprise Business-Level Modeling.....	9
2.3.5. Requirements For Enterprise Modeling.....	10
2.3.6. Model Integration .....	10
2.3.7. Model Configuration Management and Formalism.....	11
2.3.8. Reference Models or Model Libraries .....	12
2.3.9. Existing Methodologies, Techniques, Languages, Approaches and Tools.....	12
2.3.10. Re-Engineering and Repositories/Data Dictionaries.....	13
2.3.11. Software Development and Configuration Management Methodologies.....	14
2.3.12. Prioritized Criteria or Requirements For Enterprise Modeling.....	14
2.3.13. Problems With Enterprise Modeling.....	17
<b>3. Description Of Major SOTA Toolkits and Methods.....</b>	<b>19</b>
3.1. Architectures and Frameworks.....	19
Company-Specific Frameworks and Architectures (e.g. NADSARP, AIA).....	19
CAM-i Computer-Integrated Enterprise (CIE) .....	21
ESPRIT AMICE CIM-OSA .....	21
ICAM Factory of the Future (FoF).....	22
James Martin Information Engineering Framework.....	23
Zachman "Framework For an Information Architecture" .....	24
3.2. Modeling Methods and Tools.....	25
Binary and Elementary N-ary Relationship (B[EN]R) Modeling .....	25
Business Entity-Relationship Model (BERM).....	26
Computer-Aided Software Engineering (CASE) and Re-Engineering.....	27
Concept Maps.....	30
Concept Propositional Analysis (CPA) .....	33
Conceptual Graphs (CG).....	33
Data Modeling Versus Information Modeling.....	36
Entity-Attribute-Relationship (E-A-R) Modeling .....	37
EXPRESS and EXPRESS-G.....	38
Extended Entity Relationship (EER) Modeling (Ross, Bachman Object-Role et al).....	41
Hypertext .....	43
ICAM Definition (IDEF) Modeling Methodology.....	43
IDEF0 Functional Modeling.....	45
IDEF1/IX Information Modeling.....	48



	Page
IDEF2 Dynamics Modeling.....	54
IDEF3 Process Flow and Object State Modeling.....	55
IDEF4 Object-Oriented Design.....	57
Interpreted Predicate Logic (IPL).....	58
NIAM (Incl. RIDL-C and NORM).....	58
Object-Oriented Modeling Paradigm (Ontic, Ptech and DaDaMo).....	60
Process Dynamics/Simulation Modeling (Petri Nets, Finite State Machines et al).....	64
Quality Function Deployment (QFD), House of Quality (HoQ) & Other TQM.....	67
Rule-Based Systems/Expert Systems.....	71
Semantic Network or Semantic Net.....	73
Systems Engineering Methodology (SEM) (Functional Flow Block Diagrams et al).....	74
The Vee Method (or Heuristic) For Knowledge Acquisition.....	77
Yourdon Methodology (Structured Analysis, Data Flow Diagrams et al).....	78
3.3. Model Extension and Integration Projects and Approaches.....	81
GUIDE Data Modeling Extensions Project.....	81
Information Resources Dictionary System (IRDS) (IRDS I and II, ATIS).....	83
Neutral Information Representation Scheme (NIRS) (Incl. IDSE).....	86
Portable Common Tool Environment (PCTE).....	87
Semantic Unification Meta-Model (SUMM).....	89
Shared Reusable Knowledgebase (SRKB) (KIF, KRSS et al).....	90
Various Other Model Integration Projects (CDIF, STARS et al).....	94
3.4. Repository/Data Dictionary Technologies.....	97
3.4.1 Categories Of Data Dictionaries.....	97
3.4.2 Capabilities Of Data Dictionaries.....	98
3.4.3 Commercially Available Data Dictionary Technologies.....	100
<b>4. Supporting Data.....</b>	<b>103</b>
4.1. Overview Of Data Gathering Approach.....	103
4.2. Survey.....	103
4.3. Coordination With Other Related Organizations and Projects.....	103
4.4. Reference Materials.....	105
4.4.1. Bibliography.....	105
4.4.2. Other Source Documents.....	111

**Attachment A — Populated Survey Form**

<b>Accession For</b>	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC QUALITY INSPECTED 3

# **1. Introduction**

## **1.1. Description Of ITKM State-of-the-Art Review Document**

This State-of-the-Art Review Document entitled "Review Of the State-of-the-Art With Respect To Integration Toolkits and Methods" is a scientific and technical report submitted as a contract deliverable in accordance with contract data requirements list (CDRL) data item #A007 under Ontek Corporation's Integration Toolkit and Methods (ITKM) contract (#F33615-91-C-5722). Documented herein are the results of a study performed of the state-of-the-art with respect to systems analysis and modeling methods and tools which aid in the integration of distributed corporate resources. One major source of data regarding the state-of-the-art was a survey conducted of industry, tool developers/vendors and academia. The survey and other data sources are described in the main body of this document, and in an attachment thereto. The scope of state-of-the-art assessment activities under this ITKM project was focused on tools and methods to support enterprise analysis and modeling, particularly in the context of enterprise information integration. Given that breadth of context, as well as the tendency towards increasingly blurred boundaries among the various information technologies utilized for enterprise integration, some of the data in this document relates to areas outside the traditional realm of enterprise analysis and modeling methods and tools. For example, broader methods and tools such as: information frameworks and architectures; repository/data dictionary technologies; Computer-Aided Software Engineering (CASE) tools; "re-engineering" methods and tools, including those for integration, access and migration of legacy databases; and systems development life-cycle methodologies were also included in the study, at least to some degree. These other methods and technologies are closely related to the more traditional enterprise analysis and modeling methods and tools and, therefore, influence them in numerous ways. Also, in many cases, capabilities previously only associated with those other methods and tools are now being seen in the context of enterprise analysis and modeling, as the ITKM notion of an integrated suite of methods and tools applicable across the enterprise analysis and systems development life-cycle is evolving and becoming more widely-accepted.

## **1.2. Scope and Objectives Of ITKM**

The Integration Toolkit and Methods (ITKM) program is intended to provide tools and methods to assist in the tasks of enterprise analysis and enterprise systems integration. The specific ITKM project being performed by Ontek Corporation is developing a matched set of techniques and tools to create enterprise models using an ANSI/SPARC three-schema common semantic model or repository built on a rich, subsumptive representation system. This representation system provides an aggregation which allows the individual tools comprising the ITKM toolkit to share data and information derived from each model. The representation also extends the breadth and depth of semantics which can be captured about the enterprise. Lower-level system capabilities such as memory management and network communications are provided by a virtual machine-type operating environment. Various human user interface mechanisms are being utilized, including Macintosh<sup>®</sup> personal computers and other window-based data acquisition and presentation devices. The ITKM methods and tools are intended to meet the objectives of: (1) enabling management of corporate data access and minimization of data access cost; (2) capturing data meaning and maintaining data integrity; and (3) assisting in enterprise-wide integration.

## ITKM State-of-the-Art Review

The initial focus of the ITKM program is to define requirements for the analysis methods and toolkit, including outlining enhancements which could be made to existing, state-of-the-art methods and toolkits. The program is addressing the need to integrate existing conceptual modeling and system design methods and tools. The program is also tasked with designing and developing new methods and tools to meet more advanced needs and requirements.

### **1.3. Background To ITKM**

The dream of the integrated enterprise has been a goal of government and industry for a long time. The application of computer technology to the task under the Air Force Manufacturing Technology (ManTech) Directorate's Integrated Computer Aided Manufacturing (ICAM) program focused industry on the issues of integration and resulted in a vision and initial framework for the Factory of the Future (FoF) — the first substantial concept of an integrated enterprise. Truly integrated, enterprise-wide information architectures, enterprise models, and standards for communications, graphics and systems software are components of that vision. That vision necessitated a major shift in the thinking of industry concerning the uses and manipulation of information. The concepts of commonality, standardization, distribution, accessibility, and automation formed the substrate of this new kind of thinking. The DoD's current Computer-Aided Acquisition and Logistics Support (CALS)/Concurrent Engineering (CE) initiative, particularly CALS/CE Phase II, shares this vision. The truth of this mind shift becomes evident in comparing the ideas of people who have been through the ICAM and/or CALS/CE experience with those who have not.

If the vision resulting from the ICAM and CALS/CE work is an automated, integrated enterprise, then the realization of this vision requires another layer of substrate to help deliver or enable the concepts envisioned. This necessitates another shift in thinking. The elements of this substrate include representational formality, genericity, a method of formal analysis, and augmentation of human analytical skills through automation of certain cognitive activities. The ITKM program is intended to provide exactly the kind of mechanism needed to accomplish this.

With its emphasis on analysis methodology and tools, ITKM fills an often neglected void in technology development; that is, a means to exploit the results of valuable research by making it possible for those in industry to apply it to their own problems. The ITKM technology Ontek is developing — based on the Platform for the Automated Construction of Intelligent Systems (PACIS) — encompasses enterprise analysis and modeling, legacy database integration, associated user interfaces and analytical methods and is, in theory, capable of providing both enterprise *integration* and enterprise *automation*. Neither integration nor automation of the enterprise will happen, though, without a methodology and a suite of analysis tools usable by people in industry.

### **1.4. Ontek Approach To ITKM**

Ontek's ITKM project is built around five fundamental requirements based on five problem statements developed during pre-contract preliminary research efforts. These requirements served as the underlying basis and drivers for a more detailed requirements analysis, the results of which are the subject of a separate but closely related Needs Analysis Document, entitled "Needs Analysis and Requirements Specification For an Integration Toolkit and Methods (ITKM)". The first two fundamental problems deal with the most basic issues of information sharing, namely: (1) *what information needs to be shared or "integrated"?* and (2) *where*

*is the information, when is it needed, who needs it and for what?* The requirements based on these problems involve cross-domain identification of information usage and the ability to represent organizational and functional relationships for efficient delivery of shared information. The third problem concerns data redundancy, necessitating a requirement for a "no logical replication of data" rule. This can be further described as requirements for data singularity, consistency and integrity. The fourth problem involves capturing the "meaning" of stored data. "Meaning" is essentially contextual information about the data which allows for efficient and effective use of that data for certain purposes or applications in specific domains. This kind of information is generally not available in a computer in a usable form. Solving this problem requires a way of representing data, as well as information about that data, at a deeper analytical level. The fifth and final fundamental problem focuses on the limitations of current, state-of-the-art enterprise modeling methods and tools. This problem actually sets a key requirement for the various tools comprising the toolkit, and the associated methods: *that they are themselves integrated*. An ITKM must encompass existing tools and methods, while at the same time allowing new methods and tools necessary to meet the requirements of enterprise analysis and integration.

The Ontek ITKM project approach consists of three threads: a framework thread, a development thread and a commercialization thread. The framework, exemplified by this State-of-the-Art Review Document and the companion Needs Analysis Document, provides two major benefits. The first is a general assessment and description of existing, state-of-the-art analysis and modeling methods and tools, as well as requirements for new or improved methods and tools. The second is to use of this data as a framework or basis for defining, designing and developing new methods and tools. The framework consists of a number of components such as: basic modeling elements, rules of use, presentation forms, storage forms, user categories, etc. The development thread produces the actual tools and methods based on the framework. The components of this thread are the tools which make up the toolkit, and include some tools already developed to varying degrees based on the PACIS representation system.

Ontek's intent all along has been, and will continue to be, making tools developed based on PACIS commercially available to industry. The commercialization thread therefore runs through the whole project in order to take into account user acceptance issues and ensure the appropriateness of the tools. One of Ontek's constant efforts during several years of research and development has been to find ways to transfer our new thinking regarding analysis of complex enterprise situations to users who face those situations on a day-to-day basis. We have found this task to be very difficult, but nevertheless absolutely necessary. ITKM is an opportunity to attack the problem of making the power of rich representation systems available to end-users, information systems analysts and modelers in industry.

Ontek's ITKM work builds on previous enterprise analysis and modeling efforts conducted in conjunction with Northrop Aircraft Division under the Air Force ManTech Directorate's Automated Airframe Assembly Program (AAAP), and under the Air Force ManTech Data Automation Processor (DAPro) program (specifically Ontek Corporation's project entitled "Inter-Organizational Computer-Integrated Manufacturing" or IOCIM). The DAPro work involved identifying strategic analysis and planning issues related to the concept of IOCIM. The AAAP work involved a proof-of-principle of the ability to incorporate existing enterprise models of several different forms into a single unified model in PACIS. The resulting model maintained all of the meaning contained in the legacy models, represented additional relationships among information in the legacy models, and added new information with deeper semantics. Product definition utilizing a Concurrent Engineering approach served as the primary domain for the AAAP modeling work.

## **2. Summary Of Industry Survey Results**

As part of the review and assessment of the State-of-the-Art (SOTA) with respect to existing enterprise analysis and modeling methods and tools, a survey was conducted of: users of modeling methodologies and tools in the aerospace/defense and commercial manufacturing industry; consulting and systems integration companies; developers and/or vendors of methods and tools; academicians in universities or related research facilities; and enterprises that are combinations of any of the other categories. The results of the survey were used as input to and drivers for many of the requirements expressed in the complementary Needs Analysis Document, CDRL #A006 (entitled "Needs Analysis and Requirements Specification For an Integration Toolkit and Methods"). The following is a summary of the results of that survey. Actual survey results are also contained in Attachment A of this document, entitled "Populated Survey Form". It should be noted that in any case where vendor-proprietary commercial products may have been mentioned by survey respondents, those responses are noted as survey results only. These are not to be regarded as direct references to products, and no attempt was made in this section or Attachment A to specifically note copyright or trademark registrations. Survey responses by no means represent endorsements of particular products or services.

### **2.1. Respondents**

Surveys were mailed to a representative mix of enterprises including: users of modeling methodologies and tools in the aerospace/defense and commercial manufacturing industry and other industries heavily dependent on information processing; consulting and systems integration companies who provide enterprise analysis and modeling and/or systems development services to industry; developers and/or vendors of methods and tools utilized for enterprise analysis and modeling and/or systems development; academicians in universities or related research facilities who are conducting research and development in these areas; and enterprises that are combinations of any of the other categories.

Responses were received from twenty-five (25) respondents. Respondents had the option of identifying themselves or submitting their responses anonymously (i.e. "blind"). All but two respondents identified themselves. Based on this, we were able to verify that the mix of the responses was indicative of the mix of the overall mailing. It was our goal in that mailing to solicit input primarily from users of modeling methodologies and tools within industry, but to also incorporate the views of consultants, tool developers/vendors, and the academic community. The respondents are noted below by category:

- 25 Respondents (complete or nearly complete)
  - 14 Industry User
  - 3 Consulting (including views representative of their customers)
  - 2 Academia
  - 2 Developer/Vendor
  - 2 Both Vendor and Industry User
  - 2 Blind

## 2.2. Observations On the State-of-the-Art and Critical Requirements

Basic forms of enterprise modeling currently receive wide-spread use in industry. Functional modeling is used to support "As Is" and "To Be" analysis both on and above the shop floor. However, the higher you go above the shop floor, the less suited existing methods and tools are to fully support functional modeling — let alone *analysis* of the modeled functions. IDEF0 is the most commonly used functional modeling method. Data modeling is also frequently used in industry. Its primary use is to develop logical data models during the initial stages of database design. Additionally, many enterprises are performing other kinds of modeling that focus on the nature of the data itself, rather than just the structure of the data as it appears in a database. This may be referred to as information modeling, versus data modeling [International Organization for Standardization (ISO), 1987b], and frequently draws upon the American National Standards Institute (ANSI/SPARC) and International Organization for Standardization (ISO) three-schema database concepts [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a] for organization and representation of this information. While data modeling and some limited information modeling are supported by methods — Entity-Attribute-Relationship (E-A-R) modeling, including IDEF1X, chief among them — as well as several tools, most information modeling is far more ad hoc. Most enterprises performing information modeling have developed their own methods and are using either home-grown tools or tools created by customizing and piecing together software such as data directories or emerging repository/data dictionary technologies. This is an area where many requirements still go unmet. Process modeling is also an area where existing methods and tools are only scratching the surface. Some enterprises perform control flow, state and limited dynamic modeling, usually in conjunction with process simulation. However, many respondents noted weaknesses in this area, particularly regarding the complexity of the methods and tools, as well as their inability to support "what if" simulation at the enterprise business process or decision-support level. In fact, respondents almost unanimously noted a lack of methods and, even more so, tools to support analysis at the enterprise business level. While Quality Function Deployment (QFD), Supplier-Input-Process-Output-Customer (SIPOC) and other Total Quality Management (TQM) techniques are being practiced, they are not yet an integral, integrated part of enterprise analysis and modeling and there are few specialized tools to support their efficient application. Many respondents also noted a need for methods and tools to allow them to further qualify or quantify basic enterprise modeling data with such criteria as time, user views and probability. This again brought out the general need to penetrate deeper into the semantics of the enterprise as a way to unlock the true value of management information and to provide productivity enhancing tools for decision-makers.

Most enterprise modeling is performed based on traditional structured analysis techniques; however a combination of both top-down and bottom-up approaches is typically used. Several companies are using object-oriented techniques either in addition to, or as a replacement for, traditional structured analysis and design techniques. Fortunately, there does not appear to be a wholesale rush to object-oriented techniques as the next, great "panacea". Where both techniques are being used, they are rarely used in a combined manner. In other words, object-oriented techniques are being used by some groups for some projects, whereas traditional structured analysis and design techniques are being used for others. This may be due to a perception on the part of industry that the two techniques are incompatible. However, the analysis of SOTA methods and tools presented in Section 3. of this document indicates some cases where academia or developers/vendors of methods and tools are combining the two techniques.

## ITKM State-of-the-Art Review

Computer-Aided Software Engineering (CASE) tools are also beginning to be utilized to provide a broader, more cohesive approach to enterprise modeling, systems design and software development. This may be attributable to the fact that performing enterprise modeling within the context of an enterprise information framework or architecture and a systems development life-cycle methodology is one of the most important issues identified by the survey. This ties closely to another major interest expressed by the survey respondents: the integration of enterprise models — in essence a logically unified modeling environment. This applies not only to a framework or architecture and systems development life-cycle for methodological context, but also to a tooling-related requirement for a common, logical enterprise modeling data/knowledgebase and a means of effectively navigating within that data/knowledgebase. CASE is viewed by many as an initial attempt at such an integrated tooling environment.

While existing methods and tools may meet the "letter" of particular modeling requirements (at least in basic, narrow areas of enterprise modeling), they seem to miss the underlying "spirit". Respondents feel that modeling methods and tools should be analytically powerful and should serve as productivity enhancing tools for modelers and analysts. They should be intuitive and easy to use — allowing rich, expressive models to be easily created, accessed and maintained. They should hide unnecessary complexity from their users. Since the objective is to capture the true nature of the enterprise, including its processes, information, rules, etc., the results of this effort — namely, the *enterprise model* — should be better utilized once created. Models should not simply be spiral-bound and put on a shelf to gather dust. At a minimum, there should be closer linkage between models and the information systems developed to automate enterprise processes and provide management information. In the best case, the models should be executable. In other words, the models and the systems should be one and the same thing. The model/system should be easily adaptable to reflect on-going changes in the enterprise environment across the business life-cycle.

As a start, respondents feel that modeling methods and tools should help the enterprise make better use of its existing data and processes. These are reflected in existing, legacy databases and application programs. Current methods and tools provide little in the way of analytical support for understanding the existing environment. They often completely ignore it. Existing methods and tools also don't take advantage of previous models or general industry knowledge that must serve as the starting point or basis for specialized knowledge in any particular application domain. They are inefficient, in that modelers are often required to "re-invent the wheel", gathering and modeling data that may have been gathered and modeled many times before either within the same enterprise, by similar enterprises, or by professional societies and industry consortia. This data should be available through automated reference models or "model libraries". This would allow modelers to rapidly copy and customize basic information (i.e. the "80% we often hear about), and then devote the bulk of the time to the critical information unique to their enterprise (i.e. the "20%").

Based on the survey, it appears that most enterprises are doing an admirable job of enterprise modeling using good basic methods and tools. This is especially notable given the lack of top-management support and limited commitment to funding that many survey respondents stated were major obstacles to their efforts. On the basis of the positive responses and ideas provided by respondents, it is hoped that improved methods and tools resulting from ITKM will not only support the modelers directly, but will provide them the successes they need to gain top-management's attention and be given the resources they deserve — and require — to do the job that really needs to be done to support enterprise integration.

## 2.3. Summary Of Responses and Observations By Survey Area

### 2.3.1. Nature Of Existing Modeling

As expected, almost all respondents are performing some form of functional and data modeling. Most also perform some [limited] degree of information modeling. Approximately half perform process modeling. Over a third use some form of modeling to further qualify, quantify or otherwise associate more data to the data or information in other existing models.

Functional modeling concentrates not just on developing node trees or hierarchical models of functions, but also involves relating other enterprise models to those functions. Data entities are frequently related to functions in the role of inputs and outputs in functional models such as IDEF0. In addition, many enterprises also model the organizations, automated systems and other mechanisms which perform, or are used to perform, the functions. Approximately half of the respondents noted that they also model cost data, schedule data (such as milestones), and in some cases policies and procedures, in relation to or as part of functional models.

As noted above, data modeling is performed by basically all the respondents. Two thirds of those respondents are performing data flow diagramming. About the same number are performing some degree of deeper data modeling, referred to as information modeling. Over half the respondents categorize data entities based on classification schemes such as "kind of", "part of", etc. About the same number partition data entities using the ANSI/SPARC and ISO three-schema database architecture of External Schema, Conceptual Schema and Internal Schema. This typically takes the form of identifying: the data entity itself; its aliases (including how the data is presented to programs or human users in the form of copy libraries, as well as screen and report layouts); and the format and means in which the data is stored in various file management or database management systems. This last category represents the kind of data typically stored in a data directory, and includes the physical storage location of data, its format and its domain and range of values.

Process modeling is considerably less widely performed than either functional modeling or data modeling. Approximately half the respondents indicated that they perform some form of process modeling. This usually takes the form of control flow modeling. This is often done using basic flowcharting techniques, and in some cases also includes descriptions of branching conditions and state or decision tables. Simulation modeling is frequently performed based on this control flow data. Particular scenarios may also be modeled as part of this effort. A limited number of respondents specifically model temporal process flow apart from control flow. Even fewer model such things as causality (i.e. pre-requisite conditions), effect/ramification (i.e. post conditions) and associated event propagation rules. A small number execute their process models, but only four respondents claim to execute these models directly. Some people refer to models that compile down into code as "directly-executable", particularly if they do not require humans to write any additional code or specify other parameters for the execution process. Others refer to "directly-executable" as meaning that the primitive constructs of the process model are interpreted by the modeling tool directly as they are, without the need to compile the model into code of any other kind (with, of course, the exception of machine-code instructions executed under the host operating system). Unfortunately, the survey left this small but significant difference ambiguous. Therefore, we cannot be sure in which manner the process models described in the survey are "directly-executable". In any case, very few enterprises currently have this capability. It was, however, noted several times later in the survey as an important capability requirement for future modeling methods and tools.



Regarding the concept of “modalizing” model data, it was clear that the respondents often have a need to further *qualify* and/or *quantify* the data in an existing model by directly referring to that data and stating something more about it. For example, many respondents noted that they are at least trying to add time, probability and other scheduling factors to simulation models. One respondent stated that he is adding “user views” to basic modeling data. Establishing a broader context is also something that is frequently mentioned. These “modalizations” are noted as applying to both data and process models.

### 2.3.2. Current Enterprise Modeling Environment

There is an almost even split between environments where one or two groups serve as the lead technical organizations for enterprise modeling (with end-users performing a substantial amount of the modeling themselves) versus those in which end-users are directly responsible for and perform their own enterprise modeling. Others noted cases where only one or two organizations perform and are responsible for enterprise modeling.

In most cases (slightly over two thirds), enterprise modeling is performed based on traditional structured analysis techniques. This is largely done using some combination of both top-down and bottom-up analysis. In a few cases, only a top-down approach is used. Slightly over one third stated that object-oriented approaches are the primary approach used for enterprise modeling. Six respondents noted that both object-oriented and traditional structured methods are being used. Two of those stated that they are phasing out traditional structured methods in favor of object-oriented methods, but are currently operating in a mixed, transitional environment. One said her organization is trying to integrate the two methods, and noted that it is a difficult challenge. One noted that there are two modeling organizations working in different areas and with differing objectives — one using the traditional structured methods and the other using an object-oriented approach. Two had no additional comments.

More respondents (two thirds versus one third) stated that they develop different models of a domain concurrently, rather than in some serial sequence. This includes one respondent who stated that the process is really done iteratively (using a spiral approach). In this case, while there may be some rough sequence generally followed during each pass through the models, that sequence is repeated several times before completed models are produced — often with several kinds of modeling activities being performed in parallel. For those cases where a certain sequence is followed, several different sequences were noted (refer to the actual populated survey for a complete listing). This includes variations among enterprises, as well as within an enterprise (presumably based on the nature and purpose of the modeling project).

Regarding modeling form: almost half the respondents stated that a mixture of graphics and text is used for most modeling. One third noted a preference for graphics-based models. Graphics were noted as being most useful for modeling processes, detailed functions and simple entity relationships. Text was noted as being used more for modeling high-level business functions, business rules and other semantics, and for non-traditional modeling data such as critical success factors and three-schema database mappings.

Almost fifty-percent of the respondents use automated modeling tools (including database technologies) for creating, storing and retrieving model data. Others use a mixture of media or digital models produced using word-processing and/or CAD technologies.

### 2.3.3. Frameworks and Architectures

Slightly less than two thirds of the respondents perform modeling within the context or guidelines of an information framework or architecture. Of those who do not, some are actively working to define a framework or architecture. Home-grown frameworks and architectures are the most common, although in several cases it was noted that leading frameworks or architectures were studied and those portions applicable to the particular enterprise were incorporated. This results in customized or "composite" frameworks and architectures that more closely fit the target enterprise. Zachman's framework was the most frequently mentioned "standard" framework, with James Martin a close second.

### 2.3.4. Enterprise Business-Level Modeling

This area, on the whole, was one of the areas most noted as needing support from modeling methods and tools. This was indicated in the preliminary ITKM analysis, as well, and was in large part the motivation for asking several questions regarding modeling in this area. In each case, the most predominant single answer to the questions (representing approximately one third of the respondents) was that this kind of modeling is currently not being done, and there are currently few, if any, standard methods and tools with which to do it. It is also interesting to note that — given the lack of methods and tools directed particularly at this area — many respondents (approximately two thirds) are using whatever methods and tools they do have to their best possible advantage. Many of the lesson-learned from using these existing methods and tools may be useful as a starting basis for defining, designing and developing methods and tools specialized to the needs of this important area.

Because of this lack of methods and tools, one third of the respondents indicated that they have no way of capturing the value or importance of a process in a model. Others use methods they have developed themselves such as: mapping processes to enterprise Critical Success Factors (CSF's), goals or objectives; relating costs to processes; and assigning weighted values or ranking to processes. When it comes to modeling the information used in the enterprise for decision-making, between one third and one half of the respondents state that they simply don't do it. Those that do perform this kind of modeling use custom methods such as: modifying the modeling form of functions and ICOM's in IDEF0 models to highlight decision-making functions and the information input and output from those functions; modified Entity-Attribute-Relationship (E-A-R) models such as IDEF1X; identifying and modeling information needs or "gaps"; and assigning weighted values or ranking to information. Half of the respondents do not model organizations. Those that do, extend the concepts of IDEF0 mechanisms and E-A-R/IDEF1X entities to better accommodate organizational modeling, or utilize the TQM-based Supplier-Input-Process-Output-Customer (SIPOC) modeling form.

As stated, there are few methods and practically no automated tools to support these kinds of modeling. One third of the respondents indicated they had no methodologies and tools for enterprise business-level modeling. Both the QFD and SIPOC methods, as well as IDEF0, were noted as being used by many of the other respondents; however, there are only a few companies offering QFD-based automated tools, and no tools were identified that were directed explicitly at SIPOC modeling (although many of the respondents are utilizing extended or modified IDEF0 diagrams for this purpose).

### 2.3.5. Requirements For Enterprise Modeling

The respondents noted several requirements that are not being met or met well by existing enterprise modeling methodologies and/or tools. These were taken into account during the needs and requirements specification task of ITKM and are reflected in the complementary Needs Analysis Document. The requirements include:

- Improved model integration, particularly between function/process models and data/information models. This also includes technical standards for model interchange between different modeling methodologies and vendor-proprietary tools.
- Tighter coupling between models and their implementation or execution. This coupling may range from automatic change/update propagation for maintaining concurrency between models and the systems which realize or implement them, to developing directly executable models that are, in fact, one and the same as the systems that implement them.
- More "active" models for simulation at the enterprise business process level. These models would provide better representation of relationships such as those between temporal flow and decision points, and would allow "what if" simulation for decision processes.
- Expanded analytical capabilities for modeling tools, so that they become "intelligent co-workers" rather than just electronic file cabinets. The analytical capabilities should aid in such areas as: assessing the value-added or detracted of functions/processes or data/information; identifying and describing interactions or relationships among functions/processes and other enterprise data; and constraint modeling.
- Changing the perspective of modeling to focus more on business performance improvement, rather than immediately putting so much focus on software engineering or the systems perspective.
- Providing standard reference models at the "coarse-grain" enterprise business level.

### 2.3.6. Model Integration

Almost two thirds of the respondents stated that they currently integrate one or more modeling forms. The most common integration was between function/process models and data/information models, especially between IDEF0 and IDEF1X. This was followed by integration of data/information models with data flow, system dynamics and/or state transition models for simulation purposes (interestingly, the implication here is that state modeling, dynamics modeling and simulation are based on the perspective of data or information rather than primarily on processes). Other model integration efforts include: cost models to function/process models and/or data/information models; data models to information models or models of deeper enterprise semantics; and an attempt to integrate all models throughout the enterprise.

When asked how model integration is currently accomplished, the responses varied greatly. They can, however, be categorized into three broad groups:

- manual, ad hoc integration consistently described as being a very difficult, tedious task;

- limited automated integration between selected model types, such as functional models and data models. This is aided by the fact that several IDEF tool developers/vendors are attempting to integrate their IDEF0 and IDEF1X tools through such means as linking the ICOM's from an IDEF0 model to the data entities and attributes in an IDEF1X glossary or dictionary;
- more general integration through the use of repository/data dictionary technology. This seems to be the direction in which many respondents are headed, although current implementations are still in their infancy. These implementations often involve customization or combinations of several software technologies and still involve substantial manual efforts on the part of a human modeler.

Of those respondents who are developing various kinds of enterprise models, but who are not currently performing model integration, all of them stated that they have a need for model integration. Many noted that they currently duplicate data among related models, and that duplicated data is often not consistent from model to model. Ensuring consistency requires manual model interpretation and consistency checking. Function/process to data/information was cited as the area in which integration was most needed. Integration among tools of the same kind (e.g. IDEF0) supplied by multiple developers/vendors (e.g. Meta Software Corporation, Wizdom Systems, DACOM, etc.) was also cited as an area in which integration is needed. This area is sometimes also referred to as "model interchange". It is apparently important to respondents since more than one modeling tool may be used for the same kind of modeling either within the same enterprise or in related enterprises where model sharing is important (for example for contractors, subcontractors, customers and/or industry consortia). The IDEF Users' Group already has a task group working on model interchange standards for IDEF. Two respondents noted that they believe object-oriented techniques might help mitigate the need for model integration.

### 2.3.7. Model Configuration Management and Formalism

Responses were evenly divided as to whether configuration management during model development was handled as an inherent part of the modeling methodology or external to the methodology. Several of the respondents using CASE tools for model configuration management noted problems if they used multiple CASE tools for modeling. Each of the CASE tools has its own configuration management methods and capabilities. Either one specific CASE tool must be chosen as the baseline and used for all configuration management of all models, or humans must perform integration and configuration management among the various CASE tools. One respondent explicitly mentioned the usefulness of the IDEF methodology's author/library cycle and "kits" for configuration management.

Two thirds of the respondents noted that models, once created, get used for purposes other than the purpose for which the models were originally intended. This indicates a more general usefulness for models that is often not recognized, particularly at the start of a modeling project. It also indicates the need to ensure that models can be easily accessed and maintained over time. There was an almost even split between those who regularly or sometimes maintained models to reflect changes in the real-world domains (including implemented systems) represented by the models, and those who did not. Of those who did, manual procedures were most often used. These

## ITKM State-of-the-Art Review

included periodic reviews, re-evaluations, comparisons, audits or other validation procedures. Others followed software Corrective Action Procedures which specify that whenever software is formally changed, associated models or specifications are also changed. Some respondents are using automated repository/data dictionary technology to assist with this kind of configuration management.

Respondents from industry were universal in noting their application of modeling methodologies as being somewhat or very flexible. Many view formal modeling rules and procedures as "getting in the way" of what they want to model, or being too difficult or time-consuming to apply during model development. Vendors were the most adamant about rigidity, followed by consultants and academicians. Automation of models, particularly in tools supported by model databases, requires considerable formalism and standardization of modeling methodologies. Many vendors and consultants also believe consistent application of formal rules and procedures is simply good modeling practice and ensures rigorous, clear models. Perhaps the point that can be made from these responses is that, to be widely and consistently applied, modeling rules and procedures must not be viewed by industry as arbitrary or established to suit the needs or limitations of automated tools. The rules and procedures must make sound, practical sense. They must also be rich enough to represent the majority of modeling requirements, while also being extensible or flexible enough that exceptional requirements can be captured as well — but without usurping the validity and rigor of the basic rules and procedures applied to the majority of data.

### 2.3.8. Reference Models or Model Libraries

A surprisingly large number of respondents (four out of five) stated that they frequently or occasionally use existing models as the basis for new models. Many of the respondents stating "occasionally" or "rarely" indicated that this was due to the lack of availability, not because of lack of interest. All the respondents expressed an interest in a library of subject area reference models which could be easily copied and customized during new model development. Two-thirds of the respondents stated that they would "definitely" use such model libraries.

### 2.3.9. Existing Methodologies, Techniques, Languages, Approaches and Tools

The most widely used modeling methodologies are IDEF (both IDEF0 and IDEF1X) and Entity-Attribute-Relationship (E-A-R) modeling (other than IDEF1X). These are used by approximately two thirds of the respondents. Computer-Aided Software Engineering (CASE) is also widely used. Additionally, flowcharting, Yourdon (primarily for its Data Flow Diagrams), Critical Path Method (CPM) and Petri Nets/Colored Petri Nets (CPN), Binary and Elementary N-ary Relationship Modeling (primarily NIAM), EXPRESS/-G and Hierarchical Input-Process-Output (HIPO) are fairly commonly used (all are used by approximately one third of the respondents). While many respondents have seen an IDEF3 specification, and slightly fewer an IDEF4 specification, almost no one is currently using the two methods and few noted that they would consider using them. This may be due to the fact that, of the one third of the respondents using object-oriented techniques, there are other methods and tools already being used. This is somewhat less the case for IDEF3 than IDEF4 — which explains the slightly higher interest in IDEF3. These results should by no means discourage work in these areas, as other aspects of the survey indicate that there is considerable interest in process modeling in general and in fully-declarative, objectified models that are directly implementable and executable.

Only three respondents noted that the use of one or more modeling methodologies had been discontinued at their site in the last two years. This seems to indicate that a shake-out has already occurred in the basic modeling methodologies, and that those methodologies currently in use are at least sufficient for modeling one or more aspects of the enterprise. This is supported by requirements for model integration, reference models and new modeling methods which expand existing modeling capabilities and penetrate deeper into key areas of the enterprise such as process causal relationships and the meaning of information (enterprise semantics).

Roughly two out of three respondents are using automated tools of some kind to support enterprise modeling. Among the most popular are various automated CASE tools, as well as IDEF0 and IDEF1X tools. Tools based on object-orientation and other aspects of artificial intelligence (other than CASE tools) are also fairly widely used. More respondents indicated discontinuing the use of a particular modeling tool within the last two years than a modeling methodology (as noted above). This seems to indicate that users are still trying out various tools to meet their needs, and that some shaking out is still taking place among tools. General reasons for discontinuing the use of tools included: poor integration capability; complexity or difficulty of use; size limitations on models; lack of compliance to industry standards; or simply insufficient modeling capabilities in one or more area(s).

Each of the methodologies noted above is described in more detail in Section 3., Description of Major SOTA Toolkits and Methods, of this document.

#### 2.3.10. Re-Engineering and Repositories/Data Dictionaries

Nearly two thirds of the respondents stated that analysis or "re-engineering" of existing, "legacy" databases and/or application programs is currently part of their enterprise modeling process. This is overwhelmingly a manual process in today's environment. Many use the ICAM "As Is"/"To Be" Systems Development Methodology (SDM) as the basis for their manual efforts. Of the few respondents using automation, tools based on the ANSI/SPARC three-schema architecture were noted as the underlying basis. These tools still require substantial human intervention. Of the one third of respondents who are not currently performing re-engineering, half said that it is an important part of future architectures and/or systems development life-cycle methodologies in their enterprises.

Half of the respondents stated that they had logical models and/or formal schemas for their legacy databases, but several of them noted that this was a only a recent occurrence — the results to-date of applying their re-engineering techniques and technologies. Others stipulated that they had mostly schemas with few logical models, and that both existing logical models and schemas are inadequate to reflect the semantics of their existing database contents. The most common database management systems (in sheer numbers of respondent sites having one or more database implemented in them, not in terms of the quantity of data stored in them) are DB2 and Oracle, followed closely by IMS, then Ingres, Rdb, DMS, RAMIS and others.

Almost two thirds stated that they currently have a data dictionary or other repository technology. In most cases, these are home-grown or customized systems that are more like passive "directories", rather than active "dictionaries". A few respondents are using CASE tool-based dictionaries.

CASE, re-engineering and data dictionary technologies are described in more detail in Section 3., Description of Major SOTA Toolkits and Methods, of this document.

### 2.3.11. Software Development and Configuration Management Methodologies

Almost two thirds of respondents follow a formal systems development life-cycle (SDLC) methodology during software design, development and support. In the majority of cases, this home-grown methodology is based on traditional SDLC methodologies such as the ICAM Systems Development Methodology (SDM), MilStd 2167, etc. The "spiral model" is becoming more widely used than the "waterfall model". Using the "spiral model", an iterative, semi-parallelized approach to systems analysis, specification and design is followed, wherein results produced in early stages or phases of the process continue to be refined as the process progresses toward actual development. In the alternative "waterfall model", the results of each stage or phase of the process are frozen and are rarely modified once the effort has progressed to a subsequent phase. This limits flexibility and may lock-in errors generated early in the process. When it comes to SDLC methodologies, most respondents appear to like to pick and choose from among the best aspects of well-established, industry-common SDLC methodologies and then create a home-grown synthesized or composite version that is particularly suited to their environment.

When asked how configuration management is actually performed across the software life-cycle, the responses varied considerably. Many enterprises said they either don't do it or do it poorly. At the other extreme, many enterprises have home-grown or customized automated systems to support the configuration management process. In the middle are several enterprises that use formal procedures, check-in/check-out library systems, version control systems, or combinations of these.

Regarding configuration management specifically among logical data models, physical database designs and actual database schemas, many respondents (almost half) stated that they don't do it. Others referred back to their repository/data dictionary technologies or to other automated systems. Still others again noted formal, manual procedures such as software Corrective Action Procedures (wherein formal changes to the actual database schema are also reflected in logical data models and/or physical database models). When a similar question focused on configuration management or correspondence among process models, programming specifications and actual program code, even more respondents (over half) stated that they don't do it. Others again mentioned repository/data dictionary technologies and/or CASE tools, or formal manual procedures. One respondent described the approach used in his enterprise as "wishful thinking, plus debugging".

### 2.3.12. Prioritized Criteria or Requirements For Enterprise Modeling

Respondents were asked to rank from highest to lowest importance ten criteria relating to requirements for enterprise modeling methods and tools. All ten should be considered to be quite important, as they were identified by a number of sources as a result of previous studies in this area (including preliminary ITKM analysis efforts). The purpose of this aspect of the survey was to determine the *relative* importance of these ten criteria and to establish priorities among them.

One criterion was clearly identified as being of the highest importance. That criterion states that methods and tools should *support enterprise modeling in the context of an information framework or architecture, and/or as part of a systems development life-cycle methodology*. Many respondents also noted through their answers and comments to other, related survey questions that frameworks, architectures and systems development life-cycle methodologies are an increasingly important part of their enterprise modeling efforts. These frameworks, architectures and systems development methodologies, in fact, act as underlying mechanisms to provide scope, context,

control and coordination for enterprise analysis and modeling activities, as well as systems development activities throughout the enterprise information life-cycle. They have a major impact on model integration and configuration management capabilities. More importantly perhaps, treating various forms of enterprise modeling as components of some larger, coordinated framework, architecture and systems development life-cycle methodology helps ensure that the pieces of the puzzle, once created, fit together. When they fit together, they form a complete, rigorous model or representation of the enterprise suitable for use for various enterprise integration and improvement efforts. In the current environment of increasing complexity and diminishing dollars, frameworks, architectures and system development methodologies ensure the most integrated and efficient use of resources.

The next highest criterion states that models should *capture lots of information or semantics*. In other words, models should be broad, deep and rich in expressive power and capability, and we should be able to fully model any enterprise information we choose. This indicates that industry has recognized that current models only scratch the surface of the information necessary to describe the enterprise at a level sufficient to support enterprise integration and decision-support capabilities. This criterion is followed closely by one stating that models should be *easy to create and maintain*. If they are aren't, they probably won't be used, or used to their fullest extent. That doesn't sound too unreasonable or difficult on face value and when considered by itself. But consider this criterion in conjunction with the previous criterion — the difference in ranking points between the two being only a small fraction. On the one hand, model makers and users want models to be deep and rich, capturing as much information as possible. But at the same time, they want the models to be easy to use and maintain. This poses a difficult challenge for the developers and vendors of methods and tools: how do you capture the complexity of an enterprise in a model — while doing the *simplest* modeling possible? The answer may depend on having extremely rich and powerful methods and tools at the core of ITKM technologies, but hiding those from end-user modelers through successive layers of abstraction. Such an approach could enable end-user modelers to indirectly make use of sophisticated methods and tools through easier-to-use and more flexible human user interfaces at the outermost layer of abstraction. These underlying methods and tools might also be used to create and store libraries of model data which could be easily and rapidly extracted from and customized to create new models. No approach may offer a single, complete solution to the problem. However, even partial or incremental solutions may make using advanced modeling tools less tedious and confusing for end-users.

The fourth-ranked criterion states that *models of one element or aspect of the enterprise should integrate with models of other elements or aspects*. Modelers want to be able to create data in one model and, to the degree it relates to some other model, carry that same model data forward into the new model. This entails ensuring consistent meaning and use of model data — or at least knowing the differences — perhaps through a common, logical model data/knowledgebase. This also means having navigational capability provided between or among various modeling methods and tools used for capturing and presenting different enterprise model data.

The fifth or middle-ranked criterion indicates that modelers want *automated tools to assist with modeling*. Given the previous criteria for ease of use and maintenance, expressive power, and model integration, it is obvious that at least some degree of automation must be brought to bear to take the burden off of human modelers applying manual methods and using manual tools. The sixth criteria states that models should translate easily into program code and/or database schemas. In other words, modelers don't want to create a rich, complex model of the enterprise only to manually interpret and translate that model into database and program specification and, ultimately, into databases and code. If there is not a close coupling between the enterprise



models and the information systems created based on them, then chances are the information systems will be maintained, but the models will not. That means that once the model has been created it will quickly become outdated. Once a change is made to the information system, the model will no longer be in synchronization with the information system. It only makes good sense to somehow automatically use the enterprise model to generate the databases and code — or better yet, to directly execute the model itself as an information system. To do so, the model must be in machine-interpretable form. This provides all the more support for the previous criterion relating to the need for automated tools to assist with modeling.

There is already a considerable amount of enterprise data, including some important semantics, described in two particular forms of enterprise models referred to as “databases” and “application programs”. The criterion ranked seventh states that the methods and tools should *assist with re-engineering of legacy databases and application programs*. Re-engineering is just now emerging as an important part of an overall enterprise integration strategy. While re-engineering has been the subject of considerable discussion over the last two to three years, most of the practical focus has been limited to restructuring old application programs written in unstructured or poorly structured COBOL. Few enterprises considered re-engineering an integral part of their enterprise analysis and modeling activities. Progress in other areas of enterprise analysis has led to a rise in interest in re-engineering. For example, the lack of logical database models did not take on much importance in many enterprises until efforts were undertaken to implement and populate data dictionaries. Also, other survey questions indicated that most re-engineering activities are currently supported by largely informal methods and there are few automated tools directed particularly to this need. Therefore, there is a requirement for highly-specialized enterprise modeling methods and tools to support re-engineering, but these methods and tools must (per other requirements) be part of an overall enterprise integration framework, architecture and systems development life-cycle methodology. They must also be easy to use and maintain, represent lots of information about legacy databases and application programs, and be integrated with other enterprise modeling tools.

The next criterion states that both methods and tools should be *based on standards and/or commonly used in industry*. Since model integration (including interchangeability) is also a requirement, and a high desire was expressed elsewhere in the survey for common reference models or model libraries, standards are certainly also important. It should be noted, however, that the respondents do not appear to be interested in standards purely in principle. They want the benefits that standards provide (for example, easier model integration and the need for less specialized training), but they appear willing to by-pass standards if non-standard methods and tools are better able to meet the requirements they view as most important.

Finally, other human user interface issues (besides overall notions of ease of use and maintenance) are also important. The last two criteria, which received exactly the same number of points and were therefore ranked equally, deal with these issues. One criterion states that the methods and tools should support *modeling of one aspect or a few aspects at a time, allowing for segregation of model elements*. For example, a separate modeling form or user interface might be used to capture a hierarchy of processes versus process flow. The other criterion states that methods and tools should support *multiple forms of modeling for the same aspect or element*. For example, information entities might be modeled graphically with E-A-R models or in text for using a Problem Statement Language or constraint language. This criterion was ranked tenth (last) by the largest number of respondents.

Since these two criteria were ranked the lowest, this would seem to indicate that most modelers are not concerned that models try to capture too much data at once (in one modeling

form), and they are apparently satisfied to have one good modeling method and form for capturing a particular type of enterprise data (for example, E-A-R models for data modeling). However, some additional, somewhat different observations may be made regarding these last two criteria and their rankings. First, the finding that modelers are not particularly concerned about methods and forms which clearly segregate and limit the number of different types of data captured in a single method and form might arise from existing practice. Other survey results indicate that existing methods and forms are if anything too segregated, as there was a major need expressed for model integration. Also, more complex forms of enterprise modeling are only being performed in selected cases within selected enterprises. As these more advanced, deeper kinds of modeling methods and forms become more widely used, modelers may find them to be too bundled and may force the developers/vendors of these new methods and tools to segregate the information captured or presented into more workable "chunks", similar to the existing segregation between functional models, process models, data models and information models. Also, modelers might be satisfied with one method and form for capturing a particular type of data, but probably only if that method and form is the one they use in their enterprise. Since other survey results indicate that the choice of methods and forms for particular types of modeling varies greatly from one enterprise to another, tool developers/vendors will still be required to offer modelers a choice of several methods and forms for the same type of information. However, this is apparently a developer/vendor issue, not a modeler (as user) issue.

### 2.3.13. Problems With Enterprise Modeling

Respondents were asked to identify the single biggest problem with enterprise modeling in their current environment. Some couldn't resist and listed two or three problems. The most frequent response was that enterprise modeling lacks the support of top-management. Management wants to know how enterprise modeling relates [explicitly] to productivity improvement and cost reduction. Management may view enterprise modeling as merely an academic exercise with little practical result -- or in some cases as simply a waste of time. The notion may be that spending six months developing a model of a system before developing it just puts the project six [more] months behind schedule. While modeling might result in a "better" system in the end, there are few if any metrics to determine what "better" really means. How do you prove that the system will better reflect user requirements or that fewer software bugs will be arise over the life of the system? In any case, enterprise modeling needs to be more oriented to the needs and language of the business world.

Several other respondents cited a lack of consistency, uniformity or standardization in the application of modeling across their enterprise. It is, therefore, hard to interpret, verify and integrate such models. Others said the biggest problem is that they aren't doing enough enterprise modeling. This probably relates back to the lack of management commitment. If it is not viewed as being important, particularly given the current economic climate and competitive marketplace, it will not get the resources assigned to it that it deserves -- and requires, to produce a meaningful, useful product. Another respondent noted that current models don't support simulation and "what if" analysis for decision-support. This kind of capability might provide something to catch management's attention and prove to them the value of enterprise modeling. The problem is that such capabilities depend upon powerful modeling methods and tools and extensive models of enterprise processes and information. Developing these methods, tools and models will require a significant up-front investment.

## ITKM State-of-the-Art Review

Others noted that a more direct link is needed between models and their implementation. This would mitigate or eliminate the problem of maintaining correspondence between models and systems developed based on them. Models that generate code and/or databases, or which are directly executable, would also greatly improve the efficiency of the systems development process. Several respondents noted that there is a lack of expert modelers and that many modelers are poorly trained. Better methods and automated tools may help, but there is no substitute for qualified, well-educated and well-trained modelers. Still others mentioned the high cost of modeling and the long lead times to develop models for any domain of significance. Finally, one respondent noted a fundamental, but often overlooked problem — it is difficult to decide where to start. It is hard to scope a project that is big enough to make an impact, but small enough to get funded, remain manageable, and be completed within a reasonable time period. Many people use this as a reason not to do anything at all. The respondents to the ITKM survey, however, are at least trying to do something. ITKM methods and tools are intended to both help them do a better job and encourage the others to join in the effort.

### 3. Description Of Major SOTA Toolkits and Methods

#### 3.1. Architectures and Frameworks

*Editor's Note: Some of the material in Section 3.1. also appears in a companion Needs Analysis Document, entitled Needs Analysis and Requirements Specification For an Integration Toolkit and Methods (ITKM), which establishes requirements for an ITKM, including requirements for an ITKM to support information frameworks and architectures. The material included here, however, is somewhat more extensive than the previously published material.*

The notion of information frameworks and/or architectures as specific templates or guidelines describing the desired objectives, functions, information and systems relating to an enterprise is a relatively new one. Prior to the early 1980's enterprise information and the infrastructure that supported it were not so much deliberately planned as they were merely *managed*. The "As Is" information environments in enterprises evolved over time starting with the introduction of the first information technologies in the 1950's, continuing through the "data processing revolution" of the 1960's and 1970's, and culminating with the dawn of the personal computer and the "personal computing revolution" during the 1980's. Beginning in the late 1970's and early 1980's visionaries in both the commercial and defense sectors began to sense an impending "information revolution". The initial buzzword for this phenomenon was Computer-Integrated Manufacturing (CIM). After several years of effort, the name "CIM" became more closely associated with one particular aspect of the information revolution, namely the implementation and integration of computer-based automation technologies on or near the manufacturing shop floor. The remainder of the information revolution — automating and integrating activities above-the-shop-floor and linking those technologies with the CIM technologies — has become known as Enterprise Integration. The acronym CIM is now also being used again by the DoD, this time to refer to a new initiative called Corporate Information Management, which is closely related to Enterprise Integration. Some companies are currently developing overall enterprise frameworks or architectures. Some have initial versions of these architectures, and are in the process of refining and expanding on particular subset frameworks or architectures such as those for information systems. Information systems frameworks or architectures are viewed as critical to the success of an overall enterprise framework or architecture, since in most cases information systems are "the glue that binds together" the other aspects of the overall enterprise into a single, logical working unit. Some specific frameworks and architectures, both those developed for general industry and those developed within specific companies, are described under separate subheadings within this section.

#### Company-Specific Frameworks and Architectures

Many individual companies have or are developing strategic plans for information systems. Those plans are statements of high-level goals or objectives, coupled with observations about which information technologies may be available for implementation at what time period in the future. They are, however, not formal frameworks or information architectures. Many companies have a formal systems development methodology, and some therefore claim they have a "de facto information architecture". This is tantamount to claiming that because you know how to machine aluminum and assemble component parts, that is almost the same as having the product

## U.K.M. State-of-the-Art Review

designs and process plans for the Cadillac Seville®. Admittedly, having a systems development methodology is important, perhaps even necessary, for utilizing enterprise information integration technologies and realizing an information architecture — but it is clearly not the same as already having the architecture, methods and tools themselves. For many companies, in fact, having a systems development methodology has crystallized the need for an information architecture and for integrated methods and tools to help realize it. It has made clear the need to know for *what purpose, in what context, and in what manner* the systems development methodology is to be applied.

Northrop Aircraft Division's Strategic Architecture Project (NADSARP) is one example of a state-of-the-art enterprise architecture project. The need to tie NAD's systems development process to the critical success factors and information needs of the enterprise led NAD to undertake this project to develop an enterprise-wide information architecture and to acquire or develop new methods and tools for enterprise analysis and systems development. Most of the work to-date has concentrated on integrating and improving the overall information systems analysis and planning process using a patchwork of existing, unintegrated methods and tools to the best degree possible. Longer-term efforts include a gradual transition to integrated tools for enterprise modeling and dynamic data dictionary capabilities. As part of the NADSARP effort, several "information objects" were identified which represent clusters of critical, strategic data that professionals use in making key enterprise decisions. Also, current and proposed information system efforts are being evaluated and replanned based on "To Be" criteria.

Westinghouse Electric Corporation (WEC) calls its strategic enterprise architecture project the Computer-Integrated Enterprise (CIE) (the same general name used for the project being performed by the Computer-Integrated Manufacturing - International (CAM-i) consortium, of which WEC is a member company). WEC's CIE has as its principle goal the delivery to users of "the right information, at the right place, at the right time and in the right form". The Design Engineering and Manufacturing Operations Information System (DEMOIS), currently being implemented in WEC's Advanced Development Operations (ADO), is representative of WEC's CIE concept. The objective of the DEMOIS project is to solve a real-world manufacturing problem: reduce the cycle time required for end-users in ADO to obtain labor and material status information from IBM mainframe databases, and to improve the way the information is presented to those end-users. Specifically, the DEMOIS approach is to provide ADO end-users with dynamic, integrated access to information contained in several heterogeneous mainframe databases from Apple Macintosh® personal computers using common end-users applications such as Microsoft® EXCEL and HyperCard®. This is being accomplished via Ontek Corporation's Information Access Vehicle (IAV)™, a tool for legacy database access and integration.

The Aluminum Company of America (Alcoa) has recently completed the initial stage of a strategic information architecture project referred to as the Alcoa Information Architecture (AIA). Alcoa expresses the objective of the AIA in the following way: *Each person in Alcoa should have the information that they require, when they require it, to enable them to excel in performing their work. Access to data is a critical process for Alcoa, as it is fundamental to the quality and effectiveness of our work.* The AIA consists of four major elements:

- The As-Is Condition, which identifies and analyzes the current state of the existing information environment, including the problems and needs that have motivated the formulation of the information architecture.

- The Vision, which establishes the overall conception of the information architecture and describes the fundamental principles that the architecture is based on. One of those fundamental principles is that end-users should have direct, dynamic, integrated access to the information they need no matter where or in what form the source data resides. This principle is referred to as the *declarative paradigm* for information access.
- The Superstructure, which establishes the scope of the information architecture, defines the methodology used to classify the information entities that fall within that scope, and describes the characteristics of the entities comprising the architecture. The scheme developed for classifying enterprise entities is based on three fundamental categories or classes: *functional entities*, which include abstract entities such as policies, functions, directives, etc.; *information entities*, which include representational entities such as labor cost data, geometric models, process statistics, etc.; and *system entities*, which include physical entities such as people, computers and other equipment such as production machinery. These categories are consistent with the ANSI/SPARC and ISO three-schema database architecture [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a]. The three categories may also be applied to one another (including recursively) using a generative classification method. This produces a richer, more explicit set of categories for classifying selected entities in more detail. The methodology also addresses the life-cycle of information from its identification and definition to its eventual retirement and archival.
- The Implementation Plan, which defines an incremental and iterative project plan for actually realizing the architecture throughout the Alcoa enterprise. The long-term goal of the AIA is to logically integrate the information assets of the entire Alcoa enterprise world-wide.

#### Computer Aided Manufacturing - International (CAM-i) Computer-Integrated Enterprise (CIE)

Computer Aided Manufacturing - International (CAM-i), a consortium of commercial and defense companies, launched a project called the Computer Integrated Enterprise (CIE) on the heels of the ICAM FoF and ManTech information technology efforts (which are described under a separate subheading below). While some of the concepts in the original FoF architecture have been revisited and, in some cases, extended, CAM-i has yet to produce what can be described as an overall formal information architecture. Progress has been made however on a number of individual projects, which CAM-i member companies such as Westinghouse Electric Corporation (WEC), have picked up and applied as part of their internal enterprise framework and architecture projects.

#### European Strategic Programme for Research and Development in Information Technology (ESPRIT), European Computer Integrated Manufacturing Architecture (AMICE), Computer Integrated Manufacturing — Open Systems Architecture (CIM-OSA)

The European Community picked up the ICAM Factory of the Future (FoF) concepts and initiated their own project to extend the FoF work. The project was known as the European Strategic Programme for Research and Development in Information Technology (ESPRIT), European Computer Integrated Manufacturing Architecture (AMICE), Computer Integrated

## ITKM State-of-the-Art Review

Manufacturing — Open Systems Architecture (CIM-OSA) project [ESPRIT Consortium AMICE, 1989].

The CIM-OSA architecture starts out quite well. The basic view of the architecture is as a three-dimensional matrix. One dimension represents three categories — Generic, Partial and Particular — for classifying enterprise analysis and planning results (“models” in CIM-OSA terminology) by industry applicability. “Generic” means that a model applies to all or most enterprises; “Partial” to some subset such as the automotive, aircraft or metals processing industries; and “Particular” to a specific instance of an enterprise such as Northrop or Alcoa. A second dimension represents four views of the models: Function; Information; Resource; and Organization. The third CIM-OSA architecture dimension represents the evolution of the models across the systems development life-cycle using three categories: Requirements Definition Models; Design Specification Models; and Implementation Description Models. The CIM-OSA approach to developing the architecture is step-wise and iterative. Several factors or criteria must be considered in the development of the architecture: scope; environment; functions; models; and methodology. Up to this point, the CIM-OSA architecture is quite useful to support enterprise analysis and information integration. However, more depth is needed in a number of areas; for example, the classes of the enterprise need to be taken to the second and third order to provide context and meaning for many of the methods and tools required for enterprise analysis and information integration; consideration needs to be given to the life-cycle of an information entity rather than just to a systems development life-cycle; etc. The CIM-OSA architecture unfortunately brings into the remainder of its description what are clearly implementational issues. Specifically, much of the material concerns itself more with the definition, installation and operation of communications networks than with information architecture issues. These issues are not relevant to specifying an information framework or architecture.

## Integrated Computer-Aided Manufacturing (ICAM) Factory of the Future (FoF)

The United States Department of Defense (DoD) was one of the first organizations to identify the need for a framework or architecture to guide Computer-Integrated Manufacturing/Enterprise Integration activities. Almost fifteen years ago a major research and development initiative was launched by the DoD with the participation of approximately 75 major aerospace and defense contractors. That project was called the Integrated Computer-Aided Manufacturing (ICAM) program. ICAM went on to spawn several information technology-related research and development projects under the Air Force Manufacturing Technology (ManTech) program. The prime responsibility of ManTech was to transition new materials and processes out of the laboratory environment and into aerospace and DoD production facilities. Information technology was then added as a focal area to support these new materials and processes. ICAM also influenced the Industrial Modernization Incentives Program (IMIP) and Best Practices program by encouraging defense contractors to apply advanced information technologies to reduce cost and improve quality. One major product of the ICAM effort was an architecture, the centerpiece of which is referred to as the Factory of the Future (FoF) Conceptual Framework. The FoF architecture proposed a radically new concept for manufacturing — one in which an enterprise would be comprised of a series of semi-autonomous, but closely coordinated “work centers” including a: Marketing Center; R&D Center; Product Definition and Planning Center; Provisioning Center; Sheet Metal Center; Machining Center; Composites Center; Electronics Center; Assembly Center; and Logistics Support Center. These centers, operating somewhat like separate “mini-enterprises” were to be logically integrated into a single, unified enterprise through

an Integrated Information Support System (IISS). IISS was to utilize the emerging ANSI/SPARC three-schema database concepts [Tsichritzis and Klug, 1978] to accomplish this. A strategic plan or "roadmap" was created for the development of other critical technologies and detailed architectures for each center. The roadmap also addressed initial implementations of the centers at contractors involved in the product definition and/or product delivery of high-profile weapon systems such as the F-16, F-18 and B1-B aircraft, Blackhawk and Apache helicopters, and Trident submarine. In addition, projects were planned to develop tools and methods to support the analysis of existing "As Is" contractor environments and to aid in the planning and implementation of "To Be" Factories of the Future based on the concept of centers. The ICAM Definition (IDEF) suite of enterprise modeling methods and tools, and the ICAM Systems Development [life-cycle] Methodology (SDM) are examples of the results of these latter efforts.

It is true that the ICAM FoF architecture was not focused solely on enterprise information, and enterprise analysis methods and tools were only one part of the effort. However, ICAM must be viewed as a visionary effort directed at radically transforming the existing environment into a fundamentally new and improved environment, through one or more major paradigm shifts. ICAM recognized, however, that the move from the "As Is" to the "To Be" must be accomplished in an orderly, evolutionary manner. ICAM resulted in substantial benefits through improvements in both efficiency and effectiveness. It is worth noting that the FoF architecture was published approximately ten years ago. The technologies envisioned as being necessary to realize that architecture are, for the most part, today's state-of-the-art technologies. The final ICAM projects (for example, the Automated Airframe Assembly Program led by Northrop Aircraft Division) are just now being completed. Many aspects of the architecture and the associated technologies do not look exactly the way they were originally envisioned. The architecture, being dynamic, has evolved and adapted based on lessons-learned and in response to the availability and capability of various enabling technologies.

#### James Martin Information Engineering Framework

Like both the Zachman "Framework For an Information Architecture" and ESPRIT CIM-OSA (both of which are described under separate subheadings within this section), James Martin's Information Engineering Framework can be viewed as a matrix [Martin, 1990]. Martin's matrix has two columns representing data and activities (processes). There are also five rows addressing: Strategy - Technology Impact; Strategy - Enterprise Model; Business Area Analysis (BAA); System Design; and System Construction and Cutover (Implementation). These are very similar to the original six rows of Zachman's matrix, except Zachman splits system design into two separate categories for high-level or conceptual design versus detailed design. The similarities between the frameworks or architectures of Martin and Zachman are probably a result of the relationship of both men to International Business Machines (IBM) Corporation, and reflects the influence of numerous IBM modeling and systems design approaches and methodologies (including HIPO® and Business Systems Planning®, both of which were precursors to IBM's current methodology, the Applications Development Cycle or AD/Cycle®). Martin's framework has the same basic, intuitive appeal as Zachman's. Aside from the additional detail in both the original version and the extended version of Zachman's architecture, the primary difference between Martin's and Zachman's is in the area of relationships to CASE tools (which are described under a separate subheading in Section 3.2., Modeling Methods and Tools, of this document). Martin's framework is inherently oriented to a CASE tool environment and assumes that the output of the modeling and design efforts in the first four rows of the matrix will be used



to generate code as part of the fifth row of the matrix, namely System Construction and Cutover. Martin's framework is closely tied to IBM's AD/Cycle systems development methodology, which includes approaches to enterprise modeling and legacy database re-engineering, and which is based heavily on the use of CASE tools.

Zachman "Framework For an Information Architecture"

As far as specific information architectures go, this is the one that appears to receive the most use, or at least *mention*, in industry. The Zachman Information Architecture was officially proposed in 1990 by John Zachman of IBM. Zachman, with assistance from Steve Pyryemybida of Northrop Corporation, first developed the framework in 1987 as an informal, practical guide for information systems analysis and planning [Zachman, 1987]. It was originally described by Zachman and Pyryemybida as a "framework for an information architecture" — implying that much remained to be done to make it into a full architecture. Regardless of that, it is commonly referred to now as an information architecture. Like CIM-OSA and James Martin's architecture (both of which are described under separate subheadings within this section), the Zachman architecture is presented as a matrix. Zachman does not address some of the peripheral aspects addressed by CIM-OSA, but in the major area he does address, he goes into more detail. Zachman's architecture is more similar to Martin's than to CIM-OSA, only Zachman's provides considerably more depth than Martin's in several areas.

Zachman's matrix is two-dimensional; however emphasis is not only on the two axes themselves, but also on the cells of the matrix that represent the intersection points of the axes. There were initially three columns in the Zachman architecture representing focus areas: Data Description (Entity-Relation); Process Description (Input, Process, Output) and Distribution or Network Description (Node-Line). There were six rows representing levels of models based on differing business perspectives: Scope Description (and/or Objectives); Model of the Business; Model of the Information System; Technology Model (or Technology Design); Detailed Description (or Technology Definition); and Actual System (or Technology Implementation). Since the Zachman Architecture is not a formally-defined, published architecture with version control, there are many different versions with slightly different names or labels appearing for the rows and columns. Regardless of the source and the labels used, the meaning of the rows and columns remains largely the same.

The Zachman architecture has certain advantages. It is intuitive and practical, in a simple sense. The contents of the cells (e.g. Entity-Relationship Data Models, Business Function Flowcharts, Data Flow Diagrams, etc.) are all things with which most systems analysts and other information management professionals are already familiar. To a large degree, they represent the existing, unintegrated methods and tools typically utilized for enterprise analysis and systems development. What is missing, however, is any rigorous, formal, logical basis for categorizing things this way, for determining what specifically to include in the contents of the cells, and for identifying and explaining the relationships between various cells — in other words, there is no formal, underlying structure for the matrix. This has already resulted in substantial confusion among various parties attempting to utilize the architecture for specific projects in their environment.

The IDEF Users' Group (UG), with Zachman's permission and assistance, has modified and extended the original architecture<sup>1</sup>. In the IDEF UG Framework, there are six focus areas (columns): Why (Value, Motivation or Direction); Who (People, Organization); When (Time); How (Process); What (Data); and Where (Network, Location). There are five perspectives (rows): Scope Description; Business Description; System Description; Technology Constraint Description; and Detail Description. The [former] sixth row (Technology Implementation) is considered to be outside the architecture and is no longer included. The modifications and extensions proposed by the IDEF UG, as well as those proposed by Zachman himself, still do not appear to have completely resolved the basic problems of ambiguity. Zachman and others have also proposed to apply what they call "recursion" to the cells in order to derive additional extensions to the framework. This proposal is similar to the 2nd and 3rd-order recursions stated as being necessary under the CIM-OSA discussion above. In this case, however, Zachman et al are beginning with 30 cells (6 X 5) as primitives. With the lack of underlying formalism or logical rigor, it can be expected that this "recursion" will not result in clarification, but rather may introduce additional confusion.

### **3.2. Modeling Methods and Tools**

#### Binary and Elementary N-ary Relationship (B[EN]R) Modeling

Binary and Elementary N-ary Relationship (B[EN]R) approaches have their roots in the fields of artificial intelligence and linguistics. They deal with "semantic networks" (described under a separate subheading within this section) and other similar notions. In semantic networks, knowledge (information) is encoded as a set of nodes connected to each other by a set of labelled arcs. Nodes represent classes (also known as concepts or objects), situations, etc. Arcs represent relations (also known as relationships, links or associations) among nodes. Semantic models are most often used to depict natural language processing. That is unfortunate since it tends to slant their application to the representation of language utterances (i.e. form) rather than the meaning (i.e. content) behind the utterances. B[EN]R approaches were introduced by Abrial [Abrial et al., 1974], Senko [1977], Bracchi [Bracchi et al, 1976] and others. B[EN]R models contain: entities; entity-names; and relationships. B[EN]R models do not include a separate component for "attribute" at the primitive level. Attributes are represented as distinct entities which are then related to the entity or entities which they are "attributes of". This point merits some additional explanation.

The real difference between the way B[EN]R and E-A-R methodologies (described under a separate subheading within this section) treat attributes is that B[EN]R modeling simply does not allow entities to have other information associated with them unless that information is explicitly represented through the use of relationships to other entities of a lower-level nature (i.e. "attributes"). As an example, whereas an E-A-R model might contain an entity PERSON with the attribute ADDRESS, the B[EN]R methodology would force the modeler to define ADDRESS as a separate entity (albeit an abstract, lower-level entity) associated with the PERSON entity by the relationship LIVES-AT. Other, still lower-level entities such as STREET, CITY and STATE would be associated with the entity ADDRESS using one of several fundamental

---

<sup>1</sup>Zachman has proposed his own extensions to the original architecture. These are basically the same as the IDEF UG's extensions, except that the names and descriptions of what entities belong in each "cell" are much more clearly and logically derived in Zachman's version.

## UKM State-of-the-Art Review

association types (e.g. membership, composition, generalization, etc.). The basic idea of these approaches is to model the environment *explicitly* and *distinctly* using sentences that express *simple elementary propositions*, thus not introducing a specific grouping of these elements. While this is typical of semantic network-based approaches to analysis, this stands in stark contrast to many E-A-R models that look more like database definitions than information models. The grouping of attributes under particular entities in these E-A-R models is motivated more by database design considerations than by the semantics of the information being modeled. B[EN]R models were largely static at first; however, some have recently been enhanced to account for rules about the dynamics of the environment being modeled.

The Nijssen or Natural language Information Analysis Methodology (NIAM) [Gadre, 1987] is an example of pioneering work in B[EN]R modeling during the 1970's. NIAM is described in more detail under a separate subheading within this section. Perhaps the most advanced of the B[EN]R approaches is the Semantic Association Model (SAM\*), and its extension, the Object-oriented Semantic Association Model (OSAM), developed by Su [1983, 1986]. In addition to an abstraction association (which differentiates object types from their occurrences or instances), there are seven other associations: membership; aggregation; interaction; generalization; composition; cross-product; and summarization. Dr. Robert Meersman of Tilsburg University in The Netherlands is currently working on a superset of NIAM based on the object-oriented paradigm. He is calling this new approach NORM. Early references to the work described the acronym as standing for Natural Object Relation Methodology. Recent references show the acronym as standing for Normalized Object Relational Model. In any case the key point is that while the specification will be object-oriented, the assumed implementation will still be relational. The intent is to combine the best of both worlds.

## Business Entity-Relationship Model (BERM)

BERM [Jorgenson, 1991a and 1991b; Jorgenson and Walter, 1991] is an Extended Entity Relationship (EER) modeling methodology. EER's are described under a separate subheading within this section. BERM represents an attempt to incorporate some elements of the object-oriented modeling paradigm (which is also described under a separate subheading within this section) into the traditional Entity-Attribute-Relationship (E-A-R) model (which is again described under its own subheading within this section). BERM also attempts to introduce a set of standards into the application of E-A-R modeling, particularly with respect to developing the content of the models. This is intended to strengthen the rigor of such models, as well as to improve the productivity of the modelers. BERM was developed at Boeing and has been applied on several projects within Boeing.

Regarding the use of object-oriented modeling approaches: BERM applies the abstraction or type hierarchy to E-A-R models by introducing a categorization scheme that divides objects in the information model into: Proposition or Fact Classes (e.g. Machine); Proposition or Fact Types (e.g. Personal Computer); and Proposition or Fact (e.g. PC # 12345-67). This may be somewhat of an overcomplication of two basic object-oriented concepts. First, there are simply [super]classes/types and their subclasses/types — regardless of what level they are at in the class or type hierarchy. Second, there is the notion that classes may have specific instances. Information in a BERM model is also categorized as: Physical Class (i.e. physical objects); Process Fact Class (i.e. rules, functions-processes); Object Fact Class (i.e. assertions, axioms, attribution); and Construction Operator Class (i.e. logic). Entities or objects are typed first; then propositions or relations involving the object are typed; then based on rules or semantics associated with the entity

classes and/or the proposition class, validation is performed to ensure that a given rule or proposition can be applied to the given entity.

Regarding the standardization of model content: standard relation types, objects, etc. have been defined and put into a general "pool" to be utilized or applied by modelers. Rather than creating unique names for relations or objects — when they are in fact the same as, or similar to, other relations or objects — the modelers are encouraged to check the standard definitions and to use either an existing standard or a more specialized version based on an existing standard. This is on the one hand a methodological issue, but on the other hand uses the American National Standards Institute (ANSI) Information Resources Dictionary System (IRDS) Conceptual Schema (CS) architecture notion of Application Schemas [Burkhart et al, 1991] to provide modelers with at least a starting set of basic model contents applicable to one or more application domains. The layers of the IRDS CS are described in more detail in a complementary document to this State-Of-The-Art (SOTA) Review Document, entitled Needs Analysis and Requirements Specification For an Integration Toolkit and Methods (ITKM).

Accelerator, a commercially-available automated E-A-R modeling tool, is utilized to create and store BERM models. BERM has demonstrated that object-oriented and other AI concepts can and are being applied in existing enterprise modeling languages such as E-A-R. Although these hybrid methods, techniques and/or languages will undoubtedly make representation and integration of enterprise modeling languages more difficult, it indicates that there is a requirement for Integration Toolkit and Methods (ITKM) technology to accommodate various kinds of modeling languages, including extensions to existing, fairly stable modeling languages.

### Computer-Aided Software Engineering (CASE) and Re-Engineering

Computer-Aided Software Engineering or CASE tools are not directed at enterprise modeling in general. Rather, they are directed at the modeling or specification of application program logic and database structure using a set of programming and database constructs that are neutral with respect to any particular programming language or database definition and manipulation language. The objectives of specifying programming logic and database structure using CASE tools include:

- Better definition of application program logic and database structure through the explicit separation of logical entities and operations from the specific constructs used in any given programming or database language to express or implement those logical entities and operations. This layer of independent is intended to allow systems analysts and/or programmers to concentrate on the semantics of the domain being specified and to not get overly focused on or, in some cases, constrained by, the syntax of particular programming and database languages.
- Automatic generation of program code and/or database structure in any particular programming language (e.g. COBOL, C, etc.) or database definition and manipulation language (e.g. IMS® DL/1®, DB2® SQL, etc.). This is accomplished by translating (in the sense of compiling) the CASE specification language constructs into specific programming and/or database language constructs using a mapping between the two sets of constructs. Programming and database syntax rules, and in many cases efficiency algorithms, may also be included in the CASE tools so that the program code and

database schemas that are generated are well-structured and operate efficiently in various target implementation environments.

CASE tools are intended to reduce the errors that occur between specification and program and database development by making more direct use of the specification itself. When making changes to a program or database during a lifetime of maintenance, changes are made to the specification and the program or database is regenerated. As long as this procedure is followed, the specification and the programs and databases will always be in synchronization. The specification can serve as documentation throughout the life of the system. The specification in this case must be rigorously and formally defined. Configuration management is still a major task in a CASE environment. Many CASE tools have built-in support tools to assist with configuration management. However, the programs and databases do not have explicit access to their logic or structure (as represented in the specification) while they are running or being accessed and, therefore, cannot be self-referencing or self-modifying.

Each CASE tool typically has its own programming and database constructs. In fact, one software vendor (Systematica Limited) markets a CASE-related product called the Virtual Software Factory (VSF)<sup>®</sup> that allows users to define their own CASE constructs and tie them to several sets of programming language constructs provided by the vendor to create their own custom CASE tool environment. Unfortunately, specifications developed in one CASE tool environment cannot be easily ported to run in, or integrate with, those of another CASE tool environment. This problem was specifically pointed out in the ITKM industry survey (refer to Section 2. above). CASE tool vendors have also been hearing about the problem from their customers with increasing frequency. Several efforts are already underway to integrate and extend CASE tools in order to eliminate, or at least mitigate, this problem. These efforts include: a new committee formed under the American National Standards Institute (ANSI) X3 - Information Processing Systems, Standards Planning and Requirements Committee (SPARC), known as X3H6 CASE Tool Integration Models (CTIM) and chartered with developing standards for CASE tool integration [Note: the name originally used by this group during its formation was CASE Integration Services (CIS)]; the Electronics Industry Association (EIA) CASE Data Interchange Format (CDIF) project also under ANSI (but not under X3/SPARC); DARPA's Software Technology for Adaptable Reliable Systems (STARS); and an effort known as Standard Data Access Interchange (SDAI) being conducted as part of the Product Data Exchange using STEP (PDES)/STandard for the Exchange of Product data (STEP) project of the International Organization for Standardization. While these tool integration efforts may relate more specifically to CASE tools, they appear to be addressing many issues common to model integration in general (or at least to *information model* integration). This is due at least in part to the fact that many proponents of CASE tools consider them to be complete enterprise modeling tools, and are constantly extending both the definition and capabilities of CASE tools to support this notion. The capabilities have been considerably slower to expand than the definition. Improving the graphics-based modeling user interfaces at the front-end of existing CASE tools is, however, a major focus of many CASE tool vendors. Also, many of the work tasks encompassed in the CASE tool integration projects and the kinds of issues being addressed as part of those efforts overlap to a considerable degree with the ITKM and ANSI X3H4 Information Resources Dictionary System (IRDS) efforts. The logical architecture for ITKM and the IRDS Conceptual Schema (CS) standards effort (which includes a Defining Schema, a Normative Schema, and various Modeling Schemas and Application Schemas) are supposed to be generic enough to handle all types of enterprise models, including those produced through the use of CASE tools. The

approach would be to initially address static models, then dynamic models, followed by higher-order models, and finally methods and tools that relate more closely to systems development, such as CASE tools. When CASE tools are the subject of the integration effort, the generic syntax and semantics of CASE tools would be "re-represented" in (in other words, *subsumed into*) the Normative Schema, thereby producing a Modeling Schema content module for CASE tools. If various CASE tool vendors have done a good job of integrating among their tools using their own neutral [CASE] model, it will be all the easier to represent that model in a Modeling Schema using the Normative Schema — including a Conceptual Schema, and mappings to External Schemas (graphical user interfaces) and Internal Schemas (vendor-specific tools). This would then provide the means to access and integrate various CASE tools within an IRDS or ITKM system.

As stated earlier, the CASE tools described above have typically been thought of as systems development tools, rather than broader enterprise modeling tools. Some people now refer to these traditional CASE tools as "lower-CASE". They then describe a new, expanded aspect of CASE tools referred to as "upper-CASE". Upper-CASE tools would be focused on enterprise modeling in general (or at least on *information modeling*) and would have primitive constructs associated with modeling or representation, rather than specifically with programming logic or database structure. Upper-CASE tools could be used to represent entities and processes at higher logical levels of the enterprise. Many of these higher-level entities and processes would not be translated or generated into program logic or database structure — at least not in the near future. Upper-CASE would allow the overall enterprise domain to be modeled, thereby providing both an enterprise model useful in its own right and a context for individual, lower-CASE application models. A number of CASE tool vendors state they are working on upper-CASE, but there are no CASE tools on the market currently that can really be described as addressing upper-CASE. One approach to realizing upper-CASE is to interface existing lower-CASE tools to existing enterprise modeling tools. For example, Oracle Corporation's CASE\*® has recently been interfaced with IntelliBase, Inc.'s RIDL\*™, a NIAM-based data or information modeling tool. Another approach is to directly extend the capabilities of existing CASE tools.

Some CASE tools are coupled with or include utilities to re-engineer existing program logic and, to a lesser degree, database structures. In the case of program logic, these utilities start out working backwards, in the sense that they parse existing, legacy program logic and create a CASE-based specification or model using the CASE constructs. This model can then be directly used to re-generate more efficient code in the same language (for example, modular structured COBOL where "go to's" existed before), or to generate code in a new language (for example, C code to replace FORTRAN). Oftentimes systems analysts or programmers modify the model before generating new code. This provides them an opportunity to correct errors in the old logic, make the logic more efficient, and/or add new logic to reflect changes in the application requirements since the time period when the program was last modified. Similar utilities can be run against existing database structures (i.e. database schemas and calls embedded in program logic) to create logical database models. The logical models can then be used to re-generate a better-structured database using the same database management system, or to migrate from one database management system to another (for example, from IBM's DB2 relational database to Oracle Corporation's ORACLE® relational database, or from IBM's IMS hierarchical database to IBM's DB2 relational database). In practice, there are a number of serious hurdles in translating from one database form to another, particularly from hierarchical to relational form [Dement and Woodruff, 1988]. Current tool capabilities in this area are quite limited. IBM's Applications Development Cycle (A/D Cycle) is heavily oriented to coupling re-engineering tools such as the Bachman tools with IBM and third-party CASE tools. Vitro's Corporation Vitro Automated

Structured Testing Tool (VASTT), including an Intermediate Language Representation (ILR) as its programming language-neutral form, is another, somewhat more limited example of a re-engineering tool for legacy programs.

### Concept Maps

Novak and Gowin [1984] have proposed using a Concept Map as a schematic device for representing a set of concept (i.e. object) meanings embedded in a framework of propositions (i.e. expressions of relationships or other statements involving the object). Concept Maps are comprised of *concepts* (named using concept labels) related to other concepts by means of *linkages* (named using linking words). The concept-linkage-concept notion is basic to semantic networks. GUIDE [1991] refers to semantic networks as using the "thing-link-thing" construct to represent knowledge. Semantic Networks in general are described under a separate subheading within this section.

Concept Maps are based on three major ideas in what is known as Ausubel's cognitive learning theory: 1) Cognitive structure is hierarchically organized, with more inclusive, more general concepts and propositions superordinate to less inclusive, more specific concepts and propositions; 2) Concepts in cognitive structure undergo progressive differentiation, wherein greater inclusiveness and greater specificity of regularities in objects and events are discerned and more propositional linkages with other related concepts are recognized; and 3) Integrative reconciliation occurs when two or more concepts are recognized as related in new propositional meanings and/or when conflicting meanings of concepts are resolved.

Because meaningful learning (acquisition of knowledge) proceeds most easily when new concepts or concept meanings are subsumed under broader, more inclusive concepts, Novak and Gowin advocate that Concept Maps be organized hierarchically, where appropriate; that is, the more general, more inclusive concepts should be at the top of the Concept Map, with progressively more specific, less inclusive concepts arranged below them. Hierarchically constructed Concept Maps help to subsume more specific meanings into larger, more general concepts. The use of hierarchies and decomposition is inherent in the process of analysis, and is found in many modeling methods and languages (including traditional structured analysis and design techniques). It is not inconsistent with the semantic network approach, and in fact, merely represents one meaningful way of organizing or relating nodes in the network. Concept Maps externalize the individual's propositional frameworks and can therefore be used to check on faulty linkages or to show that relevant concepts may be missing. Concept Maps, used as a tool for negotiating meaning, can make possible new integrative reconciliations that in turn lead to new and more powerful understanding.

There are, however, some problematic issues which arise when a hierarchical organization of concepts and propositions (such as Concept Maps) is utilized. Novak and Gowin state that written or spoken messages are necessarily linear sequences of concepts and propositions. In contrast, knowledge is likely stored in our minds in a kind of hierarchical or holographic structure. When we generate written or spoken sentences, we must transform information from a hierarchical to a linear structure. Conversely, when we read or hear messages, we must transform linear sequences into a hierarchical structure in order to assimilate them into our minds. Maintaining the semantics intended by the speaker or writer while parsing natural language sentences is an extremely difficult task. Because it is so difficult, some argue that the concentration should be on the form of the sentence (its syntax) rather than its content or semantics. Limiting ITKM to "form" over "substance" would do little to improve the state of enterprise information from what

it already is today. This area is the subject of considerable research by Sowa [1984; 1991a, 1991b and 1991c] and others.

The basic form of Concept Maps is circles representing concepts with lines between them representing relations or linkages. As noted above, the general form is hierarchical, but there can (and should, according to the suggestions by Novak and Gowin) be “cross linking lines” that go outside of or around the general hierarchical structure to indicate “bigger picture” contextual relationships between groups or clusters of concepts and relations. These lines actually go from one fairly general concept to another, somewhat orthogonal, but equally general concept to indicate that the hierarchical nature of the diagram introduces “branches” that may in themselves be related in some ways. The general direction of an arrow is assumed to be downward (in correspondence with the hierarchical decomposition, which is based on moving from more to less general concepts); upward arrows on lines are used to indicate when this is not the case.

The biggest problem with Concept Map syntax and basic model semantics is a lack of rigor or regularity, particularly in semantics, but also to some degree in syntax. This looseness, rather than providing necessary flexibility, introduces ambiguity and opens up the possibility of missing important semantics in the model. Concepts can be either objects (in the sense of static information entities, e.g. “Rain”, “Water”) or events (i.e. “Raining”). The relation between any two concepts is supposed to be labelled using a linking word. The problem is that in some cases the linking words is a very weak verb like “has” or “is”, sometimes it is a preposition like “by”, sometimes it is a phrase like “such as”, and sometimes it is a stronger verb or verb phrase like “[is] made of”, “[is] kind of”, “causes”, “influences the selection of”, etc. Many of the linking words could be expressed as concepts — as events in the case of most verbs, but in some cases as either objects or events. For example, “rains” might be modeled as a relation between “Sky” and “Water”, instead of making “Raining” or “Rain” an event or object concept and then having relations between that concept and both “Sky” and “Water”, as in “Water Can Appear As Rain” and “Rain Falls From Sky”). In the case of “such as”, there is already a syntax convention that allows you to give an example (i.e. specify an instance) of a concept through a label placed outside and directly underneath the concept (e.g. if the concept were “Dog”, it might have the label “Terrier” underneath it, or it might have “Sparky”). [Note: alternately, Terrier might also be modeled as a concept, appearing below “Dog” on the map with a relation labeled “kind of” or “type of” between them. However, Concept Maps do not deal with type hierarchies in the sense of class-subclass or “class as instance” distinct from the basic “is a” abstraction relation holding between a class and an individual instance of that class].

Also, although the notion of clustering is used to relate groups of concepts with other groups of concepts within a given Concept Map, some of the examples in the methodology textbook seem to show linkages between two concepts that only have meaning when other higher-order concepts back up the tree are considered at the same time — and clustering was not used in these cases. That is to say, the relationship between just the two concepts at hand really has no meaning unless it is considered in the broader context of one or more ancestor concepts, but there is no explicit indication of this situation in the Concept Map.

In some examples, the same concept is shown as a parent (superordinate) in one Concept Map and as a child (subordinate) in another “equivalent” Concept Map. If the change in the relation (directionally et al) is represented explicitly that might be O.K., but it just seems to be a gloss to, in effect, flip what is supposedly a hierarchically ordered diagram and say it is still equivalent to the original. This seems to be intended more as a point about navigation (i.e. how one works one’s way to a given concept) or presentation, than a semantic point about how a concept that is a subordinate in one relation can be a superordinate in an equivalent relation. Navigation



and the relation itself are two very different things. There may be many paths or routes to take to get to an object or concept based on the relations it stands in and the other concepts in those relations. But each of those relations has a certain logical ordering to it and that cannot or should not be arbitrarily altered with making corresponding changes in the relation itself. For example, "Is Part Of (Tire, Car)" and Has Part "(Car, Tire)" are two differently ordered variations of the same relation, but one when a switch is made from one to the other, it is noted by the change in the relation name as well as the reordering of the objects in the relation. That is also a different issue than stating that one can get to the concept "Tire" by knowing that it "Is Part Of a Car", "Is Made of Petro-Based Material", "Has Tread", etc.

Novak and Gowins note that the most common problems with Concept Maps are usually signaled either by a linkage between two concepts that leads to a clearly false proposition or by a linkage that misses the key idea relating two or more concepts. The former seems like it involves relationships which simply do not hold (i.e. they are simply false) or are incorrectly identified (e.g. there is a relation between "Water" and "Lake", but that relationship is not properly described as "Lives In"). The latter could be a valid relationship between two concepts, but it is just not the most appropriate relation given the context being modeled. That could include relationship names which are technically correct, but are too vague to provide much useful meaning. For example, "Water Is In Lake" may mean many different things. It could be intended to indicate that the water is physically located in the lake versus someplace else, for example if it had been pumped into the fields for irrigation. It could mean that the lake has water in it, rather than being a dry lake (which, in California at least, is probably more often the case). Or it could even mean that water is one of the things that make up or comprise a lake, with others things being algae, fish, reeds, tree stumps, etc.

Some general guidelines are provided for a method of using Concept Mapping: Start with a concept or small set of related concepts. These might be extracted from oral or written material. Identify related concepts. Organize the concepts from most general to most specific [Note: Identifying and organizing the concepts can be performed in parallel in an iterative manner]. Label the linkages (relationships) between concepts using linking words. This is not to suggest that one and only one correct linking word exists between any two concepts. Often there are two or three equally valid ways to link two concepts, but each will have a slightly different connotation. For example, if the concepts "Water" and "Ice" are linked with words such as "Can Be", "Becomes" or "Sometimes Is", each proposition thus generated has a similar but not identical meaning. It is necessary to isolate concepts and linking words and to recognize that although both are important language units, they have different roles in conveying meaning. Sometimes it is useful to apply arrows to linking lines to show that the meaning relationship expressed by the linking word(s) and concepts is primarily in one direction. As noted above, hierarchical maps ordinarily imply relationships from higher-level concepts downward to subordinate ones. To reduce clutter, the convention used is that no arrows are shown unless the relationship indicated is something other than a superordinate to subordinate linkage between two concepts. Finally, identify the larger-context (i.e. "big picture") linkages between clusters of concepts on the same Concept Map. These are cross-linkages and, like other linkages, should be labeled with linking words.

In summary, Concept Maps represent an informal method that leaves much open to subjective opinion and personal preference. Concepts Maps may be useful, however, as a "brainstorming" methodology to be applied during the very early stages of a modeling effort.

### Concept Propositional Analysis (CPA)

Concept Propositional Analysis (CPA) is a method for evaluating interviews with subject experts about their field or domain of expertise, as well as descriptions or models produced based on those interviews. For example, CPA might be used to interpret models of the material acquisition, handling, storage and delivery processes based on interviews with personnel from the Materiel function or organization within the target enterprise. CPA is based on the psychological notion that the meaning of any concept for an end-user or domain expert is shown by the set of propositions that the end-user or domain expert constructs incorporating the concept.

CPA is often used with the Vee Knowledge Acquisition Methodology (which is described under a separate subheading within this section). CPA/Vee, when used as a tool for interpreting responses in an interview, is read from knowledge claims (statements made by the interviewee) to the objects and events [embedded in the questions] presented, in order to construct inferences about concepts, principles, and theories held by the interviewee. Based on Gowin's Vee analysis, it is thought that the propositions an interviewee makes in response to questions are the interviewee's knowledge claims based on his or her interpretation of the objects or events and data provided. Given a fixed set of events or objects and questions, the concepts, principles, and perhaps theories the interviewee is using to make a knowledge claim can be inferred.

The underlying premise of the CPA technique is that subject-matter experts should not really be allowed to develop their own models or that, in any case, they cannot be counted on to directly express the underlying concepts, relations, rules, theories, etc. they use to make decisions and perform other tasks as a domain expert. The idea is that a modeler captures a mid-world or "user view" model based on interviews with the expert, and then analyzes and interprets that model to capture and model the real, underlying concepts, relations, rules, theories, etc.

### Conceptual Graphs (CG)

Conceptual Graphs (CG) [Sowa, 1984] is a modeling form based on Typed Predicate Calculus. CG draws upon the work of Charles Peirce (1883) in Existential Graphs and on the notion of semantic networks as used in artificial intelligence. Semantic Networks are described under a separate subheading within this section. While CG utilizes a formal language for expressing propositions or assertions, the language is easier to read than traditional Interpreted Predicate Logic (IPL). IPL is described briefly under a separate subheading within this section. CG is particularly geared toward capturing propositions from natural language statements (i.e. representing linguistic relations). In addition to the ease of capturing propositions from natural language statements, CG has another strong point: the graphical and linear forms are equivalent [Note: this is somewhat aided by the fact that the graphical form is not purely graphical, i.e. it relies on text in cases where graphical symbols would be confusing or ambiguous]. This is in contrast to the linear or textual language EXPRESS, the graphical form of which (EXPRESS-G) allows only the expression of a limited subset of the semantics representable in the linear form (EXPRESS is also described under a separate subheading within this section). The CG form can also be easily utilized for inferencing. There are currently few, if any, automated tools to directly support CG. However, a project is being formulated to develop a general CG software tool. That project would involve a number of universities worldwide.

CG is used frequently to represent abstract data types, with emphasis on constraints among abstract data types. This is accomplished specifically using the notion of type hierarchies found in the object-oriented paradigm (which is described under a separate subheading within this section).

However, CG can be used to represent instance-level assertions or constraints, as well — or mixtures of abstract and instance-level constraints (for example, where most constraints apply at the abstract or class level, but others apply only to specific instances). Type definitions are specified through Lambda expressions (i.e. they are based on the Lambda calculus). Existential (“there exists an X...”) and universal (“for all [for every] X...”) quantifiers may be specified. Sets of referents may also be specified.

The basic graphical modeling form uses boxes for concepts (which are objects or abstract data types), and lines as relations which link concepts. Concepts may be “data objects” (like Person or Horse), or they may be processes, events or states (like Lead or Leading, or Sit or Sitting). In this form, nouns and verbs are treated uniformly. In other words, to represent the statement “Person Leads Horse to Water”, concepts would be modeled for: Person, Lead (or Leading), Horse and Water. Person relates or stands in a relation to Lead, as do Horse and Water. So relation lines would connect Person to Lead, Horse to Lead and Water to Lead. This is syntactic. A relation line may be separated into two parts, with a circle separating the two segments of the line. Within the circle, the relation or role name may be specified. The relation between Person and Lead would be Agent, whereas the relation between Horse and Lead would be Patient, and the relation between Lead and Water would be Destination. These are semantic. An alternative form would be to treat Lead as a bundled relation. This requires producing two separate but related graphs, with one containing Person, Horse and Water all related [syntactically] to a concept called Lead, and a second relating the concept Lead [semantically] to the roles or relations for Agent, Patient and Destination. The two-graph form is a little more difficult to manage, since both graphs must really be considered together to fully understand the proposition being represented. However, using a semantic network of pointers or a similar representational mechanism in an automated support tool would largely mitigate this issue and would allow clean separation or unbundling of semantics.

Boxes (i.e. objects) can be further quantified (e.g. “any 2 horses”, “[specifically] the horses named Trigger and Silver”, etc.). They can also be referred to by other graphs in the overall model (which is, in effect, a semantic network of individual, related Conceptual Graphs). In this case the objects are said to be “co-referent”. Circles (i.e. relations or roles) cannot be quantified or specifically referred to (although the same relation or role can, of course, appear numerous times in the overall model). Graphs may also be nested. Nested graphs may be expressed textually using nested “IF THEN” rules. Graphs may be negated by putting a box around the entire graph and placing a “not sign” next to it.

To-date, CG has predominantly been used to represent static models, i.e. static objects and propositions or assertions about them. This is not to say that some of the propositions or assertions cannot involve actions or states (for example, “The Cat Sat On the Mat” or “The Cat Is Sitting On the Mat”) — in this sense CG even allows differentiation of states from actions). Rather, this is only to say that the full dynamic semantics of a process are not typically modeled using CG. The basic constructs of dynamic or process models (e.g. operation, sequence, antecedent, consequent, etc.) could be represented using CG’s, but the language does not directly lend itself to the modeling of process semantics. In fact, current practice does not necessary reflect the full capability of CG for modeling. Per GUIDE [1991], CG is both expressive and flexible. Definition can take place at any level of detail. Conceptual Graphs can represent everything in predicate calculus, including modal, intensional or higher-order logic. Because of this, CG forms a well-developed notation for the schemata of situation semantics. Situation semantics provide for the representation of an event or state of affairs independently of the contents of the event or state of affairs. This allows for reasoning about the event itself (e.g. Who said it

happened? When? Where?) separately from reasoning about the contents of the event (e.g. What Happened?). For example, the proposition "John saw the punch press come down on the operator's hand" really deals with two things: John's seeing of some event, and the event itself (i.e. the punch press coming down). John definitely saw something, but it might turn out that the punch press did not come down on the operator's hand. It may only have appeared that way from where John was standing. Many other events may have occurred because of what John thought he saw. He might have called 911. He may have gotten excited or upset. The notions of agents, belief states, and other aspects of modal logic are important to situation semantics and can be modeled, to at least some degree, using CG.

In application, the CG language does not come with a large set of standard language constructs such as those found in EXPRESS, although several primitive constructs are described in the primary textbook on CG [Sowa, 1984], and a CG electronic mail network informally shares some higher-level basic constructs among CG users. The primitives of the CG language could, of course, be used to define the higher-level constructs (and this is typically the case with constructs shared over the e-mail Network). Certain CG modeling syntax and conventions provide some of the capabilities of these constructs, such as defining domains and ranges (including particular instances) and lexical referents. While both CG and EXPRESS may be described in a broad sense as constraint languages, it should be noted that CG is purely a conceptual information modeling tool, whereas EXPRESS can quickly be taken to the point where it looks like a Problem Statement Language (PSL). In other words, EXPRESS crosses far over into the domain of systems design and development, specifically database definition and program logic structuring.

Sowa has proposed using CG in conjunction with the Entity-Attribute-Relationship (E-A-R) methodology and/or the NIAM methodology (both E-A-R and NIAM are described under separate subheadings within this section). E-A-R or NIAM would be used for high-level, conceptual modeling (to identify major entities and establish context) and CG would be used for specifying the details of particular lower-level relations of interest and for expressing other propositions involving the identified entities.

Given the nature of CG, it appears to be suitable as the basis for a Normative Language for representing the "meta modeling" constructs in Layer 2 - Normative Schema of the American National Standards Institute (ANSI) Information Resources Dictionary System (IRDS) Conceptual Schema (CS) architecture [Burkhart et al, 1991]. In fact, the ANSI IRDS CS Task Group has recently decided to use CG as the basis for its initial Normative Language for purposes of expressing, in an ANSI standard, the constructs necessary for the IRDS CS to support model integration. The neutral, meta model constructs expressed in CG form would then be mapped (via IRDS CS Layer 3 - Modeling Schemas) to specialized constructs in other enterprise modeling languages such as NIAM, EXPRESS, IDEF0 and/or IDEF1/IX, SADT® Acti-Grams and/or Data-Grams, etc. which may be more appropriate for developing particular kinds of models (e.g. data models, process models, logical database designs, etc.) at IRDS CS Layer 4 - Application Schemas. For readers interested in exploring conceptual schema issues further, the layers of the IRDS CS are described in more detail in [Burkhart et al, 1991] and in a complementary document to this State-Of-The-Art (SOTA) Review Document, entitled Needs Analysis and Requirements Specification For an Integration Toolkit and Methods (ITKM) [refer to Section 3.3.2., Requirements Specification For Architectural Elements or Components Of an ITKM, subheading on "Layers Of Schema [Language] Definition For Model Integration", pages 48-50].

CG can be directly mapped *to* and *from* natural languages, or more accurately, stylized natural languages. However, the "from" part is extremely dangerous to take at face value: as Sowa himself states, parsing a natural language sentence and transforming it into formal logic does not

## ITKM State-of-the-Art Review

guarantee that the intension (read, the “real meaning”) of the sentence will be captured. Sowa notes a major role for humans in this effort. It should not be done automatically by an automated modeling tool, or if it is, the result should not be accepted into the production knowledgebase without the review and approval of a human.

### Data Modeling Versus Information Modeling

First, to clarify the difference between the terms *data* modeling and *information* modeling: according to the International Organization for Standardization (ISO) [1987b], data modeling concentrates on the form of the data that is modeled. That is to say, structural forms for data are defined that are convenient for storage and/or manipulation in a computer. In particular, update possibilities are optimized, although access path structures, important for retrieval, are often emphasized as well. Information modeling concentrates on the meaning (semantics) of “data”, i.e. the aspects that make “data” into “information”. Semantic models or information models are considered to be independent of, but of critical importance to, the data models that describe the representation and storage forms of the information. In the terminology of the ANSI/SPARC and ISO three-schema database architecture [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a], data models represent the implementational or Internal Schema view of data, whereas semantic models represent the abstract or Conceptual Schema view of data. Information models formally define exactly what the data represents, which semantic rules and constraints exist, and what information manipulation actions are needed by the user. The International Organization for Standardization (ISO) [1987b] identifies several methods for information modeling, including:

- Abstract Data Types,
- Binary Relationship Models,
- Conceptual Graphs,
- Deep Structure Sentence Models,
- Entity-Attribute-Relationship (E-A-R) Models,
- Function, Process or Operation-Oriented Models,
- N-ary Relationship Models,
- Network Models (Including CODASYL),
- Object-Role Models,
- Process-Interaction Models,
- Relational Models,
- Semantic Networks or Nets, and
- Set Theoretic Models.

ISO groups these methods into three broad categories based on their fundamental concepts and characteristics. Since some of the methods above might fall into more than one of the three categories (i.e. they possess some aspects of two or even all three categories), there is no firm mapping of the methods to the categories. The categories of approaches are:

- Entity-Attribute-Relationship (E-A-R),
- Binary and Elementary N-ary Relationship B[EN]R), and
- Interpreted Predicate Logic (IPL).

Each of these is described in more detail under its own subheading elsewhere in this section of this document.

### Entity-Attribute-Relationship (E-A-R) Modeling

According to the International Organization for Standardization (ISO) [1987b] and Vernadat [Vernadat and Kusiak, 1988], Entity-Attribute-Relationship (E-A-R) approaches are based on the conceptual work of Bachman, Chen and others. Specific modeling approaches or methods have been put forth by: Chen [1976] (the Chen Entity-Relationship Model); Codd [1979] (RM/T, for Relational Model/Tasmanian); and others. The IDEF1 and IDEF1X modeling methods developed by the Air Force under the Integrated Computer-Aided Manufacturing (ICAM) program fall into the E-A-R category. The Chen Entity-Relationship Model and IDEF1X are probably the two most widely used E-A-R modeling methods. The ITKM State-Of-The-Art (SOTA) survey, as well as other SOTA reviews, have found E-A-R to be among the most widely used methodology for data and [limited] information modeling, as well as one of the most widely used modeling forms in general. GUIDE [1991] states that the reason E-A-R have been so widely accepted is that the models draw upon the basic "thing-link-thing", particularly "noun-link-verb", relationship which most humans find natural for modeling.

The primary components of an E-A-R model are: entities; relationships between entities; attributes of entities; and values of attributes. The overall domain or universe of discourse of the model may also be specified in some form. All E-A-R models have the notion of identity, i.e. some way to refer to a specific entity. Some E-A-R modeling methods that are oriented more specifically to logical data modeling and/or physical database modeling include the notion of a key attribute. Key attributes establish a logical and/or physical access path into an entity and often also serve as logical and/or physical linkages between related entities. The domain and range of attributes may also be specified in the model. Attributes, while almost always treated as separate components or constructs of E-A-R models are sometimes described as really being "characterization relations". Some extended E-A-R modeling methods allow relationships themselves to have [limited] attributes.

The relationships represented in E-A-R models have traditionally been binary, although some extended versions of E-A-R modeling methodologies allow  $n$ -ary relationships among entities. Individual binary relationships are frequently further quantified through a construct called "cardinality". In precise terms, cardinality refers only to cases where specific cardinal numbers are used to quantify the relation (e.g. 1 Purchase Order can have no less than 1 nor more than 10 Purchase Order Items). Stating that there may be many Purchase Order Items for one Purchase Order would be qualification, but not really cardinality. However, the term cardinality is commonly used in the modeling community to refer to quantification in general (e.g. many, several, etc.), as well as strict cardinal quantification. Regarding "many-to-many" relations, there is an age old question of whether the relation must be normalized (a process in which intermediate entities are introduced to further decouple the many-to-many relations and reduce it ideally to a purely quantifiable relation). Many-to-many relations do not pose a logical data or information modeling problem. They are, however, impossible to implement using relational databases. Since logical E-A-R models often get implemented using relational databases, and since the general E-A-R form is frequently used or adapted for physical database modeling as well — many practitioners stipulate that all relations in E-A-R models should be normalized to at least 3rd Normal Form (essentially no many-to-many relations).

## ITKM State-of-the-Art Review

Several E-A-R models distinguish between Kernel (Independent) versus Dependent Entities. Independent entities can exist on their own in the model (or at least be identified uniquely), without any other entity in the model. Dependent entities by their nature depend on one or more other entities in the model for their existence (or at least, identity). A common example involves a Purchase Order and a Purchase Order Item. Most enterprises allow a Purchase Order to exist (at least certain kinds of Purchase Orders at certain points in the procurement process) without any Purchase Order Items. Purchase Order in this case would be modeled as an independent entity. However, the inverse is not allowed in most enterprises, i.e. a Purchase Order item cannot exist without a Purchase Order. Purchase Order Item would then be modeled as a dependent entity. In terms of "identity" versus "existence", it could technically be said that a Purchase Order Item could not be *identified* without knowing the Purchase Order on which it is an item [Note: this is not to say that physical or even logical access would necessarily require first accessing the Purchase Order entity to get to the Purchase Order Item, but it does indicate that identifying the Purchase Order Item requires knowing the Purchase Order. For example, the key attribute of the Purchase Order Item would likely be a concatenated key attribute that includes the Purchase Order key attribute (Purchase Order Number, for example), as well as some unique component (the Purchase Order Item Number itself)]. Because "identity" in this sense is so closely allied with keys and other logical and physical access issues associated more with the implementation of relational databases than with pure logical data modeling or information modeling, we prefer "existence" as the determinant of independence or dependence. "Identification" can be handled with other syntax conventions, as described under the subheading of IDEF1/IDEF1X within this section. IDEF1X also supports the modeling of independent versus dependent entities quite well (using squared versus rounded boxes). Some people consider this a feature of extended E-A-R models, but we consider it a differentiating feature among E-A-R models.

E-A-R models are oriented towards the definition of static semantics. It is, therefore, difficult, if not impossible, to fully model the dynamics or behavior of information in environments that experience high-levels of complex information interactivity and/or changes in information structures or relationships. In other words, it is difficult to use E-A-R models to model the processes that act on the entities and the states that the entities have before and after the process. Also, it is difficult to model relationships that are "optional" or which occur only under certain conditions, since the optionality rules or conditions cannot really be expressed using basic E-A-R constructs. E-A-R models are themselves "static" in that they represent only a point-in-time or snapshot view of the data or information in some universe of discourse such as a manufacturing enterprise.

## EXPRESS and EXPRESS-G

EXPRESS [Schenck and Spiby, 1991] is a conceptual schema language (as defined by the International Organization for Standardization (ISO) [1987a]). It is being developed by ISO TC184/SC4/WG5 in support of the ISO Standard for the Exchange of Product data (STEP) project. As a conceptual schema language, EXPRESS is intended to provide for the specification of the objects belonging to a universe of discourse (UOD), the information units pertaining to those objects, the constraints on those objects and the permissible operations on those objects. The UOD in this case is those aspects of product definition necessary to develop STEP. EXPRESS is a formal specification language for expressing a standard information model (information modeling is described under the subheading of "Data Modeling Versus Information Modeling"

within this section). EXPRESS consists of language elements which allow an unambiguous data definition, with the constraints on the data clearly and concisely stated. A future version of EXPRESS is intended to address process modeling as well. Currently, however, EXPRESS does not include language elements which allow input/output, information processing or exception handling to be represented. EXPRESS is not a programming language or a database language. However, it draws upon several well-known and widely-used programming languages including Ada; C; C++®; Modula-2; Pascal and PL/1, as well as the database language SQL. EXPRESS is driven by a number of requirements including:

- The size and complexity of STEP demand that its specification language be parsable by computer, not just humans. Therefore, EXPRESS must be a formal, rigorous specification language that lends itself to machine interpretation, automated consistency checking and view generation.
- The diverse material encompassed by STEP must be partitionable into smaller working sets. The STEP schema, created using EXPRESS, will serve as the basis for this partitioning.
- The language must focus on the definition of *entities*, which are the things of interest. Definition must be in terms of *data* and *behavior*. Data represents the properties by which an entity is realized, and behavior is represented by constraints and operations on that data. EXPRESS allows specification of entities, attributes and relations, as well as description of the constraints and permissible operations on those entities, attributes and relationships. Because of this, EXPRESS may be considered as somewhat of a hybrid between traditional Entity-Attribute-Relationship (E-A-R) models (which are described under a separate subheading within this section) and constraint languages.
- The language must avoid, as much as possible, specific implementation views. EXPRESS, as noted above, is a conceptual modeling language following the ANSI/SPARC and ISO three-schema database architecture [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a]. Therefore, it is neutral with respect to particular physical implementations.

The basic form of an EXPRESS model is textual (or linear). Model semantics are provided in the form of declarative statements or sentences using EXPRESS syntax and a large number of standard EXPRESS constructs. A group of related declarative statements comprises a block, which can be thought of a submodel of the larger, overall EXPRESS model for the entire UOD. Perhaps the biggest difference between EXPRESS and other information modeling languages, in general, or constraint languages, in specific, is the relatively large number and varied nature (i.e. ranging from purely abstract constructs to database definition-related constructs) of its standard constructs. This provides a certain degree of rigor and ensures that the declarative statements are machine-interpretable. Since these constructs are largely at the fairly primitive or meta level, they do not overly restrict the ability of modelers to define more unique entities and relationships at the application domain or "mid-world" level. The constructs include some concepts typically not found in information models, but rather more closely associated with the object-oriented paradigm (which is described under a separate subheading within this section). These concepts include: [super]types and subtypes; derived attributes; aggregations; bags; sets;



lists; rules; relational operators; functions (including many standard arithmetic functions); procedures; and data types (e.g. date, string, real, etc.). Some statements may be executable. That is to say, statements can be written at a low enough level such that they can be executed. Because of the EXPRESS syntax and the nature of its basic constructs, declarative statements in EXPRESS look fairly similar to Pascal code. Executable statements, of course, look even more like program code. Some attribute definition statements using data types (such as integer, string, etc.) and cardinality rules look like database definitions.

There is also a graphical form of EXPRESS, known as EXPRESS-G ("G" for graphical). It was specifically developed for the graphical rendition of information models already defined in the EXPRESS [textual] language form, but it may be used a modeling language in its own right. However, it should be noted that EXPRESS-G only supports a subset of the EXPRESS language. In other words, there are many semantics in an EXPRESS model that cannot be depicted in the EXPRESS-G graphical form, and using the graphical form as a data acquisition or modeling tool there are many semantics of EXPRESS that cannot be captured.

In addition to the general requirements noted above, there are several requirements which EXPRESS-G in particular was intended to meet. For example: a diagram must be able to span more than one sheet of paper; the graphical form should support levels of abstraction (for example, entities may be modeled without their associated attributes being modeled); the actual pictures should require only a minimal level of computer graphics capability and should be printable using non-graphical symbols on a line printer; and a processor facility should be able to be developed for converting from EXPRESS textual form to EXPRESS-G graphical form.

EXPRESS-G supports the notions of entity, attribute, types, relationships and cardinality as found in the EXPRESS language. It also separately supports the notion of a schema, as distinct from an instance-level model. Currently it does not provide any support for the constraints and constraint-related mechanisms provided by the EXPRESS language (e.g. rules, functions, procedures, etc.). In that sense, EXPRESS-G can be thought of as addressing the data modeling aspects of EXPRESS, without the constraint language portion of EXPRESS. There are three major types of symbols in the graphical form of EXPRESS:

- Definition: Symbols for denoting things (i.e. entities and attributes) which form the basis of the information model. These symbols are used to represent the semantics or content of the model.
- Relation: Symbols which describe the relationships that exist among the definitions. Like the definition symbols which they relate, these symbols are used to represent the semantics or content of the model.
- Composition: Symbols (such as off-page connectors) which allow the diagram to be displayed on more than one sheet of paper. These symbols are not really used to represent model content or semantics, but rather to support the presentational semantics of the model.

EXPRESS-G graphical syntax has some similarities to Entity-Attribute-Relationship (E-A-R) modeling forms, but in other respects is closer to Binary and Elementary N-ary Relationship (B[EN]R) modeling forms (with respect, at least, to logical data models or logical database definitions). E-A-R modeling, B[EN]R modeling and data modeling are all described under separate subheadings within this section). To be specific, a box is used to represent an entity,

and a line between two boxes (or in the case of recursion to the same box) represents a relation. The entity name or label appears inside the box. A tree relation (for example, a "person" may be either a "male" or a "female", or "outer shape" may be either "topology" or "geometry") is represented by a darkened relation line. A normal, required relation is represented by a normal tone, solid line. An optional relation is represented by a normal tone, dashed line. The "to" direction of the relation is denoted by putting a circle at the end of the line. The line is labeled with the relation name or role. One major difference with E-A-R models is that in EXPRESS-G attributes are modeled as relations, not as name labels or "fields" within some portion of the entity box. The name of the relation (i.e. the label on the line) is the name of the attribute. A normal entity appears on one end of the relation, and a data type appears on the other end. For example, the entity "person" might have a relation line with the name "birth-date". That represents the attribute "birth-date" on the entity "person". At the other end is the data type "date", which represents that "birth-date" is a "date". This form is closer to B[EN]R modeling, where attributes are treated as entities themselves with specific "attribution relations" drawn (and individually labeled) between them and their associated entity. "Date" is then related to another entity (in this case a primitive construct) "integer", representing that "date" is an "integer". Cardinality (in the broad sense of domain and range rules) is defined on this last relation. This allows for low-level specification of attributes in a form more closely associated with data models or database definitions.

### Extended Entity Relationship (EER) Modeling

*[Note: includes DATAID, M\*, Ross' EER and Bachman Object-Role Model]*

Extended Entity-Relationship (EER) modeling approaches were developed by Ceri [1983], Albano [Albano et al., 1985], DiLeva [DiLeva and Giolito, 1986], Ross [1987] and others. While the nature of the extensions varies considerably, many of them relate to further organization or classification of the entities and/or relations in the basic E-A-R model. For example, an EER approach is used in the Italian DATAID logical database design methodology and the Canadian M\* methodology for analysis and modeling of production-rule systems. It differs from basic E-A-R approaches in that it includes a new component, the Abstraction Hierarchy, which allows for the identification of such things as subset relations (referred to as the subset hierarchy) and categorization relations (referred to as the partition hierarchy). The abstraction hierarchy can be thought of as a way of introducing some aspects of the object-oriented paradigm into the E-A-R model. It should be noted that the syntax of IDEF1X allows for limited modeling of categories, too. It also has a number of other syntactical and methodological conventions which make it particularly well-suited to logical data and database modeling. For this reason, some people refer to IDEF1X as an Extended Entity-Relationship model.

Ross' EER supports a notion very similar to the Abstraction Hierarchy through a syntactical construct called an Entity-Type-Hierarchy (ETH) or Generalization Hierarchy. Ross uses ETH's specifically to categorize entities into [super]type/subtype or class/subclass relations. ETH was introduced into Ross's EER largely to provide the basis for [future] inferencing. Since ETH and the Abstraction Hierarchy of DATAID and M\* are strict hierarchies, multiple inheritance capabilities would be required. Ross also introduces other constructs often associated with the object-oriented paradigm. Examples include Subsets (similar to the subset hierarchy in DATAID and M\*) and Aggregate Objects (known as "bags" in object-oriented terminology). An Aggregate Object is simply a collection or group of other entities. It is represented as a [collective] entity to which the members of the group relate. The collection or grouping is usually

based on some, logical, shared context — although in theory any entities could be grouped together for some purpose. GUIDE differentiates two kinds of aggregation: one relating to composition or part-whole (the archetypical example being a Bill of Materials); the other more casually related to grouping (GUIDE calls this “predicated”). Ross’ EER also allows relations to be treated as entities. The relations become Association Types, which can have attributes or stand in other relations (in some cases, even with other Association Types). Whereas many extensions to E-A-R models deal only with extensions to the static nature of entities, Ross’ EER extends into constraint checking and even into some aspects of behavior modeling. The behavior modeling introduces constructs and modeling syntax for modeling states, events and propagation rules. Behavior modeling is described under a separate subheading for process modeling within this section.

Like DATAID, M\* and Ross’ EER, the Business Entity Relationship Model (BERM) [Jorgenson, 1991a and 1991b; Jorgenson and Walter, 1991] developed at Boeing attempts to incorporate object-oriented approaches into E-A-R modeling. BERM is described under its own subheading within this section.

Another EER model is Bachman’s Object-Role Model. Some people classify this as a separate kind of E-A-R — something besides an extended E-A-R. However, the same argument could probably be made about any EER where the extension is in a fairly segregated area. In this case, Bachman’s Object-Role Model simply allows entities to “role play”. For example, if we have two entities Employee and Consultant, we can create a role entity called Project Manager. Either or both Employee and Consultant can play the role of Project Manager, but certainly not every Employee or every Consultant plays that role. The underlying premise is that many entities found in typical E-A-R models, particularly those that do not really seem to fit the normal modeling syntax, are not really entities at all. This methodology proposes identifying the real entities (referred to as “objects”) and then expressing many of the would-be objects as roles and associated relations.

There are a number of other interesting extensions to E-A-R approaches, some of which are currently implemented in one or more E-A-R or EER models and others that are being proposed by groups such as GUIDE. Some of these extensions relate to modeling more complex relationships, for example, imposed existence constraints. This is an extension to, or further qualification of, the optional relationships already represented in some E-A-R models. In this case though, the existence rule is specifically defined. That may require a whole set of other complex relations to be described in the model. A simple example is the case where two types of relations can possibly exist between two entities, but only one or the other may exist at any given time (determining which one depends on checking certain conditions, which may only occur at the entity or attribute instance level). Other E-A-R models are being treated as Conceptual Models (based on ANSI/SPARC and ISO three-schema database concepts) and are being mapped to various data models which contain data about the implementation and/or presentation of data represented in the Conceptual Model. Functions, processes or operations can currently be modeled in any basic E-A-R model — but only as static entities. Extensions in this area deal with linking these static representations of processes, etc. (through various formal relations) to the static representations of the data or information that the processes take as input, manipulate and/or create as output. As noted above, Ross’ EER already includes some extensions in this area. Finally, per GUIDE, Bachman has proposed extensions to his Object-Role Model to deal with something called “concepts” or “data concepts”. The example cited is that Telephone Number is a “data concept” that identifies phone lines and/or people in relation to the phone system they are connected to or use. Business Telephone Number and Home Telephone Number are “roles” of that

data concept. Bachman refers to this as "dimension". This seems like a somewhat contrived, overly bundled and unnecessarily complex way of representing what could easily and more clearly be represented using several separate, but closely associated relations. For example: a Person is related to one or more Telephone(s) or Telephone Line(s) in some manner (call the relation whatever you like); Telephone Numbers are a typical way of accessing a given Telephone Line or Person; Telephone Numbers may in turn relate to Home or Office or Car or lots of other Places. It is hard to see what is terribly unique about any of that or why new primitive constructs such as "data concepts" and "dimensions" are needed to represent it.

### Hypertext

By dividing text (such as that produced by any word processor) into individual text nodes and using the "node-link-node" model to link nodes, Hypertext systems allow a user to identify and navigate to related text nodes by following the links [GUIDE, 1991]. Hypertext provides somewhat more semantics than other text processing, in that keywords or concepts may be identified as text nodes and linked or related to other words or text descriptions to provide some expression of meaning. The means of providing this capability may, in fact, utilize an underlying database or knowledgebase to store the text node and linkages.

Hypertext may be useful in that it provides a means for non-modelers to express some semantics simply by performing "modeling" using narrative text descriptions. Rather than to allow completely free-form text, however, many hypertext systems use some form of guided text selection with associated node assignment.

### Integrated Computer-Aided Manufacturing (ICAM) Definition (IDEF) Modeling Methodology

The Integrated Computer-Aided Manufacturing (ICAM) Definition (IDEF) modeling methodology, and various automated tools developed based on the methodology, have been utilized by numerous DoD, defense contractor and commercial enterprises since they were first developed in the late 1970's. IDEF is the most widely used modeling methodology in the aerospace and defense industry, and is among the most used in the commercial industrial sector. The ITKM industry survey results, presented in Section 2. of this document provide supporting data about the widespread use of IDEF. IDEF was developed under Air Force contract because the Air Force recognized a need for a standard approach to describing manufacturing enterprises, both in terms of their current ("As Is") state and their envisioned state after implementation of new technologies ("To Be"). The majority of the new technologies being considered for implementation at that time were "hard automation technologies" oriented to productivity improvements on the shop floor. The IDEF methodology has, however, been used subsequent to that period for modeling in support of technology projects directed at non-touch, above-the-shop-floor labor.

According to Sarris [1988] three distinct but related IDEF methods were originally conceived: IDEF0, for functional modeling of enterprise activities; IDEF1, for modeling of the data and information used and/or produced by the various activities; and IDEF2, intended to represent the dynamics of the enterprise. The IDEF0 method was developed under the ICAM Program by SofTech, Inc., as a refined and expanded version of SofTech's Structured Analysis and Design Technique (SADT) methodology. The IDEF1 method, in its original form, was developed by Hughes Aircraft and D. Appleton Company (DACOM), also under the ICAM Program. IDEF1 was later improved by DACOM under a follow-on ICAM contract and became

## ITKM State-of-the-Art Review

IDEF1X. IDEF2 was only partially defined under the ICAM Program, but has since been used as the basis for several proprietary methodologies and tools which were developed under private funding. ITKM industry survey respondents noted SLAM, TESS and SAINT as examples of such technologies.

As a method for modeling functions (or activities, or at a superficial level, processes), IDEF0 has several strengths and several weaknesses. However, many of the weaknesses first occur, or only become significant, when the method is applied to complex white-collar (i.e. analytical or logical) functions rather than physical functions. Because IDEF0 models are among the most rigorous and exhaustive of current functional modeling approaches, they can provide useful data in a form that is acceptable as a starting point for more advanced modeling of functional interrelationships and the information involved in those interrelationships. The biggest concerns regarding current IDEF0 modeling efforts are to ensure that the models contain sufficient detail and that proper leveling be achieved in the models, as this is critical to the future usefulness of any IDEF0 model. Although functions (or activities or processes) are identified in IDEF0 models along with their inputs, controls or constraints, outputs and mechanisms (known collectively as ICOM's), full understanding of the functions is dependent upon more detailed modeling of the ICOM's using IDEF1/1X. Unfortunately, there are several difficulties in linking IDEF0 and IDEF1/1X models. Although one can ascertain a limited understanding of functional interrelationships by studying IDEF0 models alone (specifically, by traversing functional linkages up, down and across the hierarchical structures and/or tracing data using a cross-reference of informational elements to the functions that use them), a more direct representation of such linkages is critical for the development of enterprise models to support enterprise integration and white-collar, management automation. Additionally, IDEF0 models identify functions from a static perspective only. Much more extensive process modeling is required to capture complete process dynamics, to support simulation or other behavioral modeling and to allow direct execution of operations based on their process models.

IDEF1/1X is a good example of an Entity-Attribute-Relationship (E-A-R) modeling method. The original version (IDEF1) drew heavily on the works of Codd in relational theory and Chen in basic entity-relationship modeling, as well as on the internal expertise of the companies that developed it (Hughes Aircraft and D. Appleton Company [DACOM]). DACOM then applied lessons-learned from numerous industry applications of IDEF1 and from the Air Force ManTech Integrated Information Support System (IISS) Program to produce a revised and extended version known as IDEF1X (IDEF1 eXtended). The primary differences between IDEF1 and IDEF1X are improved (i.e. cleaner, clearer) graphical representation and modeling procedures, the capability to have role names for the information entities/attributes, and the introduction of categorization relationships (also called generalizations). As with all *information* (versus *data*) models, the information represented in IDEF1/1X is supposed to be independent of any physical data processing media that might be used to store and retrieve it. The models are intended to represent the neutral or Conceptual Schema component of the ANSI/SPARC and ISO three-schema database architecture [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a]. In practice, though, this is frequently not the case. Oftentimes IDEF1/1X models simply look like models of relational databases. This can be attributed partly to the fact that IDEF1/1X models (and E-A-R models in general) lend themselves to modeling in relational form, and also to the fact that they frequently get used to design relational databases (i.e. they are used to develop logical data models and physical database models for relational databases). The problem is that a distinction is not always made between IDEF1/1X models that are intended as logical information models versus those that are logical data models or physical

data models for relational databases. As with all E-A-R models, IDEF1/1X represents entities, relationships, attributes, and values (when populated with specific data).

Like IDEF0, IDEF1/1X reaches its limits when modeling environments with a high degree of domain interaction (i.e. the target areas for enterprise integration). Most of the above-the-shop-floor, white-collar domains fall into this category. One of the biggest weaknesses lies not so much with IDEF1/1X itself, but with the nature of its linkage (or lack thereof) back to IDEF0. A good information modeling method must deal both with intricate relationships among various pieces of information and between information and the processes (or activities or functions) that create, utilize, modify and destroy it. The concept of closely linking processes to information (i.e. entities and their attributes) is fundamental to capturing and representing the dynamics of an enterprise. This is particularly true of the integrated information required to support decision-making and other analytical processes performed at the enterprise business level. This focus on enterprise processes and the information required to support them is part of a new *declarative paradigm* which places information in the hands of end-users. This is in contrast to the *procedural paradigm*, which has traditionally been concerned with capturing data from end-users for purposes of volume-intensive data processing. In the procedural paradigm, the control of information is really in the hands of information resources management or data processing personnel, not end-users. One specific limitation of IDEF1/1X in this area is the difficulty of using it in close cooperation with IDEF0 and other process modeling methods to model domain interaction and the associated integrated information, particularly at a level sufficient for representing domain views, cause and effect relationships, event propagation rules, etc. The semantics of information are critical to this kind of modeling. This kind of modeling also requires explicitly representing the "process of interaction" itself as an "object". In other words, there is a need to represent processes in objectified form. These objective representations of the processes of interaction are necessary to describe both the relationships among components of information and between information and enterprise processes. Other limitations of both IDEF0 and IDEF1/1X are that they do not explicitly represent complex user, agent, or domain views, modes such as "time" (i.e. temporal data), belief states, probability, etc., or context (as provided by a collection of modal data which qualifies events or states represented in a model). These are all necessary for determining under what situations or scenarios certain information is used, in what form, by what users, for which functions, and for what purpose.

### IDEF0 Functional Modeling

Sarris [1988] states that IDEF0, as a method for modeling functions (activities), has both several strengths and several weaknesses. However, many of the weaknesses first occur, or only become significant, when the method is applied to complex white-collar functions such as analysis or decision-making, rather than to physical functions such as those performed closer to the shop floor.

IDEF0 is a fairly recent functional modeling method (developed in the late 1970's and early 1980's) and was therefore able to draw upon lessons-learned gained from the use of earlier functional modeling methodologies. Many of its philosophies and conventions are based on the concepts of top-down, structured analysis and design and segmentation of problem spaces into reasonably-sized modules — concepts popularized by data processors during the 1970's. In this regard IDEF0 is similar to IBM's HIPO (Hierarchical Input-Process-Output), the Yourdon methodology [Yourdon Inc., 1980], and other modeling methodologies of this genre. HIPO (or variations of it) is still used by several companies in industry, but is no longer a part of IBM's

official systems development methodology (now called Applications Development Cycle or A/D Cycle). No official documentation of HIPO could be found. The Yourdon methodology is still fairly widely-used and supported. It is described under a separate subheading within this section. IDEF0 is somewhat more formal in syntax than Yourdon. This results in a uniformity that acts as an advantage particularly in large-scale modeling efforts. However, some flexibility is lost through this formality. In some cases the environment being modeled must be modified to "fit" into the model, rather than vice-versa. Unfortunately, this means that the model may then not accurately reflect the real semantics of the environment being modeled. In other cases the IDEF0 rules are bent, opening up the possibility of inconsistent, ambiguous or incompatible models. The ITKM State-Of-The-Art survey indicated that industry modelers strongly prefer flexibility and are inclined to bend the rules if necessary to meet their modeling requirements. Consultants and vendors of modeling tools understandably prefer more rigid application of modeling rules and standards. This may explain the initial reluctance of some companies to try IDEF0. Its use requires a serious commitment to modeling. The IDEF Users' Group is helping to gain acceptance of IDEF0 standards, adoption of new syntax where reasonable and necessary, and sharing among its members of approaches to developing complex models which otherwise might lead modelers to modify the syntax or create unique syntax rules as "quick fixes" to difficult problems.

A case in point of IDEF0 flexibility limitations is the "3 to 6 rule" which stipulates that a parent function may have no fewer than three and no more than six children. The reasoning is that fewer than three children indicates that the parent may not really need to be decomposed: its one or two subfunctions can be dealt with at the same level as the parent (e.g. if there is only one subfunction, it might be added as a new function at the same level as the parent; if there are two children, the parent might be replaced with the children — each treated now as a new parent at the same level as their former parent). More than six children suggests a missing parent at the current level or a missing interim level (e.g. a new parent might be added at the current level, under which some of the children could now be placed; alternatively, some of the children could be bundled together at the current level — those children would then get treated as parents at the next level down and could then be decomposed into three to six children). The "3 to 6 rule" is also said to have been based loosely on the idea of "chunks" of information which the human mind can deal with most effectively [Miller, 1956]. Miller suggested the optimal number was seven, plus or minus two. For something as complex as a multi-leveled functional model, a smaller number was considered more appropriate. This is a case, however, where the "seven, plus or minus two" memory guidelines suggested by Miller may have been overzealously applied. The Yourdon methodology, for examples, cites Miller's guidelines as being useful during functional decomposition, but is not rigid about application of the guideline as a formal rule. IDEF0, on the other hand, is quite rigid about application of the "3 to 6 rule". In some cases, application of this rule results in mixing, within a given level, of functions that do not belong together at the same level. In other cases, somewhat arbitrary "virtual parents" or extra children are created merely to satisfy the syntax rules. This may, in effect, distort the accuracy and internal consistency of the model.

Despite its "formality", IDEF0 is in many respects simpler than Yourdon and other similar methodologies. This may arise from the fact that the Yourdon methodology and others like it were developed specifically for use in [functional] analysis done as part of an information systems analysis effort. The influences of data processing concerns and requirements are evident. IDEF0 is more neutral in the sense that it was developed for enterprise functional modeling done as part of an overall factory analysis effort. There are few, if any, purely data processing-related

conventions. However, to conduct a full information systems analysis using the IDEF methodology, one would have to perform IDEF0, IDEF1/1X and IDEF2 (or some comparable system flow/system dynamics) modeling. As described under the subheading of IDEF1/1X within this section, these disparate types of IDEF models do not always link together as smoothly or accurately as methodologies that incorporate functional, informational and dynamics modeling elements into a more unified methodology (such as Yourdon). Yourdon, for example even extends fairly easily into structured programming specifications and population of a data directory or data dictionary.

IDEF0 drew heavily on SofTech, Inc.'s SADT (Structured Analysis and Design Technique) methodology and is, in fact, derivative of SADT [Freeman, 1987]. SADT is not described separately in this document, since the most widely-used aspects of SADT are so similar to IDEF0. The explicit concept of maintaining a consistent purpose and viewpoint in the model is directly inherited from SADT. SADT is a formal, complete information systems analysis methodology. It differs from many other systems analysis methodologies in one major way, namely the explicit segregation of data from function for at least some aspects of systems analysis and design. The SADT methodology on the one hand emphasizes *data flow* diagrams (in its "data-gram" form, which has no direct correlate in the IDEF methodology), and on the other hand *functional flow* diagrams (in its "acti-gram" form, from which IDEF0 is most closely derived) [Leite and Freeman, 1991]. The functional flow diagrams lack many of the more complex elements typically found in control flow diagrams, state transition diagrams and other process dynamics-oriented modeling forms. In the case of IDEF0, this translates into the need for IDEF2 or other models that provide more complete system dynamics, such as a way of representing: temporal order; causal order; antecedents and consequents; conditional checks and branching; and other process relationships that oftentimes get operationalized in program logic. SADT was also responsible for contributing to one of IDEF0's richest features — the capture of data relating to functions in terms of four explicit categories: inputs, controls, outputs and mechanisms (often referred to in IDEF0 as ICOM's).

Where IDEF0 reaches its real limits (i.e. where its weaknesses become significant deterrents to clear, precise modeling) is in the area of modeling analytical or decision-making functions at the enterprise business-level. In these cases, the nature of inputs and outputs, as well as the nature of the functions themselves, are much less well-defined than at the level of clerical or data processing functions, or physical functions closer to the shop floor. Many of the functions require deep, commonsense knowledge, as well as broad and deep knowledge of the particular domain. None of that is easily represented in IDEF0 models. Other limitations of IDEF0 models are that they do not explicitly represent complex user, agent, or domain views, modes such as "time" (i.e. temporal data), belief states, probability, etc., or context (as provided by a collection of modal data which qualifies events or states represented in a model). These are all necessary for determining under what situations or scenarios certain information is used, in what form, by what users, for which functions, and for what purpose.

Still, IDEF0 models provide useful data in a form that is acceptable as a starting point for more advanced modeling of functional interrelationships and the information involved in those interrelationships. The biggest concerns regarding current IDEF0 modeling efforts are to ensure that the models contain sufficient detail and that proper leveling be achieved in the models, as this is critical to their usefulness as the basis for more advanced models. Although functions (processes) are identified in IDEF0 models, much of their description is dependent upon modeling of their informational elements (inputs, outputs, etc.) using IDEF1/1X (the informational modeling component of IDEF). As discussed under a separate subheading for IDEF1/1X within this section,



there are several difficulties in linking IDEF0 and IDEF1/1X. Although one can ascertain a limited understanding of functional interrelationships by studying IDEF0 models alone (specifically, by traversing functional linkages up, down and across the hierarchical structures and/or tracing data using a cross-reference of ICOM's to the functions that use them), a more direct representation of such linkages will be critical for enterprise modeling in support of enterprise analysis and enterprise integration.

### IDEF1/1X Information Modeling

IDEF1/1X is a good example of an Entity-Attribute-Relationship (E-A-R) modeling method. The original version (IDEF1) drew heavily on the works of Codd [1979] in relational theory and Chen [1976] in entity-relationship modeling, as well as on the internal expertise of the companies that developed it (Hughes Aircraft and D. Appleton Company [DACOM]). DACOM then applied lessons-learned from numerous industry applications of IDEF1 and from the Air Force ManTech Integrated Information Support System (IISS) Program to produce a revised and extended version known as IDEF1X (IDEF1 eXtended). The primary differences between IDEF1 and IDEF1X are improved (i.e. cleaner, clearer) graphical representation and modeling procedures, the capability to have role names for the information entities/attributes, and the introduction of categorization relationships (also called generalizations).

As with all *information* (versus *data*) models, the information represented in IDEF1/1X is supposed to be independent of any physical data processing media that might be used to store and retrieve it. The models are intended to represent the neutral or Conceptual Schema component of the ANSI/SPARC and ISO three-schema database architecture [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a]. In practice, though, this is frequently not the case. Oftentimes IDEF1/1X models simply look like models of relational databases. This can be attributed partly to the fact that IDEF1/1X models (and E-A-R models in general) lend themselves to modeling in relational form, and also to the fact that they frequently get used to design relational databases (i.e. they are used to develop logical data models and physical database models for relational databases). The problem is that a distinction is not always made between IDEF1/1X models that are intended as logical information models versus those that are logical data models or physical data models for relational databases.

The IDEF1X Modeling Manual [D. Appleton Company, 1985] describes the basic constructs as follows:

- *Entities* — Things about which data is kept, e.g. people, places, ideas, events, etc. — represented by a box. Entities are named with nouns. The entity name is technically its "identifier", although many IDEF1X models that serve as logical or physical relational database models also indicate one or more key attributes (described below) as either logical or physical access paths into the entity and consider this, in practice, as an identifier of the entity.
- *Relationships* — Relations or links between those "things" denoted as entities — represented by lines connecting the boxes. All basic relationships may be referred to as "connection relationships". IDEF1X also supports "categorization relationships" which are a special kind of relationship used to represent classes and subclasses (also variously referred to as types/subtypes, genus/species and categories of things/things in the categories). Relationships are named with verbs.

- *Attributes* — Characteristics of those “things” denoted as entities — represented by attribute names within the box. Attribute names are descriptors.

Syntax rules define *entities* as either “independent” or “dependent” in regards to other entities with which they have a relationship. In some other entity modeling techniques, independent entities are referred to as “kernel entities” and “dependent” entities as “characteristic entities”. Independent entities can exist on their own in the model (or at least be identified uniquely), without any other entity in the model. Dependent entities by their nature depend on one or more other entities in the model for their existence (or at least, identity). A common example involves a Purchase Order and a Purchase Order Item. Most enterprises allow a Purchase Order to exist (at least certain kinds of Purchase Orders at certain points in the procurement process) without any Purchase Order Items. Purchase Order in this case would be modeled as an independent entity. However, the inverse is not allowed in most enterprises, i.e. a Purchase Order item cannot exist without a Purchase Order. Purchase Order Item would then be modeled as a dependent entity. In terms of “identity” versus “existence”, it could be said that a Purchase Order Item could not be *identified* without knowing the Purchase Order on which it is an item [Note: this is not to say that physical or even logical access would necessarily require first accessing the Purchase Order entity to get to the Purchase Order Item, but it does indicate that identifying the Purchase Order Item requires knowing the Purchase Order. For example, the key attribute of the Purchase Order Item would likely be a concatenated key attribute that includes the Purchase Order key attribute (Purchase Order Number, for example), as well as some unique component (the Purchase Order Item Number itself)]. Because “identity” in this sense is so closely allied with keys and other logical and physical access issues associated more with the implementation of relational databases than with information modeling, we prefer “existence” as the determinant of independence or dependence. Identification is best handled through another convention of the syntax, referred to as “identifying versus non-identifying connection relationships” (which are described below). In any case, IDEF1X supports the modeling of independent versus dependent entities quite well (using squared versus rounded boxes). A proposed extension to IDEF1X [D. Appleton Company (DACOM), 1988] allowing optional/non-optional parents to be specified for non-identifying relationships is also related to this issue, as is the No-Null Rule mentioned below. Some people consider these features of extended E-A-R models, but we consider them differentiating features among E-A-R models.

*Relationships* are defined as “identifying connection relationships” if the child entity is identified by its association with its parent entity. If every instance of the child entity can be uniquely identified without knowing the associated instance of its parent entity, then the relationship is described as a “non-identifying connection relationship”. In standard IDEF1X, any child entity in an identifying relationship is also a dependent entity, since it “depends” on its parent for its “identity”. As noted above, the semantics of existence might better be determined on their own merits, with identifying relations being a separate, secondary consideration. Relationships have “cardinality” — the specification of how many child entity instances may exist for a given parent entity instance.

As noted earlier, IDEF1X allows the modeling of entity classes (entities that represent categories of things) through the use of categorization relationships. The categorization relation is also sometimes referred to as class/subclass, type/subtype, genus/species, categories of things/things in the categories, and kind/kind of. The cardinality for a category entity is always zero or one. Category entities are always dependent. A category entity can have only one general

entity. That is, a given category entity can only be a member of one category. This is, of course, quite limiting semantically and sometimes leads to the introduction of contrived category entities to get around the limitation. The general entity is always independent unless its identifier is inherited through some other relationship (i.e. unless it is itself a child in some identifying relationship).

The categorization relationship may provide the basis for very limited inferencing (the limitation arising from the fact that a given category entity can appear in only one category). Unfortunately, categorization relationships in IDEF1X are often misused. Many of the relations modeled as category relationships could be explicitly modeled as connecting relationships through the introduction of additional entities and their relationships. For example, some IDEF1X models might show a category entity called ENGINEERING ANALYSIS with entities in the category such as STRESS ANALYSIS, THERMODYNAMIC ANALYSIS, LOADS ANALYSIS, etc. Alternatively, this could be modeled by introducing entities for THERMODYNAMICS, LOADS, and STRESS and relating those to ENGINEERING ANALYSIS through specific relations. Categorization relations certainly have a place in models with limited focus and in working models that will be subject to further analysis, decomposition and expansion. However, modelers should be careful not to hide important semantics through overuse of the categorization relation.

Another refinement in the IDEF1X syntax and procedures over IDEF1 is the explicit representation of many-to-many relationships between entities through the use of "non-specific relationship semantics". In IDEF1X many-to-many relationships are procedurally allowed *during the initial stages of development*, but before the model is finalized all many-to-many relationships must still be normalized out of the model. This is accomplished by inserting fictitious, intermediate "intersection"-type entities which serve to decouple the direct many-to-many relationships. These "fictitious" entities serve to represent the relation between two or more other entities. They enable the representation of other information or semantics about the relation itself (including such things as combinations of foreign keys, or new attributes that serve to further qualify or describe the "intersection" relation itself). However, once created these must be inferred from an analysis of the model, since they look just like other "real" entities. That is to say, these are not explicitly represented as entified relations by the modeling syntax of either textual or graphical constructs in IDEF1X.

To give an example of this situation, the IDEF1X Modeling Manual [D. Appleton Company (DACOM), 1985] uses an example of two entities: EMPLOYEE and PROJECT. The relationship rule in the particular example enterprise being modeled is that a given employee (an instance of the entity EMPLOYEE) may work on many projects at one time (relates to many instances of the entity PROJECT), and a given project (an instance of the entity PROJECT) has several employees working on it at one time (relates to many instances of the entity EMPLOYEE). The normalization method recommended in the manual is to add a new "intersection" child entity (called PROJECT-ASSIGNMENT) indirectly linking the two parent entities (i.e. by creating a child common to both EMPLOYEE and PROJECT).

It is the opinion of some that the use of the normalization convention in IDEF1/1X came about largely because of the limitation of many existing database management systems (DBMS's) to handle the physical implementation of many-to-many relationships. However, as noted earlier, the International Organization for Standardization (ISO) [1987a] explicitly stipulates that an Entity-Attribute-Relationship information model is supposed to represent the neutral view (Conceptual Schema) of enterprise information *independent of any physical data processing constraints* (such as the limitations of DBMS's). Others are of the opinion that the intent of

IDEF1/IX was merely to use a higher-level “normal form” that decomposed many-to-many relationships into more elemental components. Still others point to the need to treat some relations as entities and to add attributes to the relations, although this is a somewhat new idea that may have had its origin in the object-oriented paradigm. In any case, the argument to be made regarding IDEFIX is that adding the fictitious child entity is a convolution of the model, rather than a purification. The “intersection entity” is serving as a weak substitute for a more direct, explicit representation of the many-to-many relationship between the two “real” entities.

An alternative, cleaner method would, therefore, be to provide the capability to directly model such relationships so that they could be explicitly defined — and so that the nature of the relationships could be represented through the use of “attributes of relationships”. For example, the kind of relation that two or more objects stand in, the cardinality or quantifier on the relation (such as many-to-many), and other data pertaining to the relationship should all be definable in the model. Additionally, rules for exercising or processing certain relationships could be described using relationship attributes. IDEF1/IX addresses only the cardinality aspect. Hull and King [1987] state that several more advanced E-A-R modeling methodologies have the capability to explicitly represent other, more complex data pertaining to relationships. In their words, “In a model that stresses type constructors, relationships between types are essentially viewed as types in their own right; thus it makes perfect sense to allow these types to have attributes which further describe them.”

As an example, the Extended Entity Relationship (EER) modeling technique developed by Ross [1987] provides enhanced relationship modeling capability to a limited degree using what he calls an “Association Type”. Unlike the IDEFIX “intersection entity” convention, association types are not “fictitious” entities that get treated the same way as “real” entities (or depicted the same way graphically). Rather association types are a separate modeling convention for explicitly representing certain kinds of relationships. Relationships are described by means of certain attributes defined under the association type. Ross calls these attributes “intersection data”. However, the limitation is that association types are severely restricted in terms of the attributes they may have. There are too few attributes to fully describe the semantics of a relationship. A more general approach would simply allow any relation to be treated as an entity in its own right using the same canonical form used for all other entities, but to explicitly denote that entity as a *relation that has been entified*. In this way, the relation (as an entity) could have whatever attributes are necessary to fully describe it, but still be explicitly differentiated from “real” entities for some purposes. Ross’ EER is described in more detail under the subheading of Extended Entity Relationship models within this section.

All relationships in IDEFIX must be modeled as binary (2-dimensional) relations. Some relations by their nature relate more than two things. Relations may, in fact, be considered potentially  $n$ -dimensional or  $n$ -ary (i.e. having an arity of  $n$ ). Proponents of IDEFIX and most other E-A-R models stipulate that  $n$ -ary relations be modeled as a series of binary relations (this can be referred to as “arity normalization”). This is at best unnecessarily complex and cumbersome, and at worst may cause confusion or misrepresentation of important semantics. On the positive side, binary relationships are individually easy to grasp and lend themselves to graphical presentation using E-A-R diagrams, tree structures and other hierarchical forms.

*Attributes*, the last of the three IDEFIX components, may be regular attributes or key (identifying or linking) attributes. Key attributes, in turn, may be: unique identifiers of the entity; primary keys versus alternate or secondary keys (only primary keys can be inherited or migrated from parent entities to their children.); or foreign keys (migrated or “inherited” from a parent to a child entity through a specific connection relationship or through a categorization relationship)

versus local keys (internal to the entity). Most of this qualification of attributes is related to implementational issues, and would not be appropriate for either information modeling or purely logical data modeling.

Every attribute is owned by exactly one entity (this is referred to as the Single-Owner Rule). Wherever else the attribute appears, it is a foreign attribute inherited or migrated through some relationship involving the entity which owns the attribute. This is good to the degree that attributes common to more than one entity indicate a relationship between the entities which share that attribute. Those relationships must be explicitly described. One entity must be determined to be the "owner" of the attribute, and then relationships must be defined between that entity and each of the other entities containing the shared attribute. The attribute then migrates into the other entities as a foreign attribute.

In other cases, there may be some other more general entity which has the common attribute as one of its attributes and which is related to each of the entities that shares the attribute. For example, the attribute WEIGHT might be an attribute of many different entities. Those entities may all be subclasses of one general entity way up the class hierarchy (for example, PHYSICAL ARTIFACT). In this case, PHYSICAL ARTIFACT would have WEIGHT as a attribute and they would all inherit WEIGHT based on class inheritance. But the entities may not be related to each other in any meaningful way without working up an inheritance tree to the point of impracticality. For example, the entities DOG, AIRPLANE and CRITERION may all share the attribute WEIGHT, but they little else in common in a practical, mid-world sense. The problem is that identification and modeling of attributes like WEIGHT, SHAPE, etc. and their relationships to other attributes all the way up a class hierarchy can be a major effort and may not be merited given the purpose and objectives of the given model. Some modelers get around this rule by introducing specialized variations of the attribute just to fit each particular appearance of the attribute on some entity (the extreme case being DOG-WEIGHT, AIRPLANE-WEIGHT, etc.). This results in unnecessary specialization of attributes. In some cases it might simply make more sense to just be able to show the attribute as a local attribute of more than one entity, but ensure that the attribute is defined in the attribute glossary and applied consistently in the model. Of course, in a fully defined, fully machine-interpretable model, all the entities and relationships including things like PHYSICAL ARTIFACT and WEIGHT would have to be explicitly represented. This drives the requirement for a commonsense knowledgebase to be used when performing IDEF1X and other forms of modeling.

IDEF1X also does not permit attributes to have null values (this is referred to as the No-Null Rule). Specialized entities are created around each of the optional attributes. Like intersection entities, these fictitious entities hide important semantics that should be made explicit in the model. This restriction has no real place in logical information modeling since, of course, there are possible attributes, or attributes of parents that only get instantiated in their children under certain conditions. Explicitly representing this level of semantics is simply beyond the capability of IDEF1X. This is again, related to the use of IDEF1X models for database (particularly, relational database) design, and has little or nothing to do with pure information modeling.

The IDEF1X method encompasses a fairly rigorous set of modeling procedures that moves progressively through at least the following major steps: informally identifying "candidate entities" (which may turn out to be true entities or which may, in fact, really only be attributes); determining which candidate entities are really entities and which are attributes (this may involve directly identifying certain "candidate entities" as entities and other as attributes, as well as discovering other "hidden" entities by studying related attributes); clustering the attributes around

the identified entities (creating informal entity/attribute pools); formalizing the structure of entities and their basic attributes; and finally identifying relationships among entities (starting with non-normalized relationships and moving through normalization, and including identification of inherited attributes and keys). This is a thorough, well thought-out process that helps ensure a consistent model built on a sound logical basis. The weakest aspects of this method tends to be the modeling of relations. Oftentimes careful attention is paid to identifying entities and their attributes, but considerably less time and effort are spent identifying and analyzing the relations between and among entities.

Like IDEF0, IDEF1/1X reaches its limits when modeling environments with a high degree of domain interaction (i.e. the target areas for enterprise integration). Most of the above-the-shop-floor, white-collar domains fall into this category. One of the biggest weaknesses lies not so much with IDEF1/1X itself, but with the nature of its linkage (or lack thereof) back to IDEF0. A good information modeling method must deal both with intricate relationships among various pieces of information and between information and the processes (or activities or functions) that create, utilize, modify and destroy it. The concept of closely linking processes to information (i.e. entities and their attributes) is fundamental to capturing and representing the dynamics of an enterprise. This is particularly true of the integrated information required to support decision-making and other analytical processes performed at the enterprise business level. This focus on enterprise processes and the information required to support them is part of a new *declarative paradigm* which places information in the hands of end-users. This is in contrast to the *procedural paradigm*, which has traditionally been concerned with capturing data from end-users for purposes of volume-intensive data processing. One specific limitation of IDEF1/1X in this area is the difficulty of using it in close cooperation with IDEF0 and other process modeling methods to model domain interaction and the associated integrated information, particularly at a level sufficient for representing domain views, cause and effect relationships, event propagation rules, etc. The semantics of information are critical to this kind of modeling. This kind of modeling also requires explicitly representing the "process of interaction" itself as an "object". In other words, there is a need to represent processes in objectified form. These objective representations of the processes of interaction are necessary to describe both the relationships among components of information and between information and enterprise processes.

Other limitations of IDEF1/1X models are that they do not explicitly represent: complex user, agent, or domain views; modes such as "time" (i.e. temporal data), belief states, probability, etc.; or context (as provided by a collection of modal data which qualifies events or states represented in a model). These are all necessary for determining under what situations or scenarios certain information is used, in what form, by what users, for which functions, and for what purpose.

D. Appleton Company (DACOM) [1988] has proposed a number of enhancements to IDEF1X. Most are of a relatively minor nature and have either been adopted under the guidance of the Air Force ManTech Directorate or, subsequently, by the IDEF Users' Group (which was established with the help of Air Force ManTech for purposes of standardizing and maintaining the IDEF methodologies, primarily IDEF0 and IDEF1X). The other enhancements of a more dramatic nature were probably intended more as long-term recommendations, and have served as input to the IDEF UG Research and Development Forum. One of the most notable of these recommendations suggests adding object-orientation to IDEF1X or developing an object-oriented IDEF method. Discussions under the subheadings of IDEF4 and the Object-Oriented Paradigm, both within this section, may provide background material relevant to this topic.

## IDEF2 Dynamics Modeling

IDEF2 was motivated by the need to supplement the IDEF0 and IDEF1/1X modeling methods (then under specification) by modeling at least some aspects of system dynamics (at a minimum, process and/or information flow). IDEF2 then became the final method within the original IDEF methodology, and was directed at the modeling and simulation of system or process dynamics.

According to Mayer and Painter [1991], IDEF2 models are really comprised of four individual, but related submodels:

- Facility Submodel* — Used to specify the model of the agents (e.g. people, machines, etc.) of the system and environment. This submodel looks much like typical models used for shop floor layout.
- Entity Flow Submodel* — Used to specify the model of the transformations that an entity undergoes. This can be thought of to some degree as overlaying an IDEF0 functional or activity model onto the Facility Submodel described above. Emphasis is on the particular "paths" that an entity takes as it is processed or transformed in some way. This is by no means a full state transition model and has no state table logic. Process triggers, conditionals and branching logic are also not included here; to the degree that they are represented in IDEF2 models, they appear within the Resource Disposition and System Control Submodels.
- Resource Disposition Submodel* — Used to specify the logic of agent assignment to entity transformation demands. Some state change triggering logic and conditional branching may be represented in this model.
- System Control Submodel* — Used to specify the effect of transformation-independent or system external events on the modeled system. Some state change triggering logic and conditional branching may be represented in this model.

IDEF2 never became as formalized or standardized as IDEF0 and IDEF1/1X. For the most part, it was not directly applied in the same sense as IDEF0 and IDEF1/1X, but rather served as the driving force behind several commercial simulation packages. This is largely because simulation models, to be beneficial, need to be active or executable. They inherently lend themselves to automation using computer technology. Several software vendors have developed and marketed vendor-proprietary simulation software packages that implement some of the ideas of IDEF2. The ITKM Industry Survey respondents cited SLAM/SLAM II, TESS and SAINT as examples. IDEF2, as well as commercial simulation software based on it, have been applied largely to simulation of shop floor processes, often in conjunction with the planning, design and implementation of Computer-Integrated Manufacturing (CIM) technology and cellular, flexible manufacturing techniques.

Since no one standard version of IDEF2 exists (at least not in the same sense as IDEF0 and IDEF1/1X), the need to perform dynamic modeling and/or simulation presents industry with the following choices: purchase one of the vendor-proprietary, automated simulation packages that implement at least some of the ideas of IDEF2; develop home-grown flow methodologies and simulation modeling tools themselves; or develop limited workarounds, often based on extended

or recast versions of IDEF0 functional models. The vendor-supplied commercial software often includes sophisticated, computer-based graphics and the capability to execute simulations. These packages, while useful for shop floor process modeling, often do not have the flexibility or inherent capabilities for use in modeling and simulation of higher-level, cognitive processes. These include the analytical and decision-making process performed by white-collar workers as part of product development, delivery and support. Home-grown simulation methodologies and software are expensive in terms of the time and resources required to design, develop and maintain them. Because of this, limited workarounds using IDEF0 models tend to be fairly common in industry. Other more advanced methods and tools for dynamics modeling and simulation exist or are the subject of current research and development efforts. These are described within this section under the general subheading of Process Dynamics/Simulation Modeling, and under IDEF3 - Process Flow and Object State Modeling.

### IDEF3 Process Flow and Object State Modeling

IDEF3 is a proposed IDEF method specified by Knowledge Based Systems, Inc. under funding from the Air Force Human Resources Laboratory - Logistics and Human Factors Division. According to Mayer and Painter [1991], IDEF3 is a scenario-driven process flow modeling method created to allow experts to describe a process as an ordered sequence of events or activities. IDEF3 is based upon the concept of direct capture of descriptions of the temporal precedence and causality relations between situations and events. The major semantic entities represented in an IDEF3 model are: objects; properties; relations; and constraints. The functions or processes from an IDEF0 model can be used as a starting point for IDEF3 model building, but there is no syntax rule that the functions and functional hierarchy must be the same between IDEF0 and IDEF3 models. IDEF3 is intended to encompass and go beyond the capability of IDEF2 to model process dynamics or behavior. IDEF2, and simulation software based on IDEF2, could still be used for lower-level simulations to predict the behavior of systems modeled more explicitly in IDEF3.

The IDEF3 method consists of two modeling modes: process flow description; and object state transition description. The process flow description captures a network of relations between actions within the context of the specific scenario or situation. An IDEF3 diagram consists of the following structures: units of behavior (UOB's); junctions; links; referents; and elaborations. The UOB is the basic "object" in IDEF3 and may be a function, activity, act, process, operation, event, scenario, decision or procedure depending upon its surrounding structure. UOB's are described using verbs or verb phrases. UOB's may be decomposed into lower-level UOB's (in other words, they may form a composition hierarchy). UOB's may also be described using a set of participating objects and relations that comprise what is called an "elaboration". The problem with elaborations is that they are essentially free-form text with little or no structure. The semantics of these diagrams is not formalized and would not be explicitly accessible to the rest of the model (specifically, if the model were in automated form, the elaboration semantics would simply not be machine-interpretable). This is unfortunate, since many of the process semantics of interest would be in represented in this form. UOB's may be reused and cross-referenced. UOB's are connected to other UOB's through the use of junctions (or branches) and links. Junctions are used to express the semantics of synchronous (parallel or coincident) and asynchronous (sequential) behavior among a network of UOB's. Specifically, the semantics are: "logical and"; "logical or"; and "logical Xor". Links come in three types: simple temporal precedence; object flow; and relational links. The link described as "simple temporal precedence" is somewhat misleading as



it really only represents "sequence", which could be either true causal dependency or simple temporal ordering. "Object flow" is basically the same as "temporal precedence" except [information] objects are also passed between processes to indicate information flow. "Relational links" are a general category for representing user-defined links.

The object state transition description summarizes the allowable transitions an object may undergo throughout a particular process. The constructs of the object state description are: object states; and state transition acts. An object state is defined in terms of property values and constraints. The properties should be attributes in a related IDEF1/1X model. An object may also have pre-transition and post-transition constraints associated with it. These specify conditions which must be met before a transition can begin or be completed. Unfortunately, none of the many existing constraint languages are used to capture the constraints in a formal manner. In fact, natural English is used as the constraint language. While this may seem advantageous for capturing constraints, the problem is that natural English can be a very imprecise and ambiguous.

IDEF3 also includes an informal procedural method for capturing process flow descriptions and object state transition descriptions, as well as for describing the scenario around which a given IDEF3 model is based. As noted above, the scenario is "described" or represented using a Unit of Behavior. However, these scenario "descriptions" are not so much descriptions as they are just name labels. There is no formal structure or syntactical construct for describing a set of assumptions which pertains to the scenario. The primary way it appears that a complex scenario can be represented is as a set or a composition hierarchy of scenario UOB's, or as an elaboration. The semantics of the relations between the UOB's would have to be specified using user-defined relational links or the sequence relations (which are intended to be applied to processes not scenarios). No other method of specifying a scenario "path" is apparent. IDEF3 is envisioned as being supported by a simulator which would include: an activation sequencer (an IDEF3 syntax parser); an abductive reasoner; a tokenizer (which would generate facts based on the elaborations); a constraint propagator/truth maintainer; and a deductive reasoner. Generating facts based on unstructured elaborations would be extremely difficult if not impossible. Likewise, propagating ambiguous or imprecise constraints expressed in English may result in the compounding of inaccurate constraint semantics.

As least one project has attempted to link IDEF0, IDEF3 and Colored Petri Nets. Colored Petri Nets are described in more detail under the subheading of "Process Dynamics/Simulation Modeling" within this section. The IDEF0 diagrams appear to be developed as they normally are. They are then converted into IDEF3 diagrams (it is not clear whether this is a manual, semi-automated or automated process). The inputs and outputs from the IDEF0 model (re-labeled "primary inputs" and "primary outputs" become IDEF3 objects which take the role of states. The IDEF0 processes become the Units of Behavior which transform the states. The controls and mechanisms from the IDEF0 model (which are referred to collectively here as "resources") become IDEF3 objects and are assigned their own input and output states. The UOB's are ordered in flow sequence and decision boxes (junctions) are added. Since IDEF3 does not really have a simulation environment to support it at this time, the IDEF3 models are converted to Colored Petri Nets for conducting actual simulations. Again, it is unclear whether this conversion process is manual, semi-automatic or automatic.

## IDEF4 Object-Oriented Design

*[Note: includes the Integrated Information Systems Constraint Language (IISyCL)]*

IDEF4 is a proposed IDEF method specified by Knowledge Based Systems, Inc. under funding from the Air Force Human Resources Laboratory - Logistics and Human Factors Division and the National Aeronautics and Space Administration (NASA). Edwards and Mayer [1989] state that IDEF4 is intended as a design tool for software designers who use object-oriented languages such as C++, SmallTalk and LISP [Note: there is nothing inherently object-oriented about LISP, but the language does lend itself quite well to implementation of the object-oriented paradigm]. The object-oriented modeling paradigm is discussed under a separate subheading within this section. As a design tool, IDEF4 is intended to specify the semantics of a domain (both its data and functions) at an abstract level, rather than encoding a particular implementation of the logic for purposes of program execution (the latter is what an object-oriented programming language, coupled perhaps with an object-oriented database, is intended do).

As is the general case with the object-oriented (hereafter abbreviated o-o) paradigm, IDEF4 emphasizes modularity or grouping based on related classes of objects. The proposed IDEF4 method divides an overall model of some domain into submodels which are comprised of various kinds of diagrams and data sheets. The two major submodels deal with object classes and methods (refer to the separate subheading on the o-o modeling paradigm within this section for an overall discussion of classes, methods, protocols and other general aspects of object-orientation).

For the Class Submodel, IDEF4 uses a graphical syntax form to represent object classes and the associated inheritance of features (also referred to as characteristics or attributes) from [super]classes to subclasses. Multiple inheritance is permitted. Object classes are represented as boxes, and arrows among boxes are used to show [the relation of] inheritance. Separate diagrams are available to individually depict aspects of object classes: *type diagrams* represent the basic attribute relations for each class of objects; *protocol diagrams* represent the means by which a class of objects is communicated with through messages which invoke certain routines (also referred to as processes or operations) — the protocol is sometimes referred to as the object's interface; *inheritance diagrams* represent inheritance relationships between [super]classes and their subclasses. Individual class boxes in an inheritance diagram may have *class invariant data sheets* attached to them. These data sheets give further information about the objects in a class, perhaps relating various distinct features, in textual form. The text need not be formalized and may therefore take any form. This opens up opportunities for inconsistency in specification and later misinterpretation by programmers. For this reason, a formal language, the Integrated Information Systems Constraint Language (IISyCL) is being defined as a formal language for stating class invariants.

For the Method Submodel, IDEF4 represents the method which can be used to implement a routine or operation for a given object. Dispatch mapping is used to relate routines, classes of objects and a set of methods. The dispatch mapping forms a link then between the Class Submodel and the Method Submodel. Methods are specified using *method taxonomy diagrams*, which represent groupings of methods into method sets, and *client diagrams*, which are used to represent control relationships or "subroutine calls" between routines (technically, the calls would be between the methods that implement the routines). In the taxonomy diagrams, boxes are used to depict methods, and arrows between boxes represent the subset relation. In client diagrams, the boxes represent routines (which will be implemented by the methods), and the arrows represent control references. *Contract data sheets* are used to represent the linking of objects to specific

## ITKM State-of-the-Art Review

methods (i.e. the "contract"), and to model the procedural or functional logic of the method. It should be noted however, that IDEF4 contract data sheets do not really represent object-oriented code, in the sense of specifying how the method is to be implemented using any particular o-o programming language. Instead, contract data sheets are really abstract descriptions (in the form of declarative statements) of the basic intent of the logic which fulfills the "contract" between the object and the method. The declarative statements or assertions include both pre-conditions and post-conditions, as well as the actual function. As with class invariant data sheets, there is currently no formal language for expressing contracts. It is envisioned that the IDEF3 process modeling language (discussed under a separate subheading within this section), coupled with the IISyCL constraint language, will eventually be used for expressing contracts.

## Interpreted Predicate Logic (IPL)

The Interpreted Predicate Logic (IPL) approaches, proposed by authors such as Steel [1978], describe the environment solely in terms of entities (which may be data or processes) — for which certain propositions or assertions hold. Models consist of a set of sentences encoded in some formal language based on formal logic (predicate calculus). Because they use formal languages based on formal logic, IPL approaches are not typically regarded as user-friendly. This severely limits the ability of end-users, as well as modelers who are not familiar with formal logic, to use such methodologies. Several IPL approaches provide for dynamically extending the expressive power of the language by adding new constructs using the capabilities that are already present. That is to say, the language may be extended by defining new constructs in terms of [compositions of] existing constructs (using the Lambda operator in calculus).

GUIDE [1991] notes that an important advantage of logic-based representation schemes, such as Interpreted Predicate Logic (IPL) or other modeling languages based on predicate calculus, is that inference rules can derive additional facts from existing facts and rules. Logical representation schemes have a formal mathematical base, but lack any practical scheme for organizing a collection of facts.

Conceptual Graphs (CG) is a modeling form based on Interpreted Predicate Logic, more specifically on Typed Predicate Calculus. CG is described in more detail under a separate subheading within this section.

## NIAM

*[Note: also includes NIAM's constraint language, RIDL-C, and NORM]*

NIAM stands for Nijssen or Natural language Information Analysis Method. NIAM appears to be useful as a conceptual information modeling language for representing the "big picture" or context of information objects and their relationships (called "roles") to other objects. NIAM makes a clear distinction between purely conceptual objects (Non-Lexical Object Types or NOLOT's) and syntactical objects (Lexical Objects Types or LOT's). LOT's provide extensional definitions for NOLOT's (i.e. LOT's are the attributes and the NOLOT's are the entities). NOLOT's often have as their associated LOT's the names used to refer to an object and/or to access it in a database. An object can be a NOLOT in one diagram (i.e. one domain of discourse) and a LOT in another. For example, if you were talking about the conceptual object Part you might model the character string "Part Number" as one of its corresponding lexical objects. Then in another model where system implementation was the domain, you might model

Part Number (a data element in a database) as the conceptual object and model as a lexical object its *name* in a particular database schema, e.g. "PHDRPTNO", a field name in an IMS database.

NIAM is a Binary or Elementary N-ary Relationship Model (B[EN]R) that is, in many respects, similar to Entity-Attribute-Relationship (E-A-R) models (both the B[EN]R and E-A-R methodologies are described under separate subheadings within this section). NIAM allows certain constraints to be more formally modeled than E-A-R models, but does not have specific conventions for cardinality. This is because NIAM deals with constraints on relations at a more formal, logic-based level, rather than the less rigorous, but more flexible and richer cardinality constraints of typical E-A-R models. Some NIAM modelers have introduced conventions to handle the kind of cardinality modeling more typical to E-A-R models, but it is not indigenous to the NIAM methodology. NIAM diagrams, like E-A-R models can in theory depict *n*-ary relations, but the recommended approach, as with E-A-R models, is to decompose or normalize *n*-ary relations down into a series of binary relations. In NIAM, all relations or roles are labeled in both directions and there is no assumed parent-child or independent-dependent relationship based solely on arrow direction or position on a diagram. E-A-R models typically do make that assumption.

NIAM is a *typed method* — it is concerned only with the conceptual level. It is not really intended to represent objects and their relations the way either an end-user or database administrator would. It represents them the way a systems analyst or knowledge engineer would see them. Instances are also never addressed in a model. Type-Subtype relations, also known as Genus-Species or Class-Subclass relations, or "category" relations in IDEF1X, can be represented, but these relations are modeled separately (i.e. as distinct domains of discourse depicted in separate diagrams) from other relations in which the *objects* that are the types or subtypes stand. In other words, the type or abstraction hierarchy is treated as a totally separate relation when modeling and there is no logical connection back to the model that contains all the other relations between the conceptual and lexical objects. Due to this rigid separation, NIAM does not lend itself to inferencing. E-A-R models, again due to less formal, more flexible modeling syntax and semantics, allow the representation of instances and type-subtype relations in any model in any combination with any other relations.

There are several modeling rules associated with the application of NIAM that deal with the introduction of constraints to resolve ambiguous or redundant relations. When the constraints become complex, they can no longer be described directly in the graphical modeling form, but rather require the use of an Interpreted Predicate Logic-based constraint language. RIDL-C is a constraint language developed by Dr. Robert Meersman, one of the creators of the NIAM methodology and a professor at Tilburg University in The Netherlands. RIDL-C is much closer to the PSL-like or quasi-code statements found in EXPRESS than the rigorous formalisms of Conceptual Graphs. This opens up the opportunity for ambiguity and makes a consistent, uniform translation to or from a natural language extremely difficult. RIDL-C has not been used much because, according to Dr. Meersman, most information modeling does not require this level of constraint description. Population of a dynamic data dictionary, based on the American National Standards Institute (ANSI) X3H4 Information Resources Dictionary System (IRDS) standard, or an enterprise model integration tool such as the Integration Toolkit and Methods (ITKM), particularly the Conceptual Schema (CS) aspects of both an IRDS/data dictionary and an ITKM, will almost certainly require complex constraint definition. Given the limitations of NIAM, including RIDL-C, it does not appear to be suitable as the basis for a Normative Language for an IRDS CS or ITKM, although it may be useful if used in conjunction with Conceptual Graphs (the CG methodology is described under a separate subheading within this section) or other

## ITKM State-of-the-Art Review

Normative Languages. NIAM would certainly be one of the enterprise modeling languages represented in the Modeling Schema of an IRDS or ITKM CS.

IntelliBase, Inc. offers an automated version of NIAM graphics coupled with the RIDL-C constraint language. The tool is referred to as RIDL\* (pronounced "riddle star"). RIDL\* also includes a relational database generator (which translates NIAM with RIDL-C into relational database tables), a [limited] re-engineering module for use with legacy relational databases, and links to other front-end modeling tools such as CDC's NIAM products (IAST™ and PC-IAST™) and Oracle Corporation's CASE tool (CASE\*).

Dr. Meersman, along with Dr. Olga de Troyer (who is also at Tilburg University) is currently working on a superset of NIAM based on the object-oriented paradigm. He is calling this new approach NORM. Early references to the work described the acronym as standing for Natural Object Relation Methodology. Recent references show the acronym as standing for Normalized Object Relational Model. In any case the key point is that while the specification will be object-oriented, the assumed implementation will still be relational. The intent is to combine the best of both approaches.

## Object-Oriented Modeling Paradigm

*[Note: also includes Ontic, Ptech and DaDaMo]*

The basic notion of the object-oriented (hereafter abbreviated o-o) paradigm is that there are objects which interact with each other by sending messages. Semantics are provided by identifying the objects themselves, the other objects with which they interact, and the messages exchanged between or among the objects. Other semantics deal with how the objects respond to, process and invoke messages.

GUIDE [1991] defines an *object* as: a thing that exists and has an identity; a logical grouping of facts that can be manipulated as a whole. A *class* is a well-defined, complete description and template for a set of objects. Because people in the o-o community tend to use this definition of class, class in the o-o sense is frequently used synonymously with *set*. This causes some problems when modelers using an o-o approach start treating classes of objects as sets in the strict mathematical sense, particularly if set theoretic operations are applied which have little or no bearing on classes of objects from the viewpoint of their representational semantics, let alone from the perspective of how the objects appear and operate in the real world. Many o-o paradigms further incorporate the notion of [super]classes and subclasses (also referred to as genus-species or generalization-specialization) typically found in artificial intelligence-based representation systems. Inheritance of attributes may be allowed from a [super]class to its subclasses. Inheritance may be single (i.e. a subclass may inherit from one [super]class only) or multiple (i.e. a subclass may inherit from many [super]classes)

Rather than concentrating on representing the object itself (in the sense of what is "inside" or comprises the object), objects in the o-o paradigm get their primary definition through their official interface or *protocol*. The protocol is the way in which objects communicate to and from other objects. The protocol of an object defines its interface in terms of all the operations that may be performed on the object and how it will behave under each operation. What it does not do is specify how the operations are performed by the objects or how the particular behavior exhibited by the object is achieved. This concept is referred to as *encapsulation* or "information hiding" and involves the hiding of the implementation (i.e. the inner workings) of an object from other objects that use or send messages to that object. GUIDE states that objects contain both data

and functions, thereby “encapsulating” both these characteristics into objects. This view of encapsulation seems to relate to another concept, that of closely coupling processes to the data that is used, manipulated or produced by the processes. In implementation, this is assumed to mean a closer coupling (or, in the extreme case, no distinction at all) between databases and programs. This is a concept that may be considered a part of broader efforts in artificial intelligence, but it is not something necessarily associated with the basic idea of encapsulation. Most object-oriented literature relates encapsulation strictly to the notion of information hiding. A *message* is the means of invoking an object to perform an operation. Messages are passed between or among objects frequently through the use of a shared work space referred to as a “blackboard”. A *method* is then the internal implementation of an operation, or, expressed another way, the internal part of an object that responds to or processes a message.

Several objects may have the same or similar operations and those operations may be invoked by the same message. However, how the objects internally process and respond to that message (i.e. the method they use to perform the operation) may be quite different from object to object. GUIDE defines *polymorphism* as the capability to send the same message to objects of different classes. They also refer to this as “operator overloading”. As noted above, the same message can have a different implementation in each class that implements it. This is referred to as “late binding”. This is in contrast to “static binding”, where the target of a message is bound at link time.

Polymorphism and late binding also bring up the related notion of “generic” operations, which may be specified as being common to a number of objects, but which may, in fact, be implemented through specialized or customized methods particular to a smaller group of objects or even an individual object. In other words, messages can be sent to an object, but the method for interpreting that message may only exist in a superclass of the object somewhere up the class hierarchy. Methods become specific to a particular subclass if the method functions differently for the subclass than the superclass. There may also be generic methods or “reusable code”, which can be used by several objects which have a common operation and which benefit in some way by performing that operation using the same method. One benefit is typically ease of maintenance.

Like CASE tools (which are described under a separate subheading within this section), performing modeling and/or conceptual system design using an o-o paradigm allows the modeling and/or conceptual design to be performed at an abstract level, largely independent of implementational issues. GUIDE defines an *abstraction* as a view of something that presents the essential characteristics and omits the inessential details. Implementational issues at the enterprise modeling and/or conceptual design stages may be considered inessential details, if not outright hindrances to a good, flexible model or conceptual design. Cox [1987] describes an interesting result of using abstraction and encapsulation as part of an o-o approach to modeling or system design: they separate the user (consumer) of services from the supplier of the services. The supplier can change the implementation without impacting the consumer. Cox’s observation seems to support at least a two-schema approach to modeling and system design. In this case, the abstract model is separated from its physical implementation. A further separation of the abstract model into distinct schemas for presentation versus purely abstract or conceptual representation would result in the three-schema database architecture proposed by ANSI/SPARC and ISO [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a]. However, the o-o paradigm does not inherently involve a three-schema architecture approach. One version of SmallTalk segregated presentational issues using an approach called “Model, View, Controller (MVC)”, which included classes for presentation and user interaction. “Model” represented the data itself; “View” represented the windows, screen fields, etc. that presented the data; and

“Controller” represented the way the user could interact with the data using the View classes and manipulation devices such as keyboard strokes or a mouse click.

Methods may be *procedural* or *declarative*. Procedural methods use traditional, procedural program code. Declarative methods use a set of directly executable rules, including pre-conditions and post-condition rules which act as triggers. The claim is made that o-o models using declarative methods are dynamic, in that as soon as changes to the domain of discourse are modeled, the methods (i.e. the o-o based system which implements the model) can be easily — in some cases even automatically — changed as a result. Rule-based systems are described briefly under a separate subheading within this section.

Having said all of the above, it should be recognized that implementations of o-o systems have not necessarily followed the o-o paradigm in all aspects or with appropriate weighting among the various aspects. Many people who use the o-o paradigm as a modeling approach get wrapped up in identifying objects and messages, but do not pay as much attention to methods or, more importantly the operations or processes which methods implement. In other words, considerable modeling may be performed of objects and their interaction, with little identification of the semantics of the interaction and the operations behind the interaction. Very few o-o models deal with explicit, detailed modeling of the semantics of the operations or processes performed by the object. Many o-o models dedicate a substantial portion of the effort to defining methods, again without the abstract operation clearly defined. These methods are more often procedural than declarative. Some modelers, particularly those concerned with the rapid creation of simulation models develop methods which emulate an operation (for purposes of supporting messaging in an interactive simulation model), but still never really model the real operation itself. In other words, they embed a chunk of code or some other “message carrier” in an object and invoke it for purposes of studying object-to-object relationships, without really knowing anything about the real process or operation itself. Such simulation models may be useful under certain circumstances, but they should not be perceived as complete models of the semantics of some domain involving complex operations. In general, a broad analogy might be made to modeling chemical reactions. We can identify several substances and model what happens when certain substances are combined, but still have little or no understanding of the true nature of the substances (i.e. their chemical composition), the nature of the reactions, or the reasons why the given reactions occur. Without this deeper understanding, it would be quite difficult to predict other reactions given changes in certain conditions or new combinations of substances. Since one of the major uses of models is typically for predictive purposes, many o-o models do not serve this purpose well.

It should also be noted that acceptance of the o-o paradigm was initially hindered by the lack of technologies to implement o-o based systems in real-world production environments. Many o-o based systems were small-scale systems built using the LISP programming language and other AI tools with little or no capability for dealing with persistent objects (such as those which are the subject of most large-scale production databases in industry). Also, few if any of these systems address the issue of legacy data access. These tools may be useful for certain limited purposes. For example, Ontic91 [Givan et al, 1991] is an o-o modeling language based closely on the syntax and semantics of LISP. It is perhaps one of the best languages for writing definitions of mathematical concepts, as well as writing and verifying associated mathematical proofs (i.e. performing mathematical programming or other programming of complex math-like functions). The Ontic language is a simple generalization of strongly typed functional programming languages like Lambda-calculus. Ontic is also a functional programming language in the sense that a simple subset of the Ontic language is actually executable as a computer program. Ontic,

however, is not intended as a programming language for traditional application domains like payroll or materials planning, particularly not in large-scale production environments. However, o-o databases suitable for production environments, coupled with more user-friendly o-o programming languages, are beginning to emerge — as are limited tools for legacy database access. The Ptech™ software from Associative Design Technology is an example of these newer o-o technologies with more potential for real production use.

Ptech is described as an object-oriented CASE tool environment [Associative Design Technology (US) Ltd., 1990a] (CASE tools in general are described under a separate subheading within this section). As would be expected from that description, Ptech includes both a graphics-based conceptual modeling tool and a program code generator. The conceptual modeling tool produces something called a Content Data Model (CDM). The code generator produces code in the C++ object-oriented programming language, as well as an object-oriented database structure. The basic object in Ptech is called an Abstract Data Type (ADT). [Super]Classes and subclasses of ADT's may be defined, and Ptech supports both single and multiple inheritance from a [super]class to its subclasses [Associative Design Technology (US) Ltd., 1990b]. Operations may be defined and linked to the methods which implement them. A method in Ptech is a script which specifies how an operation is to be carried out. The script is represented in the form of an event schema. Events (i.e. changes in the state of an object) are denoted through messages which serve to trigger or invoke the methods. The message events can have conditionals (both pre-conditions and post-conditions) specified.

As an example, an Abstract Data Type called ACCOUNT might be defined with subclass CUSTOMER ACCOUNT, which in turn has subclass DELINQUENT CUSTOMER ACCOUNT. All three might have the operation CLOSE [ACCOUNT]. There might be no method specified for the operation CLOSE at the general level of ACCOUNT. However, CUSTOMER ACCOUNT might have a method specified for the operation CLOSE, and that method might differ somewhat from the method for the operation CLOSE for the object DELINQUENT CUSTOMER ACCOUNT. The message event triggering the closure of a normal customer account might be REQUEST FROM CUSTOMER TO CLOSE ACCOUNT, whereas a delinquent customer account might only be closed if a message was received from Accounting to the effect: DELINQUENT ACCOUNT WRITTEN-OFF/TERMINATE CUSTOMER.

A series of events and operations may be modeled in temporal order, including concurrency. A method in Ptech is specified in rule-based ("IF-THEN") form and the method script is used to generate C++ code. Ptech uses the Ontos® object-oriented database from Ontologic, Inc. as its underlying object representation system. When application systems are developed using Ptech, both the C++ code and the Ontos database structure are automatically generated. In one application development project (at General Dynamics), the customer estimated that 75% - 80% of the application code and database structure were automatically generated. However, the customer had to develop its own application end-user interfaces. In addition to the Ptech software itself, Associative Design Technology provides a broad, step-wise methodology for modeling a domain and developing o-o based application systems to implement the model.

DaDaMo [Radeke, 1991] is a modeling tool being developed under the DASSY project of the German Ministry for Research and Technology (with the help of a number of technical schools in Germany). The DASSY project is currently in the prototype stage. DaDaMo incorporates many of the concepts common to the object-oriented paradigm, but also has some more traditional data modeling concepts (including aspects of the basic relational model and EXPRESS). The motivation for this project comes from the domain of Computer-Aided Design



(CAD), specifically from a need to integrate very-large-scale integrated (VLSI) CAD tools with database management systems (DBMS's). The project has a short-term phase involving extensions to the Electronic Data Interchange [transfer] Format (EDIF) for describing test data and process devices. A longer-term phase address design and implementation of an interface between VLSI-CAD tools and DBMS's (both relational and object-oriented). The DASSY Data Model (DaDaMo) is intended to be a data model of an independent tool interface. It is object-oriented and encompasses mechanisms for data description and data manipulation. The Data Description Language (DDL) includes mechanisms for defining information models in the form of either the C programming language (called CDDL) [with plans for C++ in the future] or EXPRESS (called XDDL). EXPRESS is described under a separate subheading within this section. Like most object-oriented modeling languages, DaDaMo has constructs and capabilities for creating and manipulating: object classes and objects; attributes, including derived attributes; and relationships. Relationships define associations between objects and may have roles (which are specified using classes) and attributes. Hierarchies of classes and relationships may be specified. Inheritance of both attributes and relationships is allowed between [super]classes and their subclasses. Both single and multiple inheritance are allowed. DaDaMo includes a consistency checking mechanism that automatically checks constraints (i.e. rules defined on an object or class of objects) for correct (i.e. consistent) behavior. The DASSY project team believes that DaDaMo is a better choice for modeling than EXPRESS, because operations can also be defined in DaDaMo, but not in EXPRESS [Note: in the next phase of its development, EXPRESS is intended to be extended into the process or operation modeling domain]. Several classes of operations may be defined in DaDaMo: constructive operations; destructive operations; modifying operations; query operations; and navigation operations. Operations are embedded in transactions.

IDEF4 is example of a methodology for the design of systems using object-orientation. IDEF4 is described under a separate subheading within this section.

### Process Dynamics/Simulation Modeling

*[Note: includes Flowcharts, Critical Path Method (CPM), Petri Nets, Finite State Machines (FSM), State Transition Diagrams (STD), State Transition Tables (STT), Finite State Automata (FSA), Marked or Partial Order Graphs, Colored Petri Nets (CPN) and Entity Life History Models, ASA, Automata, Design/CPN, Object-Role Modeling, Precedence Temporal Model (PTM), Structures Inheritance Network (SI-Net) and SmartSLIM]*

Modeling of process flow or more complete system dynamics is usually performed to depict and understand processes from the perspective of: the information the processes use, transform and produce; the ordering (sequence and parallelism) of the processes; causal and control relationships among processes; and the particular paths, iteration and timing of processes, given certain environmental assumptions or scenarios. There are a number of methodologies, many operationalized in automated modeling and simulation tools, to accomplish these objectives. Almost all such models are represented in graphical form, although a text list, in the sense of a program listing, is also quite common. More complex models may also be represented in algebraic or parametric form. At the lower end are simple models such as flowcharts which are used to depict process flow (sequence) and, in some cases, limited control flow. Circles are typically used for processes, boxes for inputs and outputs, lines between circles and/or boxes for sequence or flow, and diamonds for decisions or branching conditions. As additional control

structures and timing are added as explicit elements of the models, modeling methodologies and languages such as Critical Path Method (CPM) may be utilized. CPM involves modeling and analyzing processes, sequence, control (i.e. start and stop dependencies among processes and branching) and timing (i.e. start/stop times and temporal duration) in order to determine the fastest path through a complex network of interrelated processes. The data or information involved in the processes and the nature of the control structures are not typically modeled in CPM. It is used primarily for analyzing processes as part of project planning and production process planning.

States and events are introduced as part of still more advanced methods. These methods include modeling of the conditions under which certain events are triggered and the state changes (or state transitions) which those events cause. Entity Life History Models, Petri Nets, Marked Graphs (or Partial Order Graphs), Finite State Machines (including State Transition Diagrams and State Transition Tables) and Colored Petri Nets are examples of these more advanced methods.

An Entity Life History Model is a tree-structured representation of the behavior of an entity and the events that affect it during its life history [GUIDE, 1991]. Petri Nets are an abstract notation for modeling all aspects of complex processes, including cases of indeterminism, distribution and concurrency (in fact, Petri Nets permit the representation of arbitrary numbers of parallel processes) [Sowa, 1991d]. Farah [Farah and Kusiak, 1988] states that Petri first devised a general graphical representation for systems in 1966. Holt and Commoner extended Petri's work in 1970 and 1971 and called the resulting representation form a "Petri Net". Since that time Petri Nets have been used in the study of parallel computing, multi-processing, computer systems modeling (for example, for defining a distributed systems environment) and application systems modeling. Petri Nets may be depicted in algebraic form, or using directed graphs (which tend to be more user-friendly). According to Miller [1973] and Sowa, a Petri Net is a graphical representation with directed edges (i.e. lines with arrows) between two different types of nodes. A node represented by a circle is called a *place* (or *state*). Places represent conditions such as: message in a buffer; teller available; CPU ready; etc. A node represented by a bar is called a *transition*. Transitions represent actions or events such as: compute some function; process a message; accept payment; etc. The places have the capability of holding *tokens*. A token is typically represented by a dot beside or inside a circle representing a place. A token in a place means that the associated condition is true. For a given transition, those places that have edges directed into a transition are called *input places*, and those places that have edges directed out of a transition are called *output places*. If all the input places for a transition contain a token, then the transition is said to be *active* (or *enabled*). An active transition may *fire* (or *execute*) at any time. The firing removes a token from each input place and puts a token on each output place. Thus a token in a place can be used for the firing of only one transaction. Concurrency is modeled by depicting two (or more) transitions that are both enabled and may therefore fire in any order. Indeterminism is modeled by depicting one place that is an input to two (or more) enabled transitions. Which one fires as a result of the input is indeterminate. To perform simulation, indeterminism and concurrency need to be resolved. Marked Graphs, Finite State Machines and Colored Petri Nets provide the means to resolve these ambiguities for purposes of practical analysis and simulation.

Marked Graphs (or Partial Order Graphs) are a special case or restricted form of Petri Nets. In Marked Graphs every place has only one input and one output transition. This eliminates indeterminism, but still permits concurrency. A Finite State Machine (FSM) [Hatley and Pirbhai, 1988] is another way of representing the behavior of a system or group of processes for purposes of

simulation. FSM's may be either combinatorial or sequential. Combinatorial FSM's do not contain memory. That is to say, their outputs are determined only by their current inputs. Sequential FSM's do draw upon memory. That is to say, their outputs are determined by current and past inputs. Most non-trivial systems are sequential FSM's, so sequential FSM's are of more use for enterprise modeling. A State Transition Diagram (STD) is a way to represent a sequential FSM. STD's model states and indicate the way in which an event (often in the form of a control signal) causes certain actions to take place that transition an input state to an output state. STD's typically represent states as boxes and transition arcs as lines with arrows. Events are shown as names or labels on the arcs of transitions they cause. Actions are shown as names or labels adjacent to the events that cause them. Events are separated from actions by a slash or line (i.e. event/action). An event is a "input" and an action is an "output". A complementary way of presenting the relationships between current states, events, actions and next states is in the form of a decision table or State Transition Table (STT). A Finite State Automata (FSA) basically automates a FSM (in this case, a STD and a STT) for purposes of computer-based simulation.

Finite State Machines (including STD's, STT's and FSA's) are really another special case or restricted form of Petri Nets, where every transition has only one input and output place (i.e. only one input and one output is related to each transition). This resolves the concurrency issues, but still leaves open the issue of indeterminism. In practice, indeterminism is resolved by data-dependent conditions that are not explicitly shown in the model. STD's, STT's and FSA's are used frequently for the design and development of embedded software and other real-time software systems. They are an integral part of many emerging Systems Engineering Environments (SEE) for product development of complex physical artifacts such as aircraft and automobiles. There are some commercial software products which operationalize this type of modeling to at least some degree. For example, ASA<sup>®</sup> [Roques, 1991] is a product from Verilog that couples IDEF0 models to a FSA simply called Automata<sup>®</sup>. It takes input and control data and produces a sequence of outputs. It models states and transitions between states triggered by events. ASA does not have a graphical simulation capability, but there are plans to link ASA to a graphical-based simulation package known as VAPS (a specific vendor reference and associated trademark restrictions were unavailable). VAPS is apparently quite popular in Canada and Europe. Verilog intends to use the Portable Common Tool Environment (PCTE) as the means of integrating ASA and VAPS. PCTE is described under a separate subheading within Section 3.3., Model Extension and Integration Projects and Approaches, of this document.

Colored Petri Nets (CPN) [Meta Software Corporation, 1989] offer another, more advanced way to deal with indeterminism and concurrency. CPN evolved from conventional Petri Nets and are mathematically equivalent to them. However, CPN contain powerful structures for capturing complex control conditions and other information. In "Colored" Petri Nets, the "color" determines the path of the processing based on [strong] typing (i.e. it resolves the indeterminism and concurrency by specifying particular cases or paths using the concept of types or classes). This is accomplished through the use of formal inscriptions (i.e. mathematical expressions) and extensions to the graphical syntax of conventional Petri Nets. This allows more explicit modeling of the process semantics (including concurrency and the conditions that determine indeterminate paths), thereby streamlining and improving the clarity of often large and difficult-to-use Petri Nets. Inscriptions allow specification of: color sets (types), initial markings, transition arc expressions, guards, reduction rules, etc. Inscriptions are encoded in Standard Meta Language (SML). SML was developed at the University of Edinburgh, Scotland. It has been extended by Meta Software Corporation and incorporated into their Design/CPN<sup>®</sup> tool, a computer-based software tool that implements CPN. Design/CPN accepts a functional model from Meta

Software's SADT/IDEF tool (Design/IDEF®) as a starting point for more advanced process modeling using the CPN method.

Some innovative work in the practical application of process dynamics modeling was also performed by Rockwell International-North American Aircraft Operations (NAAO) with the support of Knowledge Based Systems, Inc. under the Air Force Human Resources Laboratory's Integrated Design Support (IDS) system project. Rockwell used Object-Role Modeling concepts in an attempt to address process timing in the context of a product life-cycle. This was done through development of a Precedence Temporal Model (PTM) (as found in the field of artificial intelligence). This work fed into the specification effort for the proposed IDEF3 method, which is intended to support more fully-developed process flow and object state-transition modeling. IDEF3 is described in more detail under a separate subheading within this section. One major difficulty is that PTM-based methods are oriented towards modeling specific operations scenarios. It is extremely difficult to integrate and/or synthesize a collection of scenario-based models into an enterprise model spanning the full life-cycle. Models of individual operations scenarios are still useful, but a deeper, underlying representational structure and associated navigational tools are needed to integrate and synthesize the individual models and allow easy navigation through a single logical "composite" model. Also, the complexities of modeling causality and other deep semantics about the nature of linkages among processes largely precludes deep causal modeling within the framework of commercially-available methodologies and tools. Advanced models of these kinds of process semantics are currently found only in research and development settings.

All of the methodologies, languages and tools discussed above work well when modeling shop floor processes and clerical/data processing activities, but often have difficulties when applied to higher-level white-collar activities, particularly those that link across the product life-cycle. At the farthest end of the spectrum are complex semantic modeling methodologies which attempt to explicitly model deeper causal relationships between activities, as well as the dynamics of such relationships. These methodologies have been developed specifically for modeling of higher-level, analytical, white-collar processes such as those involved in managerial decision-making. Vernadat [Vernadat and Kusiak, 1988] states that an expanded graphical representation concept described by Elam [Elam et al., 1980] and based on Semantic Networks [Quillian, 1968] served as the basis for these approaches. Semantic Networks are described in more detail under a separate subheading within this section. Advances in artificial intelligence, in particular the commercial availability of rule-based systems, are also playing a large part in the expansion of semantic modeling capabilities for use in modeling system dynamics. The Structures Inheritance Network (SI-Net) and SmartSLIM (Systems Laboratory for Information Management) are cited by Vernadat as examples of semantic network-based modeling approaches. Some commercial, rule-based simulation software also employs such concepts.

#### Quality Function Deployment (QFD) and House of Quality (HoQ)

*[Note: also includes Total Quality Management (TQM), Supplier-Input-Process-Output-Control (SIPOC), Cause and Effect (Fishbone or Ishikawa) Diagrams] and Wisdom Systems' IDEFine-0-to-International Technica Institute QFD]*

Quality Function Deployment (QFD) is an analysis and modeling methodology associated with the philosophy of Total Quality Management (TQM). TQM is not concerned only with the narrow definition of "quality" as it relates to product defects or reliability, but

rather TQM emphasizes the broader notion of doing the right things at the time and doing them well. This broader concept of "quality" comes into play in many enterprise analysis projects both as a driving force motivating the project and as one of the only means of linking enterprise success criteria or objectives with enterprise business functions, as well as the organizations and information needed to support those business functions. "Total quality" not only reduces the number of product defects and improves the reliability of the products, but also ensures that the right products are produced on time, at a competitive price and with the characteristics and functional performance that customers want. Achieving total quality in an enterprise requires an analysis of all aspects of the enterprise's functional structure (in the sense of processes) and organizational structure (in the sense of the human resources, policies, etc. that perform processes). Included in this analysis are such things as: marketing's ability to judge customer requirements or desires; the product definition and delivery processes which produce the actual products to meet those customer demands; relationships with suppliers and co-producers; product support and maintenance processes; the policies and procedures which support (or, in many cases, inhibit) the performance of product definition, delivery and support processes; and the organizations which are tasked with performing those processes (analysis in this last case might address conflicting goals, reward structures, etc.). Analysis of these aspects of the enterprise requires the support of methods and tools. To some degree the methods and tools required to support TQM encompass all of the various methods and tools covered within this section. However, there are some specialized methods and tools more closely associated with TQM in particular. Quality Function Deployment (QFD) is perhaps the best known of these methods and tools.

QFD is a methodology for "deploying" the concept of quality throughout all functions, organizations and other aspects of the enterprise — hence the name Quality Function (or Functional) Deployment. According to Hill [1991], QFD is said to have originated in the Mitsubishi Kobe Shipyards in 1972. It was there that Dr. Mizuno of the Tokyo Institute of Technology developed what were referred to then as "quality tables". Quality Tables eventually evolved into QFD. Hill notes that some claim that the QFD work was derived from, or at least influenced by, the Unified Program Planning (UPP) technique developed by Dr. Warfield in the U.S. during the early 1970's. In any case, it is interesting to note that many of the concepts behind QFD, and TQM in general, originated in the U.S. through the work of Dr. Deming, Dr. Juran and others. However, they were first put into practice in Japan starting the late 1950's and 1960's, and then [re]introduced into the U.S. in the 1980's. Ford Motor Company started using QFD in 1984. However, it was not until a Harvard Business Review (HBR) article was published [Hauser and Clausing, 1988] that QFD began to receive widespread notice in the U.S.

As noted several times above, QFD can be described as a *methodology* — meaning that it actually entails several methods involving a number of procedures and rules. Some argue that it is this overall analysis process and the insights that it brings into fundamental TQM-related concepts (such as customer, supplier, optimization of the enterprise rather than its individual parts, etc.) that are the real heart and soul of QFD. Others concentrate more specifically on the more formal modeling aspect of QFD, which is referred to as House of Quality (HoQ). In either case, QFD embraces the TQM notion of two dimensions in which an enterprise organization (and the functions it performs) can be viewed [Sullivan, 1989]. One dimension (the "vertical dimension") is intradisciplinary. For example, design engineering might concentrate on optimizing the engineering aspects of a product, with little or no regard to its manufacturability (and, in fact, that is all too often the case in practice). The second dimension (the "horizontal dimension") is interdisciplinary. This dimension focuses on the overall enterprise, including customers, suppliers, etc. An example of applying this notion would be the concept of Unified Life Cycle Engineering

(ULCE), where all aspects of a product — from customer needs through engineering, manufacturing (including suppliers and co-producers) and support — are taken into consideration. Trade-offs are, of course, made based on various weightings, but the focus is still on optimization across the entire life-cycle, viewing the enterprise as a single “system” to be optimized globally rather than within each area. Sullivan calls this the “Voice of the Customer”, where “the customer” starts first and foremost with the actual end-customers who buy the product or service, but where the notion of “customer” also extends across the life-cycle to any organization or person who uses something another organization or person produces. For example, manufacturing is a customer of engineering, since manufacturing uses the “products” produced by engineering, namely product designs and other aspects of a product definition. Likewise, the customers of an information resources management (IRM) organization are all the users of the systems supplied by that IRM organization.

QFD starts with a definition of the customer requirements, i.e. what the customer wants in terms of the characteristics or attributes of the product or service. This could involve anything from color or shape, to performance characteristics, to price. The requirements are listed down a page and then ranked (with a rank number assigned to each of them). Some QFD modelers prefer to re-write the list once the ranking has been completed so that the requirements appear in rank-order. One principle of TQM and QFD, however, is that lower-ranked requirements should not be ignored completely, but instead they should simply be viewed as less critical relative to higher-ranked requirements.

The next step is to determine the engineering characteristics necessary to meet the customer requirements, i.e. how the product or service would meet those requirements. A list is made of those characteristics. The engineering characteristics are then mapped to the customer requirements. This is potentially a many-to-many mapping, as a given requirement might necessitate a number of engineering characteristics to be fully satisfied, and a given engineering characteristic might apply toward satisfying more than one customer requirement. Using the House of Quality (HoQ) modeling approach, the mapping between customer requirements and engineering characteristics can be represented using a matrix rather than two lists with lines drawn between items on the list (which can become fairly difficult to follow, given the intersecting lines and many-to-many mappings). In a HoQ matrix, customer requirements are simply listed down the left-hand side of the matrix, forming the labels of the rows in the matrix. Engineering characteristics are listed across the top of the matrix, forming the labels of the columns. The cells at the intersection of each row and column represent a potential pairing of a given customer requirement to an engineering characteristic. If that pairing or mapping is relevant (i.e. true), then a check mark is placed in the cell. In some QFD/HoQ matrices, modelers use the cell to further qualify or describe the pairing. That is to say, they make the matrix large enough such that there is space in the cell for a few words or a brief note. An alternative is to place a reference number in the cell, which refers to an entry in a complementary document, such as an requirements allocation sheet (or resource allocation sheet) or an engineering specification. These documents then describe how the engineering characteristic meets the requirement.

The final step is to internally map the engineering characteristics against one another. The reason for this is that engineering characteristics do not function or get realized in the real-world in isolation. They interoperate as part of a system (i.e. the product or service). Some obviously interact with or affect one another more than others. Since changes to one characteristic must be analyzed for their potential effect or impact on other related characteristics, it is important to represent these interrelationships in the QFD/HoQ model. This could be accomplished using another full matrix, where the engineering characteristics form both the rows and the columns.

However, anytime the same thing is mapped against itself in a complete [square] matrix, half of the square (a triangle formed along the diagonal) is redundant, since it is simply a mirror-image of the other half of the square. Therefore, the triangle representing the mapping of engineering characteristics to themselves can be logically extracted and placed on top of the columns of engineering characteristics mapped to customer requirements in the first matrix. This looks like a "roof" placed on top of the matrix of customer-requirements-to-engineering characteristics and therefore led to the name House of Quality for this QFD modeling form.

Other information can be hung off of and related to this existing HoQ matrix. A Customer Preference Chart may be used to assess relative competitiveness in the marketplace. A Technical and Cost Assessment can be used to allocate resources to elements of the system (i.e. engineering characteristics) based on engineering difficulty and on the benefits generated by the characteristics. A full QFD analysis does not stop at the top-level with end-customer requirements. The engineering characteristics from this initial HoQ matrix become requirements at the next-level down, and those requirements in turn must be met by lower-level engineering characteristics. This downward allocation continues until elemental engineering characteristics have been defined which represent directly implementable system (i.e. product or service) components. QFD-based analysis and modeling can also proceed across a product life-cycle. The output of a requirements analysis (a HoQ matrix of low-level engineering requirements) can serve as input to design analysis. The engineering requirements become the rows and product designs become the columns. Further, product designs can become requirements to be realized by manufacturing processes. In this case, the product designs are the rows in the matrix and the manufacturing processes are the columns. In this sense, the QFD process is quite consistent with the Systems Engineering Methodology (described under a separate subheading within this section). In fact, Hill has proposed that QFD/HoQ could be used as a analysis and modeling methodology in support of an overall systems engineering process or methodology.

Technically, based on the notion of "customer" described above, any case where a "customer" has requirements to be met by some "supplier" can be modeled using QFD/HoQ. Part of the difficulty, however, in doing QFD/HoQ analysis and modeling is simply determining who in the enterprise (i.e. which organization or person) is a customer or supplier for which other organization or person, and what products and services do they provide and use. To support this kind of analysis, The Aluminum Company of America (Alcoa) and many other companies have utilized the Supplier-Input-Process-Output-Customer (SIPOC) analysis method and associated diagramming techniques also included as part of the Total Quality Management (TQM) philosophy. At Alcoa, SIPOC analysis was useful primarily for identifying and raising an awareness of internal "supplier-to-customer" relationships among Alcoa organizations. Analyzing the particulars of those relationships and assessing or measuring the level or degree of current performance (for example, assessing customer satisfaction) were tasks performed informally in groups. SIPOC as an analysis method has intuitive, natural appeal for users, but (as practiced under most TQM projects) SIPOC lacks rigor, standardization and a formal basis for its application. Ontek was unable to identify any formal modeling languages or tools directly implementing the SIPOC method, although several other methodologies or languages, including IDEF0 (described under a separate subheading within this section), are being utilized to some extent to support SIPOC analysis.

Another TQM analysis and modeling method is the Cause and Effect Diagram (also called a Fishbone Diagram or Ishikawa Diagram) [Alcoa, 1989]. These are figures composed of lines labeled to represent important process elements. The diagrams show relationships between an effect and its cause(s). The basic method is to work from symptoms to their root causes, so that

corrective actions may be determined and initiated. A Cause and Effect Diagram includes a primary trunk line, which is anchored by a clear description of the effect of interest. The major categories of possible causes become branches. Typical branches include the "four M's": man; methods; materials; and machines. Specific possible causes become the detailed twigs.

TQM analysis is most often accomplished using a number of enterprise analysis and modeling methodologies, tools, languages, etc. used in combination with one another. Enterprise analysis and TQM analysis are at a minimum closely related, and in some cases, probably synonymous. TQM analysis methods and tools are frequently used to support enterprise analysis, particularly at the business-level or top-level of the overall enterprise. For enterprise analysis at the business or enterprise-level, for example, Northrop Corporation has utilized the Quality Function Deployment (QFD) analysis method and House of Quality (HoQ) diagrams described above. Northrop coupled these with more detailed IDEF0 (functional modeling) and IDEF1X (data modeling) methods and diagrams. Other methods, techniques and tools were used to capture and present critical success factors or objectives (which were transformed into the requirements or "voice of the customer" that make up the horizontal axis of a House of Quality diagram). Still other methods, techniques and tools were used to identify, analyze, document and present such related information as: organizations and organizational structures; existing, legacy databases and the nature of their data (including data element names, but little in the areas of structure or semantics); existing, legacy application systems and the nature of their [application] processing; and current and planned information system projects (including modifications to existing systems and development of new systems). These efforts were hindered to some degree by the lack of methods and tools *within* individual areas of analysis (such as representation of legacy hierarchical databases). However, the efforts were hindered to a much greater degree by the lack of integration *among* the different methods and tools which are suited for use at particular stages in the analysis and modeling process. As a start, at least one of the vendors of IDEF-based enterprise modeling tools (Wisdom Systems, Inc.) is finding ways to link its tool to the overall TQM approach [Sloan, 1991]. Specifically, Wisdom has already developed an interface between its IDEF0 functional modeling software (referred to as IDEFine-0™) and QFD/HoQ analysis and modeling software from International Technica Institute, the result being referred to as IDEFQual™.

#### Rule-Based Systems/Expert Systems

*[Note: also includes Constraint Languages]*

According to GUIDE [1991], a business rule is a statement about the way a business runs. Formal descriptions of business rules are typically found in corporate policy and procedure manuals, strategic plans, corporate statements of direction, division mission statements, coding standards, Requests for Proposals, and numerous other text references. Business rules may be modeled in a rule-based system or expert system. In such a system, rules are expressed in the form of mathematical logic. Mathematical logic employs a simple, direct format for stating rules, i.e. the IF-THEN form:

IF Condition, THEN Assertion  
or  
IF Condition, THEN Action



## UKM State-of-the-Art Review

where Condition is a true or false statement, and Assertion and Action are other statements. While these mathematical rules look similar to the IF-THEN selection constructs of programming languages, there are important differences. In mathematical logic:

- The order of the rules is not important. A set of rules can be shuffled to any order and they express the identical knowledge.
- Rules cannot be nested. Each rule supplies an independent, atomic piece of knowledge. A single, complex rule may be presented using nesting, but the rule must truly be a single rule rather than a combination of several distinct rules.
- An IF-THEN rule can be read as "if whenever", in the sense that the rule applies at any time instead of "if now" when programming.

In automated rule-based or expert systems an inference engine may be used to combine rules and facts to arrive at a logical conclusion based on some description of a problem or condition.

Many rule-based or expert systems allow a modeler to define the semantics of the domain being modeled using other syntactical constructs besides rules. For example, rules frequently are about some *object*. Therefore, the object-oriented modeling paradigm (described under a separate subheading within this section) is often used in conjunction with rule-based systems or expert systems. The IF-THEN "rule form" of mathematical logic may actually be used to model the broader notion of constraints, rather than rules in the strict sense. GUIDE defines a *constraint* as a specification of the valid states of an object (or set of objects) and the valid transactions among those states. Types of constraints include:

- Domain Constraints;
- Static Constraints, including Referential Integrity Rules;
- Dynamic Constraints;
- [Business] Rules;
- Triggers (i.e. rules plus a mechanism for initiating a test to see if the conditions of the rule are met, and in some cases a mechanism to initiate the appropriate actions); and
- Daemons (i.e. global rules that may or may not be attached to an object).

In addition to the IF-THEN rule form of mathematical logic, there are other forms or languages for expressing constraints. All of these may be generally referred to as constraint languages. These include:

- First Order Predicate Logic or Interpreted Predicate Logic (described under a separate subheading within this section), including Horn Clauses;
- Conceptual Graphs (described under a separate subheading within this section);
- Production Rules; and
- Constrained Natural Languages, including Problem Statement Language and pseudo-code [Note: EXPRESS, described under a separate subheading within this section, is also considered by some to be a Constrained Natural Language].

Rule-based or expert systems were very popular for a brief period in the mid-1980's. However, the number of such systems which were fully-developed (versus prototyped) and transitioned to production use remains fairly small. In general, rule-based or expert systems are most successful when applied to tightly-defined domains of expertise where specific knowledge can be directly specified in the form of rules or constraints. Reliable domain experts must also be available. An example of such a domain would be diagnosis for repair of machinery on the shop floor. Broad domains requiring substantial background or commonsense knowledge and involving "fuzzy" rules are not good candidates for rule-based or expert systems. Rule-based or expert systems are also not recommended in cases where cross-domain knowledge is necessary. An example would be the domain of operations management for a large manufacturing enterprise. There are probably few people who are really "experts" in the entire domain, and even the experts would have trouble describing the "rules" they use to make decisions in that domain.

### Semantic Network or Semantic Net

Sowa [1984] states that concepts ("classes" in Ontek terminology) are building blocks for constructing mental models. Rules or patterns are needed to organize the building blocks into larger structures. He refers to Kant's term "schema" as a rule that organizes perceptions into a unitary whole (i.e. into a concept). Whereas an earlier theory of the mind (associationism) had assumed diffuse links between concepts, Selz expanded on Kant's schema and defined it as a network of concepts and relations that guides the thinking process. This served as the foundation for the Semantic Network representation scheme originally conceived as part of research in the field of artificial intelligence [Quillian, 1968]. According to GUIDE [1991] part of the appeal of Semantic Networks is the fact that they use the "thing-link-thing" construct, which most humans find quite natural, to represent knowledge. Classes, instances, and attributes appear as nodes, interconnected by lines representing relationships. Relationship types often include: "is-a" (abstraction); "is kind of" (generalization)/specialization); "is-part-of" (composition or aggregation); "is-attribute-of" (characterization or projection), and other associations between things.

Like Entity-Attribute-Relationship (E-A-R) diagrams, Semantic Nets are relatively easy and natural to draw or model. This apparent ease of use can be deceiving, however, as the models produced frequently lack rigor and a formal base. There are also multiple ways in which a semantic network representation scheme could be implemented as a computer-based representation system; these include using: a file management system with pointers; a traditional database management system, such as ones based on a relational or Codasyl/network model; or a knowledgebase management system, the underlying representational scheme of which may be more directly based on the notion of a semantic network.

One limitation of most semantic networks currently found in the field of AI is their restriction to binary (i.e. 2-dimensional) relations. To model many of the complex relationships which underlie any domain, whether it is a manufacturing enterprise, a bank, or the process of making a spaghetti dinner, multiple or  $n$ -dimensional relations are required. Some of these might be expressible through the use of several, related 2-dimensional relations, but use of this convention may well introduce unnecessary complexity or fictitious relations which act only as placeholders. In many other cases, a  $n$ -dimensional relation simply cannot be expressed as a set of 2-dimensional relations — at least not without losing the semantics inherent in the real  $n$ -dimensional nature of the relation. This, of course, does not preclude presenting some  $n$ -dimensional relations to modelers using 2-dimensional presentational forms. That is, however,

only a presentational issue. What is really required is a semantic network-based representation scheme capable of truly representing  $n$ -dimensional relations.

### Systems Engineering Methodology (SEM)

*[Note: also includes Functional Flow Block-Diagrams, Schematic Control Block-Diagrams, N-Squared Charts, RDD-100, P-Flow, Sys-Flow, C-Flow and Sys-Talk]*

Systems engineering is a structured, rigorous methodology directed primarily at the critical early stages of systems development. Systems engineering has been widely-applied to the development of software systems — particularly real-time, embedded software [Department of Defense, 1991]. It has also been applied to a lesser degree to the development of application software systems, and recently to complex weapon systems such as advanced aircraft [Griskey et al, 1991]. In fact, systems engineering is now considered a major methodological approach to implementing several of the key concepts and philosophies of concurrent engineering [Sarris, 1991]. It encompasses many of the general aspects of structured analysis and design techniques, including top-down decomposition and modularity.

Systems engineering starts with high-level logical requirements that relate directly to some mission or overall set of goals. All other requirements are then derived from these high-level goals based on a formal process of requirements identification and iterative decomposition. For a requirement to be separately identified, it must represent a distinct logical need for a specific functional capability or architectural (i.e. structural) property. The timing considerations of requirements are considered in time-line analyses. One must also be able to specify a set of criteria and a method to ensure that each requirement has been met. In other words, each requirement must eventually be *verifiable* through some measurable aspect of the system implemented to meet that requirement.

The derived requirements represent the necessary functional capabilities and/or structural properties of the technological and organizational systems that are to be developed to realize the goals. Functional capabilities (i.e. functions or processes) are typically modeled using Functional Flow Block-Diagrams (FFBD's). FFBD's were developed in the 1960's, partly for use by the Department of Defense. According to Blanchard and Fabrycky [1990], FFBD's were developed for the primary purpose of structuring system requirements into functional terms. FFBD's employ a mechanism for portraying system design requirements in a graphical manner, illustrating serial and parallel relationships, the hierarchy of functions, and functional interfaces. The graphics are quite simple. Boxes are used to depict functions, with the name of the function and its reference number specified inside the box. The numbering scheme establishes the level within the functional hierarchy (e.g. 4.1, 4.2, 4.3 are all subfunctions of 4.0). Circles are used to represent logical junctions (i.e. and, or, Xor, etc.), which are labeled "AND, OR, etc." inside the circle. Lines are used to connect boxes and/or circles (i.e. to show functional flow). Some lines may represent separate paths to be followed based on certain conditions. In this case, the condition is specified through a label on the line.

There are a number of limitations with FFBD's [Alford, 1991]. First, as is obvious from the description above, FFBD's support only weak representations of control structures. FFBD's do not fully depict the conditions under which functions occur, including whether particular functions are "normal", or "exceptions" that only arise in specific cases. As noted above, conditions can be shown on the diagram (simply as named lines connecting two related function boxes), but there is no form or mechanism to further specify or qualify the condition shown. FFBD's also do not

depict the input and output relationships between or among functions (in fact, "data" is completely ignored in FFBD's). Both control structures and input/output data are critical elements of a complete, explicit functional or process model. Additionally, FFBD's do not easily support modeling of performance criteria associated with the functions. Finally, there is no clean mapping of FFBD's to downstream systems development tools such as CASE tools (which are described under a separate subheading within this section).

Alford suggests that one alternative or complementary form to FFBD's is N-Squared Charts, as developed by TRW. The purpose of N-Squared Charts is to present input/output relationships between functions in a compact, easily mappable form. An N-Squared Chart is constructed by placing functions (in the form of boxes) down the diagonal of a matrix (usually a page), and then placing the input and/or output data (as names or reference numbers in circles) at intersection points formed by drawing a horizontal and vertical line between each pair of related functions. Where there is no relationship, no lines or circles are shown. Used alone these charts do not capture such critical information as flow sequence or control structures. They do, however, serve to complement FFBD's by adding input/output data. The problem in this case is that this forces the modeler to create and maintain two separate, but closely-related diagrams. This opens up the opportunity for inconsistencies to arise between the two diagrams. These problems are not unique to FFBD's and N-Squared Charts. In fact, these are the same sorts of problems involved in integrating IDEF0 functional models and IDEF3 process flow models (both of which are described under separate subheadings within this section). The use of a dynamic, three-schema data dictionary, including the ability to have a single, logical functional database with easy navigation among related presentational forms (such as FFBD's and N-Squared charts, and IDEF0 and IDEF3) would solve many of these problems.

At an appropriate level of decomposition, a requirement is allocated to a system, or element of a system, which is capable of meeting that requirement. The structural aspects of these systems are typically modeled using Schematic [Control] Block-Diagrams (SCBD) and formal specifications. In SCBD's, named or numbered boxes are used to represent systems or system elements and [labeled or numbered] lines are used to represent relationships or interfaces among them. The nature of the relationships or interfaces may be described in complementary specifications, which are typically related back to the SCBD's using reference numbers for the system or system elements and/or the relationships or interfaces. Specifying a system at any given level may generate additional requirements at one or more lower levels. Therefore, the requirements decomposition and allocation processes are tightly coupled. Additionally, many systems or system elements may be necessary to meet a given requirement, and any one system or system element may be utilized to partially or fully meet many requirements. In other words, many-to-many relationships must be represented.

Performing the allocation process involves analyzing a number of possible alternative means (i.e. the systems or system elements) for meeting a requirement. Choosing the best system or system element requires conducting a rigorous trade-off analysis or "trade study" among the alternatives. This involves defining a set of evaluation criteria that are crucial to meeting the requirement, given some set of operational or other environmental constraints or characteristics. Once the selection of the optimal system or system element has been made, that system must then be analyzed in the context of systems selected for other, related requirements. This involves analyzing necessary system integration and control issues, as well as defining system interface requirements. Schematic [Control] Block-Diagrams are typically used again to support this analysis. This process works its way back up the requirements tree, aggregating and analyzing at each level of the tree until the target domain (i.e. the overall system) is analyzed as a whole. Only

when this portion of the process is complete are detailed specifications, and eventually designs, developed for the individual systems.

It is obvious given the nature of requirements decomposition, allocation, trade-off analysis and other aspects of systems engineering, that this is an extremely complex process involving many kinds of relations. If the process is not performed deliberately using some formal, rigorous method, the opportunity exists for failing to identify or allocate important requirements, not catching or resolving requirements that stand in conflict, specifying systems or system elements which do not interact well when operating in aggregate (including as a global system to meet the overall goals or mission), or specifying unnecessary systems or system elements for which there are no valid requirements. For this reason, there must be *requirements management* and *traceability* throughout the entire systems engineering process. That means completeness checking, conflict checking and other verification and validation to ensure that all requirements have been properly decomposed and allocated, the system meets requirements at all levels, and there are no superfluous systems or system elements for which there are really no requirements.

Additionally, the entire systems engineering process must be conducted with a risk management process operating in the background. At key points and for major milestone deliverables in the systems engineering process there should be quantifiable or qualifiable factors that represent possibilities for erroneous decisions or the simple fact that assumptions will not work out quite the way they were envisioned. Not all requirements have an equal likelihood of being met and not all specified systems or system elements have the same probability of being realized or implemented by available technologies or organizations in the real-world. Risk analysis serves to identify and estimate the degree of risk and to drive the development of a risk management plan.

Design and development activities are then performed resulting in the delivery of the system. As the technological and organizational systems are put into place or implemented, thereby meeting some or all of the requirements to varying degrees, the final feedback loop needs to be closed. This closure involves comparing the actual operational system to its mission or goals, including the derived requirements at various levels. Using requirements traceability and bearing in mind the risk analysis results and risk management plan, a "post mortem" must be performed to document both successes and shortcomings. Because the groundwork has already been formally established, any deficiencies of the system should be explainable, and a corrective action plan should be able to be developed — again by applying the systems engineering process.

There are several vendors who market commercial tools to support the systems engineering methodology. One of the best examples of the state-of-the-art (SOTA) with respect to these technologies is RDD-100®, a product from Ascent Logic Corporation. RDD-100 includes several graphics-based modeling interfaces for capturing and presenting requirements data in the form of Functional Flow Block-Diagrams (described above) and Resource (or Requirements) Allocation Sheets. Data models produced in RDD-100 are based on the Entity-Attribute-Relationship (E-A-R) modeling approach (described under a separate subheading within this section). Like many SOTA technologies for systems engineering, RDD-100 utilizes a relational database as its underlying representation system. Relational databases work well for representing some of the basic flowdown relationships in the requirements allocation process, and for querying the database during requirements traceability and other processes. However, it is difficult to represent complex relationships (particularly those using higher-order logic such as context or modes), many tools are brittle when confronted with changes or variations in the basic systems engineering process, and the ability of end-users to logically navigate through the database using various presentation forms is quite limited [Griskey et al, 1991; Sarris, 1991]. These are problems which necessitate the use of

dynamic, three-schema data dictionaries (which are described in Section 3.4., Repository/Data Dictionary Technologies, of this document). Ascent Logic is currently working on "executable specifications", built-in metrics (primarily cost), and model integration among legacy modeling tools using a relational database [Alford, 1991] and links to CASE tools (which are described under a separate subheading within this section). These efforts are definitely still in their infancy; in other words, at this time they are more "ideas" than they are concrete projects. Alford states that "upper-CASE modeling won't work until someone formally defines what the relationship is between analysis/modeling and software engineering." He is not aware of anyone who is seriously working on that issue.

As another example of systems engineering tools, Boeing is developing a complete software environment for systems engineering. Boeing developed P-Flow, which uses Ascent Logic's RDD-100 relational model as a basis for performing simulation. Another Boeing-developed software tool, Sys-Flow, then does rapid prototyping of systems based on the simulation model. Boeing has even defined their own constraint language, although there are many widely-accepted constraint languages already available in industry. Next, Boeing's C-Flow tool generates C code. Additionally, Boeing is working on an object-oriented tool called Sys-Talk. Boeing states that the biggest problem is that all these tools require modelers to exercise "a high degree of abstraction". This limits the number of people who can really use the modeling tools effectively. Few end-users are able to use such tools directly. Knowledge engineers perform most of the modeling, based on data collected from end-users.

#### The Vee Method (or Heuristic) For Knowledge Acquisition

Vee [Novak and Gowin, 1984] is a methodological approach to analyzing knowledge, particularly the construction of new knowledge based on the application of existing knowledge. This methodology is based on the theory that when we are involved in constructing [new] knowledge, we use concepts we know to observe events or objects and make some form of records of our observations. We then interpret those records. The kind of records we make is also guided by one or more focus questions. Acquiring new knowledge also allows us to enhance and/or alter the meanings of the concepts, principles and theories we already know, and to see new relationships between them. Principles are significant relationships between two or more concepts that guide our understanding of the significant actions in the events studied. Theories are similar to principles in that they explain relationships between concepts, but they organize concepts and principles in order to describe events and claims about events. Another way of describing Vee then is as a tool for acquiring knowledge about knowledge and how knowledge is constructed and used.

The basic Vee method or heuristic (a heuristic is something employed as an aid to solving a problem or understanding a procedure) is as follows: First, one or more focus questions is formulated. A focus question should be broad enough that you can identify several meaningful subquestions to be posed during the process of answering the focus question, but a focus question itself must be sufficiently well-defined that it is answerable [Note: This is analogous to the fact that requirements in the Systems Engineering Methodology (SEM) have to be verifiable. You must be able to verify whether the requirement is met or not (and likewise, verify whether the question was answered or not). This is not, however, to say that the question was answered correctly]. An example focus questions to an expert manufacturing engineer might be, "How do you develop a process plan?"

Next, concepts relevant to answering the focus question are identified (from most to least general, i.e. from world views through theories and eventually down to statements of regularities

## ITKM State-of-the-Art Review

and individual concepts). Concepts relevant to the example focus question on process planning might include: the nature of processes themselves; the notion of routing a part through a series of related processes; serial flow, parallel flow and causal dependency; material handling issues; machine capabilities; human time and motion concepts; etc. Next, real-world (or simulated) events or objects are observed and records are made of the observations. Using the process planning example, this step might include: observing actual processes on the shop floor; interviewing expert process planners; modeling the process of process planning; etc. Based on this data: facts are stated; interpretations, explanations and generalizations are made; and knowledge and/or value claims are made. There must be traceability throughout the process [Note: this is again analogous to the Systems Engineering Methodology (SEM)]. That is to say, the events or objects observed are driven by a set of concepts, which are also used to interpret or explain the facts. Outcomes that do not support the concepts must be explained. No knowledge claim should be made without supporting concepts and "hard" events or objects to support it.

In summary, data gathering and record making are crucial steps in constructing new knowledge. Once the data or records have been gathered, the next step is to transform those records into interpreted data using our concepts, theories, etc. and various analytical tools such as statistical analysis. Models, in general, are an important means of accomplishing this. Transforming the data involves seeking regularities or patterns in the records. Knowledge and/or value claims can then be made, using the transformed data and concepts, principles, theories, etc.

## Yourdon Methodology

*[Note: also includes Structured Analysis and Design, Data Flow Diagrams (DFD), Yourdon Data Dictionary, Mini-Specifications, and Vienna Development Method (VDM)]*

Structured programming was developed in the mid to late 1960's to improve the clarity and predictability of program code. Based on these concepts, Constantine developed structured design techniques in the late 1960's and early 1970's. Dr. Doug Ross and others developed structured analysis techniques in the early to mid-1970's to improve the front-end (e.g. requirements specification and conceptual design) portions of the systems development process. However it was Dr. Ed Yourdon, along with Dr. Tom DeMarco, whom many regard as the two people who really popularized structured analysis, design and programming techniques and made them an integral part of the systems development process in 1970's, 1980's and even now into the 1990's. Even though some tout the object-oriented paradigm (described under a separate subheading within this section) as the systems development methodology for the 1990's and beyond, others are attempting to merge the best aspects of structured analysis and design into the best aspects of the object-oriented paradigm, to ensure that future systems development projects continue to benefit from structured analysis and design [Keuffel, 1991]. For example, many object-oriented approaches in use in industry today apply the concepts of hierarchical decomposition and modularity, both of which are an integral part of structured techniques.

The major advantage of the structured analysis and design approach over the functional specification-based approach which preceded it was that the structured analysis and design approach produced a more formal model of the proposed system. Much of the semantics of a system were buried in text when informal, linguistic-based functional specifications were used. In theory a program could be written using structured programming techniques (for example, using modules with a tree-based, top-down calling sequence) even if functional specifications were the source document for the program. However, it is much less likely that the program would take

efficient advantage of such techniques if the specification from which it was developed did not provide a strong underlying foundation. Even if the program were well structured, there is always a good possibility that errors would occur in interpreting an unwieldy functional specification, and those errors would get encoded in the logic of the program.

The Yourdon methodology [Yourdon Inc., 1980] was one of the first and foremost structured analysis and design techniques, and it has received widespread use. Like most structured analysis and design techniques, Yourdon builds a formal model of the proposed system using several non-linguistic (i.e. largely graphics-based) modeling techniques and tools. These will be described in some detail in the following paragraphs. The Yourdon methodology was one of the first to explicitly recognize legacy systems issues, largely because previous methodologies developed computer systems where none had been before, whereas Yourdon came into practice at the time when the first generation of computer systems was being substantially enhanced or replaced. Yourdon took into account the notion of physical versus logical models. The physical model reflected the way the process was performed given the existing system. The logical model reflected the nature of the process itself, i.e. what was being done, not how it was currently being done. Systems analysis was performed at the logical level. Systems design was performed at both the logical and physical levels. This takes into account cases where the existing physical environment needed to be modeled as the baseline against which a new physical model (implementing the logical model) could be defined.

Specifically, the product of the Yourdon methodology is called a Structured Specification. It is typically comprised of: Data Flow Diagrams, including the organization of a group of these diagrams into a complete model of a target system; a Data Dictionary; and Mini-Specifications. In keeping with the principles of top-down decomposition and modularity, each of these elements of the Structured Specification is intended to help parse and organize the model of the domain (and, therefore, the system developed from that model) into a set of clearly-defined and maintainable components. Specifically, Data Flow Diagrams are intended to identify the mini-systems which make up the overall system, as well as the interfaces among those mini-systems (i.e. how they interoperate to form the overall system). The Data Dictionary is intended to define the composition and organization of the [data] interfaces. The Mini-Specifications model the [processing] logic of the mini-systems, using Structured English instead of narrative text.

A Data Flow Diagram (DFD) is a modeling technique that allows a system to be depicted as a network of processes (or mini-systems) connected to each other by paths of data. Stated another way, DFD's declare mini-systems and the interfaces among them. Processes and data can represent physical activities and physical artifacts (for example, machining processes and parts in a manufacturing company) or logical activities and logical artifacts (such as a manufacturing engineer analyzing a process plan, or an automated system calculating payroll). A circle (or "bubble") is used to represent an automated or manual activity or process that transforms data in some way. The name of the process appears in the circle as a verb or verb phrase. A line with an arrow is used to represent a data flow (or path or pipeline of data). The name of the data item (a noun or noun phrase) is used to label the line. Data items move along these paths on their way from one process to the next. Therefore, data flow lines and arrows connect process circles. Lines with arrows at both ends may be used for two related data items that flow in opposite directions (such as Credit Request and Credit Authorization), and the line is then be labeled at both ends. There is a "Law of Data Conservation" that applies to DFD's. It states that no data may flow out of a process for which there was no in-flow. In other words, a process cannot miraculously create data, without having some source data from which to create it.



If there is a time delay between the data flow from one process to the next, or if the "receiving" process uses the data in a different order than the "sending" process, the notion of a data store is modeled. This is represented by the name of the data item bounded on two sides by a set of parallel lines (either one on top and one on bottom, or one on the right-side and one on the left-side, depending on the position in the diagram). These data stores may come between two processes, in the sense of interrupting the data flow for some [temporary] storage or reordering of the data. Or they may be off to the side of one or more processes (representing, in effect, access of the data from some more formal data storage mechanism such as a database controlled by a database management system).

A process from one diagram may be decomposed into its constituent processes in a separate diagram. Diagrams are usually one page [Note: there are no explicit syntactical conventions for off-page connectors]. The "seven plus or minus two" memory guideline [Miller, 1956] is used for decomposing processes and establishing modularity. This, of course, also serves to limit the number of processes in a diagram (on a page). This is, however, a guideline, not a rule to be strictly enforced (as is the case with IDEF0, described under a separate subheading within this section). In other words, if it requires 10 processes to accurately model a particular system or mini-system, then 10 processes are somehow drawn on the page (by reducing their size slightly, for example). As with IDEF0 and Functional Flow Block-Diagrams (described under the subheading of "Systems Engineering Methodology" within this section), reference numbers are typically used to keep track of the hierarchy of DFD's. At the top-level or Context Diagram, square boxes may be used to represent other systems or entities that are outside the bounds or scope of this model. DFD's in essence show: processes ("mini-systems"); data flows, including sources and destinations; and data stores. They do not show: data composition relations; data access rules; decisions; loops; calculations or other processing algorithms; quantities; or timing data. Because several lines may be drawn coming off of the circles or bubbles on a diagram, DFD's are sometimes referred to as "spider diagrams".

The Data Dictionary is used to further describe or define the data items from the Data Flow Diagrams. The Data Dictionary may also define the relationships between individual data items and groups of which they are a part. This provides a link between more macro data items found in higher-level DFD's and their more elemental data items (i.e. constituent parts) found in lower-level DFD's, and thereby establishes data interfaces. This is the same as the notion of "bundling" found in IDEF0. Again as with IDEF0, reference numbers may be used to link data items to the Data Dictionary, again helping to specify data interfaces. There is a standard set of symbols used to provide some uniformity in definitions in the Data Dictionary. Aliases may also be specified in the Data Dictionary.

A Mini-Specification (or Mini-Spec) is intended as a semi-formal, "bite size", logical (meaning implementation-neutral) statement of the logic of a process (or mini-system). Mini-Specs are typically produced only for the lowest level processes from the DFD's. DFD's for higher-level processes are, in effect, a specification of calling sequences. As noted above, Mini-Specifications are written in Structured English instead of narrative text. This is to avoid the ambiguity arising from complex, compound sentences with nested phrases, as well as other confusing syntax in narrative English. Structured English is a restricted subset of English in terms of both syntax and vocabulary. Problem Statement Language (PSL) is an example of a formally-specified or standardized Structured English. Mini-Specifications specify program logic independent of its implementation in any particular programming language and database manipulation language. Mini-Specifications in the Yourdon methodology are similar in many respects to what is referred to as pseudo-code. However, the language of pseudo-code looks more

like a programming language than Structured English, and pseudo-code typically includes specifications of flags and other low-level control structures, error handling routines, etc. Mini-Specifications do not normally deal with lower-level programming constructs such as error handling, flags and variable assignment, as these are at least somewhat dependent on the specific implementation using a particular programming language and operating environment. Some structured analysis and design methodologies other than Yourdon advocate the use of Data Flow Diagrams (DFD) and other front-end modeling techniques along with formal, pseudo-code languages such as the Vienna Development Method (VDM) [Fraser et al, 1991]. VDM emphasizes detailed specification of program logic including pre- and post-conditions, and typically includes lower-level constructs for condition checking, error handling, etc.

Decision trees are also used in Mini-Specifications to represent complex decisions. These are a graphics-based substitutes for IF-THEN statements, particularly useful when defining complex cases. They force the analyst or designer to clearly define both the branching condition and the possible paths. Assuming that the DFD's were modularized properly and taken to a low enough level of decomposition, a Mini-Spec should be no longer than between one-half page and one page. In many cases programmers should be able to code from Mini-Specifications, although for complex programs it is suggested that traditional programming specifications be written in a pseudo-code close to the target programming language. This allows specification of the particular program structure, taking into account the syntax and constructs of the target language and including variable assignment, low-level control structures and error handling routines.

### 3.3. Model Extension and Integration Projects and Approaches

#### GUIDE Data Modeling Extensions Project

GUIDE (actually GUIDE International Corporation), along with SHARE, is one of the two largest IBM users' groups. GUIDE has an on-going project to look at improvements or extensions to existing enterprise modeling methods and tools. The motivation for the project is partly, if not primarily, due to problems with legacy systems, both application programs and their associated databases. Many of the modeling requirements identified by GUIDE are necessary to support re-engineering. GUIDE [1991] states that much of the logic in a typical commercial application program enforces the rules and constraints for maintaining a database in a consistent state. Those rules and constraints are rarely expressed on a data model for the database, and only occasionally in specifications for programs maintaining the database. Because different people maintain the data model and the program specifications, those two system documents are seldom synchronized. Moreover, the code enforcing database rules is often not consistent between programs. The problem is that these rules are distributed among the programs that access the database. If the rules could be expressed in a central place on the data model, and were in computer-interpretable form (not just text), they could be checked for consistency. It would be better still if the rules were processable or transformable directly, or if one could generate COBOL maintenance code automatically. Such transformable rules could also create tests to verify the implementation [of the rules]. If the techniques of artificial intelligence and other related disciplines could be applied to data and system models, then the rules for database maintenance could be captured. And what better place to capture them than in a central repository. The key is to ensure that the repository can represent more powerful models.

Other motivations for the GUIDE project include: an increasing backlog of requests for maintenance and enhancements to existing application systems, as well as the development of new

## ITKM State-of-the-Art Review

applications systems; increasing system complexity, making it harder to debug and maintain application programs; the fact that many existing application systems simply don't work or don't work the way the users wanted them to; and the growing importance of information. Regarding the importance of information: capturing business rules and constraints in models in transformable, processable form could speed up system delivery and provide the "information edge" for a corporation.

According to GUIDE, modeling methods and/or languages should be: expressive; economical to use, for both creating and maintaining models; processable or transformable into programming code; verifiable; and well-suited to the communication of information. They should improve productivity, communications and quality. As an example of improving quality, a modeling language should automatically check assertions within an information system. The simplest checks could detect inconsistent pairs of assertions. A underlying knowledge representation system coupled with an inference engine could detect inconsistencies within whole sets of assertions, whereas a pair-wise comparison of system constraints would be difficult, impractical or impossible to perform.

The GUIDE project categorized current and emerging modeling technologies as shown below. This is more limited than the category scheme used in the ITKM industry survey, and appears to have a somewhat different focus.

- Data Models
  - Bachman Object-Role
  - Chen Entity-Relationship
  - Mathematical Models
  - Irreducible Models (Binary Relational)
  - Relational
    - Extended Model Relational (e.g. RM/T)
- Linguistic Models
  - Case Grammars
  - Shallow and Deep Meaning
  - Semantics
  - Shank's Conceptual Dependencies
  - Sowa's Conceptual Graphs
- Artificial Intelligence/Knowledge Representation
  - Logic
    - First and Higher-Order Logic
    - Non-Monotonic Logic
    - PROLOG
  - Semantic Nets
  - Procedural
    - Production Rule Systems
    - Pattern/Action
  - Frames
  - Rule-Based Expert Systems
- Programming Languages
  - Abstract Data Types
  - Object-Oriented Programming

- Text Processing
  - Tag and Markup Languages
  - Full Text Databases
  - Outline Processors
  - Hypertext

It should be noted that traditional text processing is an extremely limited form of "modeling" at best, since the real content or semantics of the "model" is hidden in the text and not explicitly or directly accessible. Tag and markup languages or full text databases with word search capabilities help users find particular words or groups of words, but they still do not deal with the representation of the meaning of the words. Outline processors may begin to provide very limited semantics, through the introduction of hierarchies (i.e. one word or concept is grouped under or is part of another word or concept). Hypertext provides somewhat more semantics than other text processing, in that keywords or concepts may be identified and linked or related to other words or text descriptions to provide some expression of meaning. Hypertext is described in more detail under Section 3.2., Modeling Methods and Tools, of this document.

#### Information Resource(s) Dictionary System (IRDS)

*[Note: includes IRDS I, IRDS II, SQL2, SQL3 and A Tools Integration Standard (ATIS)]*

The Information Resource(s) Dictionary System (IRDS) is a standards project being conducted by a committee of the American National Standards Institute (ANSI), specifically designated X3H4. X3H4 meets quarterly, with some working sessions conducted in the interim. There are approximately 35 organizations actively participating in X3H4. There is also an IRDS activity within the International Organization for Standardization (ISO), for which the ANSI X3H4 committee is the U.S. representative or "TAG" (for Technical Advisory Group).

Previous, shorter-range standards efforts (IRDS I) have dealt with the use of SQL (the standard Structured Query Language for relational database management systems) as a standard for limited integration among database management systems (DBMS's), particularly those based on the relational model. The IRDS I standard [American National Standards Institute (ANSI), 1988] is based on the ISO concept of "level-pairs" which allows for levels of abstraction for database representation (essentially the notion of pairing a class or schema with an instance). Proprietary DBMS's can still have their own internal implementations, but can interchange database files with other DBMS's using a [relational] SQL-based neutral exchange format — and without losing important syntax and semantics about the data structures being exchanged. A Tools Integration Standard (ATIS) [American National Standards Institute (ANSI), 1991a] is an extension to IRDS I that involves an object-oriented approach to the integration of DBMS tools, again using SQL. It has been described by some as an attempt to bring the advantages of an object-oriented paradigm to the existing SQL-based DBMS environment. The object-oriented modeling paradigm is described under a separate subheading within Section 3.2., Modeling Methods and Tools, of this document. ATIS provides a set of standard interfaces that support schema-driven dispatching of behavior. ATIS provides an interface to and defines a data model for an initial Information Resources Dictionary (IRD) or "repository". The ATIS interface and data model support the services interfaces defined under IRDS I and, to some degree, the expanded services interfaces emerging under IRDS II.

The concentration of ANSI X3H4 is now on longer-term efforts (IRDS II) involving advanced representation technologies. There is some divergence of opinion on this subject between the ANSI IRDS committee working on IRDS II and the international IRDS community within ISO. Specifically, some of the ISO representatives are currently quite comfortable with SQL and believe that SQL can be easily extended to meet the requirements of IRDS II. The ANSI X3H4 IRDS committee does not hold the position that SQL need necessarily serve as the [only] basis for IRDS II, and instead is advocating an approach using advanced representation schemes with powerful primitives based on symbolic logic and knowledge representation. ANSI IRDS holds that there may be several representation languages sufficient to address IRDS II requirements, and that any future logic-based version of SQL might be one of those languages. In all fairness, ISO only recently started its IRDS II effort and has not had time to fully evaluate the ANSI IRDS II work and formulate a formal ISO position. ISO may well broaden its initial thinking once it has had sufficient time to consider the ANSI IRDS II approach and other ideas evolving internationally with respect to three-schema, dynamic data dictionary technologies.

As additional background material, it should be noted that SQL2 was just recently adopted as a new international standard. SQL2 is an extended version of SQL that incorporates some aspects of behavioral modeling into the basic SQL database language. Implementation of the SQL2 standard will not be an easy task. The standard is specified in a voluminous set of documents numbering several hundred pages. The specification is written largely in narrative English, with limited use of a formal language for expressing SQL2 syntax and/or semantics. There are currently no prototype implementations of SQL2. Some of those familiar with the standard have stated that it will be difficult to teach people how to use it. Some people are suggesting a moratorium of up to five years on implementing the SQL2 standard to allow more time for practical work and prototyping — which they believe will lead to useful revisions and/or extensions to the standard. Since SQL2 certainly does not meet all the IRDS II requirements, some people are already anxiously awaiting SQL3, which is attempting to incorporate object-orientation into the basic SQL database language. Many other people, however, have concerns about the inherent limitations of the relational form (on which SQL is fundamentally based) in terms of its ability to “re-represent” other representational forms such as directed graphs (i.e. hierarchical databases) and non-directed graphs (i.e. Codasyl or network databases) [Dement and Woodruff, 1988] — not to mention advanced knowledge representation schemes that are capable of representing the meaning of information. This last category includes semantic networks and other knowledge representation schemes capable of describing complex concepts such as states of affairs, modes (e.g. time, belief states, probability, etc.) and contexts (i.e. complex collections or groupings of modalized states of affairs).

The objective of the IRDS II project [Note: all references to IRDS from this point forward apply to IRDS II unless otherwise noted] now being conducted under X3H4 is to define requirements, architectures and standards for a three-schema-based [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a] information repository which will represent and manage the information resources of an enterprise. Repositories may also be referred to as data dictionaries, and in the more advanced sense as common semantic models or federated semantic database management systems. Repositories, data dictionaries, etc. are discussed in more detail in Section 3.4., Repository/Data Dictionary Technologies, of this document.

There is a specific IRDS Task Group (X3H4.6) concerned with defining the requirements, architecture and standards for the Conceptual Schema (CS) of the IRDS. A major thrust area of this task group is to define and develop a meta conceptual schema (the Conceptual Schema of the IRDS itself) capable of representing all other conceptual schemas. Since X3H4 equates conceptual

schemas with "models" (and some .6 task group members, Ontek included, further stipulate that models can be equated with systems of representation themselves), an objective of the .6 task group is to define a Conceptual Schema capable of enabling enterprise model integration. In general, the IRDS CS architecture could serve as the basis for integration efforts, specifically "unification", within any group of enterprise modeling tools. For example, if the IDEF vendors agree on a neutral model of the syntax and semantics of IDEF0, IDEF1/1X, etc. (as is the objective of one of the subgroups of the IDEF Users' Group), then that will not only allow integration among the IDEF tools, but — using an Integration Toolkit and Methods (ITKM) based on IRDS standards — it will enable integration between IDEF and other enterprise modeling tools such as NIAM, Conceptual Graphs and dynamic process models, as well as database languages such SQL and programming languages such as COBOL. This will, of course, require the representation of the IDEF "neutral model" as a standard content module at the IRDS CS Modeling Schema layer, using the IRDS Normative Schema constructs, as expressed in an IRDS Normative Language. For a detailed discussion of the layers of the IRDS Conceptual Schema, the reader is referred to [Burkhart et al, 1991] or Section 3.3.2., Requirements Specification For Architectural Elements or Components Of an ITKM, in an companion document entitled "Needs Analysis and Requirements Specification For an Integration Toolkit and Methods (ITKM).

The IRDS effort, in particular the work of the .6 IRDS Conceptual Schema Task Group, is being harmonized with the Product Data Exchange using STEP (PDES)/STandard for the Exchange of Product data (STEP) effort and the DARPA Knowledge Sharing Effort (KSE), as well as other related industry efforts described in this section. The U.S.-led Initial Graphics Exchange Standard (IGES)/PDES Organization (IPO) is being utilized as one source for overall IRDS user requirements. The IPO Dictionary/Methodology Committee (DMC) is coordinating with X3H4.6 in specific regarding requirements for Conceptual Schemas and possible contents for a meta CS. The work being conducted by DMC is referred to as the Semantic Unification Meta Model (SUMM) and is described briefly under its own subheading within this section. ANSI X3H4 IRDS has not formally reviewed the SUMM, since there is no intent to use the SUMM as a direct basis for the IRDS CS.

The IRDS Conceptual Schema activity is rapidly being recognized by industry as the leading standards work in general enterprise modeling and model integration. ANSI has officially sanctioned the IRDS CS work and recognizes the effort as one of two official U.S. standards efforts involving Conceptual Schemas [Note: the other effort, under ANSI X3T2, is targeted at CS to support communications or Electronic Data Interchange (EDI) rather than the IRDS context of CS definition, which targets CS languages for enterprise model integration and database integration. X3H4 and X3T2 harmonize their work and are part of the same joint technical committee under SPARC/SC21]. There are plans to restart ISO standards-related efforts in the area of Common Data Modeling and Conceptual Schema Facilities. These efforts would draw heavily on the work being done by the ANSI X3H4.6 Task Group. The first organizational meeting leading to new ISO efforts was held in March 1992 in Renesse, The Netherlands. Attendees included Joost van Griethuysen, Martin King, Dr Eckhard Falkenberg, Dr. Robert Meersman and others who were involved in the original ISO three-schema database architecture work [International Organization for Standardization (ISO), 1987a], as well as Japanese, German, Canadian and other experts working in the field of conceptual schemas.

Neutral Information Representation Scheme (NIRS)

*[Note: includes Integrated Development Support Environment (IDSE)]*

This work is being performed by Knowledge Based Systems, Inc. (KBSI) under the Integrated Information for Concurrent Engineering (IICE) contract sponsored by the Air Force Armstrong Laboratory. KBSI states that they are using "mathematical formalism to define the underpinnings of modeling to accomplish subsumptive model integration". This includes performing graph and set theoretic validation of IDEF models based on semantics attached to particular graphical constructs. They separate this into three categories: graphical syntax, isomorphic logical or linear syntax, and semantics for logical syntax. KBSI believes that IDEF1/1X, and probably IDEF0, can be represented using first-order [predicate] logic (using what KBSI refers to as "non-temporal objects"). Representing the other [proposed] IDEF methods such as IDEF3 and IDEF4 would require higher-order logic (using what KBSI calls "temporal objects", which can be represented either by introducing new relations and constraints, or directly through the introduction of modal logic or modes). The Integrated Information Systems Constraint Language (IISyCL) is being defined by KBSI as a formal language for stating class invariant constraints. IISyCL is described in more detail under the subheading "IDEF4 Object-Oriented Design" within Section 3.2., Modeling Methods and Tools, of this document.

The KBSI/IICE approach involves developing a "platform" called the Integrated Development Support Environment (IDSE). It has four levels that are somewhat analogous to the four levels of Product Data Exchange using STEP (PDES). IDSE includes functionality for "configuration management of artifacts". The IDSE architecture is based on an "integration services strategy like the Engineering Information System (EIS) and Computer-Aided Software Engineering (CASE) Data Interchange (CDI)". This approach is described as being different from either a three schema-based subsumptive model unification approach using a common semantic model (as advocated by the American National Standards Institute (ANSI) X3H4 Information Resources Dictionary System or IRDS II project described under a separate subheading within this section) or a network-based federated systems approach using pair-wise translation (as advocated by the Microelectronics and Computer Technology Corporation (MCC) and other members of the Enterprise Integration Program (EIP) Suppliers' Group, as described briefly at the end of this section). The IDSE architecture does still have some elements in common with, or similar to, those of three schema-based unification and/or network-based federation approaches. KBSI stated plans to demonstrate Level 0 of the IDSE in late 1991. They recently submitted design specifications to their Air Force program office for Levels 1-3, but those specifications have not yet been approved for distribution to the general public.

The underlying representation for the IDSE is called the Neutral Information Representation Scheme (NIRS). NIRS was described as being like a "periodic table for modeling elements. All modeling elements must either be primitive in NIRS or derived from the elements that are primitive". In this respect, NIRS follows the subsumption-based unification approach. KBSI states that IDSE, including NIRS, is a better project than either the CYC project at MCC or the Tacitus project at the Stanford Research Institute (SRI). In KBSI's opinion, CYC is text-oriented and is too wide and shallow [Note: CYC is intended to be a commonsense knowledgebase [Lenat et al, 1986]. CYC might support model integration by providing some general domain semantics that might be useful to aid in integration. However, CYC was never intended as a model integration project per se]. KBSI describes Tacitus as too narrow and deep, and not natural language-oriented.

KBSI is also working on automated methods for model interpretation based on the use of domain ontologies and commonsense knowledgebases (such as CYC). An example was given wherein "the system knows that when you model casting on the shop floor, you are talking about the physical object or artifact not the symbol or the data associated with it". These automated methods would automate the "ontology of a method for building other ontologies". This would be used to rapidly create concept libraries or encyclopedic knowledgebases. This aspect of the project was noted as being geared toward knowledge engineers who are trying to create domain ontologies, not toward end-users. In many of the above respects, the KBSI IICE work appears to be consistent with the results of Ontek's AAAP work and the proposed ITKM approach. However, KBSI's effort does not directly encompass legacy database integration at this time.

KBSI's analysis method involves defining basic relations about things including: defining properties, essential properties and accidental properties. There are also other basic relations such as: Part-Whole (Car, Tire), Kind (e.g. Mammal, Dog), Generalization-Specialization (e.g. Fastener, Machine-Threaded Fastener), Description-Subsumption (e.g. Square, Rectangle), and the generic relation type for defining all other relations. Kind is not equal to set "as set is extensional and kind is intensional, with an *s*." They start with an empty or nearly empty "proto-kind" and then overlay or define its properties over time.

KBSI uses *situation theory* for dynamic process modeling. They can "treat an event as an object, and also provide access to what is in the event". This is done through the use of "infons" which are structured semantic objects in situation theory. Infons were defined by Devlon in his book on situation theory from Cambridge Press (an explicit reference is not available at this time). Infons are like *states of affairs*. This is fairly consistent with the work that Dr. Terry Parsons has done at the University of California at Irvine (UCI) in situation theory. There is an "infolgebra" for manipulating infons. Infons have polarity (i.e. they are either true or false). One can construct complex infons out of other infons. KBSI has a syntax for representing infons and infon construction. For example, the sentence "Dick saw Rosemary erasing the tape" would be expressed something like (Erasing, Rosemary, Tape, t) and that would be pointed to by another event (Dick, Seeing, (Erasing.....)). This is, of course, similar to Ontek's "hat syntax".

It may be useful to note that IBM is also doing privately-funded research and development work in the area of situation theory. They have had trouble with what they call "relationals". These are basically any second-order relations, i.e. any relation that points or refers to another relation. In the previous example, the "tape" that Dick saw Rosemary erasing is problematic because much of semantics of the tape may have been introduced elsewhere in the model (in relations more specifically about the tape itself), and synonyms or contextual references may have been used in those relations. This muddies the representational water even more and poses problems for model integration. Specifically, it may be difficult to determine all the relevant aspects of one model that may be needed to provide semantics in conjunction with another model (with which the first model is to be integrated). Situation theory inherently deals with these kinds of representational issues. In Ontek terminology, these are represented by modalizing (or adding a mode) to an event — they are simply implemented as one relation modalizing or qualifying another relation, which in this case happens to represent an event. These modalizations may be built up or layered, and when considered collectively they provide the semantics of the overall context or situation.

#### Portable Common Tool Environment (PCTE)

The Portable Common Tool Environment (PCTE) project [Davis, 1991] is being conducted by the European Computer Manufacturer's Association (ECMA), specifically the



## ITKM State-of-the-Art Review

PCTE+ Specification Group. ECMA is a part of the European Community (EC), and has no affiliation with the International Organization for Standardization (ISO). PCTE is intended to support the development of an integrated project support environment. Rather than to build such an environment, including various tools and methods, from scratch, the PCTE approach is to provide a standard public tool interface to which existing and future vendor-supplied tools and associated methods would be encouraged to conform. The "tools" addressed by such an environment (and, therefore, by the PCTE interface) include any and all tools which "amplify the creativity and productive effort of people involved in software development and maintenance". PCTE design aims include:

- provide functions common to software engineering tools and a basis for their integration;
- allow portability of tools across hardware and operating systems;
- provide general mechanisms, not policies (i.e. the intent is to provide something more concrete than the broad policy guidelines specified in many standards);
- be implementable on a local area network of engineering workstations and servers;
- provide security mechanisms suitable for both civilian and military use;
- be complete.

Specific objectives include development of a Framework Reference Model and specification of Reference Model Services. Initial versions have already been developed and were expected to be approved by the PCTE+ Specification Group in late 1991. The core service of the reference model concerns itself with Object Management. Objects, links and attributes would be specified in Entity-Attribute-Relationship (E-A-R) to populate the underlying schema of the PCTE. Other services include: process management (in the operating system sense); transparent distribution management; security and access control; data repository services; messaging; data integration; task management (including event monitoring); and numerous user interface services. PCTE would have its own user-oriented access language, but would be compatible (at some level) with SQL (most likely based on SQL3).

The PCTE+ Specification Group views PCTE as having its own niche in the overall integration architecture. PCTE would be built on and utilize standard operating system technology (e.g. POSIX) and database management system technology (e.g. SQL) as its platform. PCTE would provide standard tool services through its public tool interface. Environment mechanisms would also be found at this level. At the next level up (the "framework level"), PCIS (described briefly under the last subheading at the end of this section), would set operating environment policy and provide higher-level integration services. Above that would be the particular tools which are found in the overall software engineering environment. These might include CASE tools, programming language tools, modeling tools, database management systems, object management systems, application tools (such as project engineering applications), etc. These tools would continue to be vendor-proprietary. Depending on how unique the tools were and how closely they conformed to standards, PCTE would enable tools to: interchange data; interoperate; or be fully integratable.

Most of the member of the PCTE+ Specification Group are European-based companies, although several of the companies are European branches of major U.S. hardware and software vendors (including IBM, HP, DEC and Sun Microsystems). The group contrasts itself with ANSI X3H4 IRDS (described under a separate subheading within this section) by stating that both groups "have the same ambition, but come from different cultures". IRDS is described as coming from a database management culture, whereas PCTE comes from a software engineering culture. In Davis' words, the IRDS market and culture have a different perception [than the PCTE market and culture] of software developers as being "programmers - at the bottom of the pecking order" (in the case of IRDS) versus "software engineers - of rather higher status" (in the case of PCTE). These cultural difference result in differences in emphasis, approach and architecture. According to Davis, IRDS addresses commercial CASE tools and modeling tools, has an emphasis on traditional DBMS's (including integration of them) and assumes an implementation environment involving PC's and mainframes. PCTE addresses technical CASE tools and other software development tools (including ones closely related to technical programming languages such as Ada and C++), has an emphasis on operating systems and object-oriented databases, and assumes engineering workstations as its implementation environment. Davis also implies that PCTE's underlying definition may be more mathematically formal than IRDS because PCTE is using an "abstract, semi-formal programming language as its specification language". The implication is that IRDS uses "diagrams" which are good starting points, but not sufficient. As noted in the discussion on IRDS, the primitives in the Defining Schema and Normative Schema of IRDS will be based on logic (both first order [predicate] logic and higher-order logic), but will also draw upon other disciplines such as philosophy, ontology, computer science, cognitive science, natural languages, etc. The Defining Schema and Normative Schema will be formally defined. The IRDS Conceptual Schema Task Group has adopted Conceptual Graphs as a language for expressing the syntax and semantics of the IRDS Conceptual Schema (including the Defining Schema and Normative Schema). Conceptual Graphs use a combination of textual and graphical forms to express formal logic in a more user-friendly form than first order [predicate] logic. Conceptual Graphs as a modeling language is described in more detail under its own subheading within Section 3.2., Modeling Methods and Tools, of this document).

### Semantic Unification Meta-Model (SUMM)

The Semantic Unification Meta-Model (SUMM) work is being conducted by the Dictionary Methodology Committee (DMC) of the IGES/PDES Organization (IPO) [Note: IGES stands for Initial Graphics Exchange Specification (IGES); PDES stands for Product Data Exchange using STEP; STEP stands for STandard for the Exchange of Product data, and is a project of the International Organization for Standardization (known by the initials ISO)].

The original objective of the SUMM effort [IGES/PDES Organization (IPO) Dictionary Methodology Committee (DMC), 1991; Tyler, 1990; Fulton, 1990, 1991a and 1991b] was to define a meta-model sufficient to serve as the underlying basis for translating among various information modeling methods, all of which could potentially be used for modeling within the PDES application domain. However, in attempting to solve the specific problems within the PDES application domain, IPO DMC came to the conclusion that any meta-model capable of solving the PDES-specific problems would also be applicable to other domains as well. This is because the real problem being dealt with is, in fact, not specific to the PDES application, but rather relates to translation among information modeling languages in general (information modeling is described under the subheading "Data Modeling Versus Information Modeling",

## UUKM State-of-the-Art Review

within Section 3.2., Modeling Methods and Tools, of this document). The SUMM model itself uses rather traditional predicate logic to represent the primitive constructs which IPO DMC believes would be necessary to translate among (at least major) information modeling languages such as NIAM, EXPRESS and IDEF1X. There is no intent to produce a neutral or normative language as part of SUMM (although some may consider basic predicate logic as serving in that role in the draft version of the SUMM model [IPO, 1991]). In fact, the draft version of the SUMM model uses a combination of NIAM, IDEF1X and E-A-R graphical syntax to depict some aspects of the meta-model. However, it is stated that these notational conventions are not part of the SUMM. IPO DMC is planning to complete the SUMM work by late 1992.

There are two major concerns about the SUMM effort. First, the breath and depth of the IGES/PDES efforts already mean that IPO has a huge workload, and it should not divert important resources away from IGES/PDES-specific tasks in order to develop a general meta model on its own, particularly when there are other symbolic logic-based efforts with which SUMM could be closely coordinated or into which the SUMM effort could be subsumed.

Second, the draft SUMM model is based on rather traditional, textbook predicate logic, with some [limited] extensions for set theory and higher-order logic. Many groups, including ANSI IRDS various efforts under the DARPA Knowledge Sharing Effort (KSE) and the IPO DMC group members themselves, have concluded that first-order [predicate] logic (or FOL) alone is not sufficient for representing the meta model for model integration. Many of the primitive constructs necessary to represent the syntax and semantics of modeling (i.e. to define a normative or neutral language in which [all] other modeling languages can be represented) will be at least second order, and most likely even higher-order. Some of the constructs necessary to represent modes, contexts or situations and other complex semantics may be difficult, if not impossible, to represent using only extended FOL. Some semantics may only be expressible in derived (i.e. less primitive) constructs, although those derived primitives should be traceable to the lower-level primitives out of which they arise. If the SUMM has no neutral or normative language, then it will be difficult to define normative constructs derived from low-level logical or "defining" primitives, as well as organize and account for (including providing traceability for) the entire process of defining and deriving all the constructs that are necessary for model integration or unification. The problem which the SUMM effort may have sought to avoid is how to use any other form besides pure, traditional predicate logic and still ensure neutrality. In other words the question is: how does one ensure the form chosen is, in fact, a neutral form for representing the primitives and does not arbitrarily restrict or influence the meaning of the primitives through its usage as a defining and/or normative language? This is a significant challenge, but it is not impossible. The most important thing is to ensure that the emphasis remains on the primitives being defined and that the form used is *conducive*, not *intrusive*, to the process.

### DARPA Knowledge Sharing Effort (KSE)

*[Note: includes the Knowledge Interchange Format (KIF) project, the Ontolingua language, the Interlingua working group, the Knowledge Representation System Specification (KRSS) project, and the Shared Reusable Knowledgebase (SRKB) project]*

The Defense Advanced Research Projects Agency (DARPA) Knowledge Sharing Effort (KSE) is an umbrella project comprised of several individual projects. The projects are logistically independent — each having its own contractors and unique set of objectives — but all

projects relate in general to communication and integration among knowledgebases. The projects are coordinated with one another, as appropriate, and some projects use the outputs or deliverables of other projects. Currently, there are four major projects in work:

- Knowledge Interchange Format (KIF) — R&D relating to a mechanism for exchanging or *interchanging* knowledge residing in multiple knowledgebases and described using multiple knowledge representation languages. KIF is therefore sometimes referred to as an "interlingua".
- Knowledge Representation System Specification (KRSS) — R&D relating to an underlying platform for integration among knowledge representation languages within the KL/One family of languages.
- Shared Reusable Knowledgebases (SRKB) — R&D relating to approaches for avoiding "mismatch" problems between multi-site (i.e. distributed) knowledgebases built based on multiple, overlapping, and potentially *conflicting* ontologies.
- Knowledge Query and Manipulation Language (KQML) — R&D relating to physical communications among knowledgebases. This project has a shorter-term focus than the other three. It is intended to provide a communications protocol that would likely be used as a low-level foundation for technologies specified and/or developed under the other projects.

KIF, KRSS and SRKB are described in more detail in the following paragraphs.

As noted above, one of the projects under the DARPA Knowledge Sharing Effort (KSE) is the **Knowledgebase Interchange Format (KIF)** work being conducted by the Stanford University Knowledge Systems Laboratory. KIF is described by its developers as an "interlingua" — i.e. it is a language for sharing knowledge between different knowledge representation languages using translation. It is also described as a language for importing, storing and exporting "concept libraries" or general domain ontologies. KIF is based on first order predicate calculus (i.e. first order logic or FOL, for short) and uses a LISP-like syntax for expressing its declarative semantics. By "LISP-like" it is meant that KIF is formal, expressive and logically comprehensive like LISP, but at the same time it is intended to be more human-readable than standard LISP. According to KIF's developers, human-readability is critical for publishing example knowledgebases and for assisting human interactive translation of knowledge between disparate knowledge representation languages. KIF itself is designed to make the epistemological-level content of a knowledgebase clear to the reader, but not to support automated reasoning. An extension to KIF, referred to as Ontolingua, supports limited [deductive] reasoning using term subsumption and both monotonic and non-monotonic inference rules (specified by the user). Ontolingua also allows for portability of ontologies in a form that is compatible with multiple [knowledge] representation languages [Gruber, 1991].

As stated above, KIF (from here on, KIF is assumed to include its extension, Ontolingua) is intended for communication between differing knowledge representation schemes, languages and/or systems. However, for this communication to work, the semantics of the knowledge being shared must be represented fairly similarly. That is to say, the knowledge representation scheme, language and/or system on either end must employ fairly similar semantics to represent the

knowledge being shared. For this reason KIF may be considered, as its name states, as an *interchange format* — rather than a knowledge representation integration language in the subsumptive sense. Efficiency is not a paramount concern for KIF, as KIF is intended primarily for use during the building of [large-scale] knowledgebases, not for after-the-fact integration of knowledgebases built independently. Also, there is no procedural aspect to the KIF language — it is strictly declarative.

The specific KIF work under the DARPA Knowledge Sharing Effort can actually be considered part of a larger working group referred to as Interlingua. The Interlingua working group is concerned in general with translation between differing [knowledge] representation languages, in this case using KIF as an intermediary language. For reasons noted above, however, KIF is not universally-accepted by all participants in Interlingua as sufficient for knowledge sharing. Specifically, several Interlingua participants reject the notion that any knowledge *interchange format* can be devised that accurately captures the semantics of knowledge representation schemes, languages and/or systems capable of representing very disparate kinds of knowledgebases. They believe that much of the knowledge sharing that needs to take place will be among very disparate systems, and that to share knowledge among them will require a deeper, common representation system. That common representation, in turn, must be capable of representing the syntax and semantics of all the knowledge representation schemas, languages and/or systems involved in the sharing. This approach would enable subsumptive integration among potentially very disparate knowledge representation schemes, languages and/or systems while at the same time also allowing for sharing among similar ones.

A second DARPA Knowledge Sharing Effort (KSE) project known as **Knowledge Representation System Specification (KRSS)** is pursuing a subsumptive approach — but only in a fairly restricted sense. KRSS is concerned with specifying a common representation platform for a limited spectrum of knowledge representation languages. This spectrum currently includes terminologic languages with a heritage in KL/One. KL/One is a knowledge representation language based on a semantic network [Note: semantic networks are described under a separate subheading within Section 3.2., Modeling Methods and Tools, within this document]. As with any semantic network, KL/One (or more precisely, the KL/One *family of languages* in general) represents nodes (for concepts) and links (for roles or relationships between concepts). The underlying concepts or constructs of KL/One are defined using a [relatively] small set of primitives with well-defined semantics expressible in formal logic. This makes KL/One-based knowledge representation languages a good choice for subsumptive integration. Also, the KRSS contractor chose to stay within the KL/One paradigm rather than attempt an “end-all, be-all knowledge representation language” because they simply felt it was premature to undertake the overall effort at this time.

The University of Southern California (USC) Information Sciences Institute (ISI) is the lead contractor on the KRSS effort, and is working closely with AT&T. To accomplish its objectives, USC ISI believes that a fully specified knowledge representation system will make the best vehicle for sharing. In other words, the KRSS effort intends to develop an implementable language specification. Unlike KIF, a KRSS-based approach would result in the development of a common, underlying knowledge representation system rather than an interchange mechanism between knowledge representation systems. Because of this distinction, KIF entirely omits operational (i.e. dynamic or procedural) semantics, since there is no need for such semantics in a knowledge exchange or interchange-only medium. However, the downside for KRSS is that the specification of the semantics for subsuming all of the syntax and semantics (both static and dynamic) of all KL/One-based knowledge representation languages, schemes, systems, etc. is

distinctly non-trivial. This problem becomes even worse for any broader scale subsumptive-based integration approaches where subsumption of all other [knowledge] representations, languages, schemes, etc. is the objective. To mitigate this issue, KRSS takes the approach that no "useful" knowledge representation language can be completely specified. They separate the re-representation aspects of KRSS into two categories or layers: an inner layer, which must be complete; and an outer layer, which will [always] remain incomplete. Inference rules are used at the outer layer to assist users (i.e. knowledge engineers or systems analysts) with translation or integration of particularly complex semantics. These inference rules are implemented as part of a set of automated "reasoners" which can be thought of as an abstract classification algorithm or a "classifier". One problem with this solution is deciding where to draw the line between the inner and outer layers. Is it always in the same place (is it rigid)? If not, how do you establish where it is in any given case (using the inference rules)? A second problem is that — as with KIF — the classifier is intended to be used primarily during the building of new knowledgebases. Knowledgebases that are built independently, but which later need to be integrated, are not directly addressed. Additionally, during the KB building process the classifier is supposed to help prevent "conceptual drift", i.e. changes in the definitions of concepts being defined at multiple times, by multiple team members and/or within multiple knowledgebases. A problem arises here in that some differences in definitions may be perfectly valid given different domain viewpoints or other contexts, and, in fact, it may be extremely useful to identify where, when and why the conceptual drifting is occurring. This creates a requirement not to *prevent* conceptual drifting, but rather to identify cases of conceptual drifting as they occur and to capture additional semantics behind the conceptual drifting process.

Finally, because a KRSS system would itself be a knowledge representation system, it will need its own facilities, such as a query language, configuration or versioning control, etc. for knowledgebase management. These will have to be specified as part of the KRSS effort as well, but have not been addressed to-date.

A third project under the DARPA Knowledge Sharing Effort (KSE), and the last one to be discussed here, is the **Shared Reusable Knowledge Base (SRKB)** project. Although both KIF and KRSS touch on the issue of multiple, overlapping ontologies, SRKB is focused more specifically on the subject. SRKB assumes an environment where multiple, distributed project teams are building a [relatively] large shared knowledgebase. The SRKB approach is based on a client-server model where the addition or revision of concepts in a shared concept library or ontology library is handled by "negotiations" and "contracts" between an "owner" of the concept and a "client" who wants to use or modify the concept. This appears to be a somewhat federated approach to knowledgebase development. SRKB relies on KQML and KIF for physical communications and interchange between knowledgebases.

Like the KRSS project, SRKB is being led by the University of Southern California (USC) Information Sciences Institute (ISI), although there is a separate project team for each of the two projects. Lockheed and Stanford University are also involved. There are a number of experiments being conducted to drive the SRKB requirements and to prototype technologies. These include: a concurrent engineering-oriented knowledgebase environment called PACT; a knowledgebase for mechanical engineering which supports the design of tools for circuit board production; and several other product definition-related experiments. SRKB is also being closely coordinated with another project at USC ISI called System Helping to Evolve Large, Team-accessed Explicit Representations (SHELTER). SHELTER involves a re-use oriented interaction paradigm for teams developing related ontologies.

## ITKM State-of-the-Art Review

Like KIF and KRSS, SRKB deals primarily with the building of new knowledgebases, and provides little or no support for integrating or translating between independently-developed knowledgebases already in existence. Project representatives claim that all three projects — KIF, KRSS and SRKB — would be of benefit for the development of large-scale knowledgebases, including commonsense knowledgebases such as CYC [Lenat et al, 1986]. CYC is experiencing significant problems with ensuring semantic consistency across a large-scale knowledgebase. To deal with this problem, the notion of “micro-theories” has been introduced into CYC. Micro-theories are simply subdivisions or segments out of the overall knowledgebase in which semantic consistency is both necessary and [reasonably] possible to achieve. It remains to be seen whether a large-scale knowledgebase truly lends itself to [relatively] simple logical division in this manner. There may be a need for hierarchies or even networks of micro-theories. The issue here, of course, would be how to ensure semantic consistency among related micro-theories (i.e. meta micro-theories or aggregations of lower-level micro-theories). The DARPA Knowledge Sharing Effort intends to coordinate with the CYC project on this issues. From the discussion above, it seems clear that KIF, KRSS and SRKB may be of use during the building of new micro-theories, but — with the possible limited exception of KRSS — are of little use for solving problems with the existing, legacy CYC knowledgebase. It is interesting to note that the chief scientist of KIF (who was also one of the original developers of the Knowledge Engineering Environment or KEE frame-based expert systems shell) stated that reusability of concept libraries and the ability to deal with multiple, overlapping ontologies require explicit representation of the purpose, view, assumptions, etc. associated with particular concept libraries or standard domain knowledgebases. This, in turn, requires the ability of [underlying] knowledge representation languages or schemes to represent modes (including intentional modes), contexts or situations, and other higher-order semantics associated with those concept libraries or knowledgebases.

## Various Other Model Integration Projects

Other efforts are also addressing the requirement for enterprise model integration, or at least portions of the requirement. These efforts include:

- Computer Aided Software Engineering (CASE) tool integration, including at least three projects directed at integration among CASE tools, but addressing many issues common to model integration in general (or at least *information model* integration). The projects are: a new committee formed under the American National Standards Institute (ANSI) X3 - Information Processing Systems, Standards Planning and Requirements Committee (SPARC), known as X3H6 CASE Tool Integration Models (CTIM) and chartered with developing standards for CASE tool integration [Note: the name originally used by this group during its formation was CASE Integration Services (CIS)]; the Electronics Industry Association (EIA) CASE Data Interchange Format (CDIF) project also under ANSI (but not under X3/SPARC); and DARPA's Software Technology for Adaptable Reliable Systems (STARS). Each is described in more detail below. A fourth project known as Standard Data Access Interchange (SDAI) is apparently also addressing at least some aspects of CASE tool environment standards, but no detailed information on that project was available as of this writing. SDAI is part of the Product Data Exchange using STEP (PDES)/STandard for the Exchange of Product data (STEP) project of the International Organization for Standardization.

The ANSI X3H4 Information Resources Dictionary System (IRDS) committee recommended to ANSI that no separate X3/SPARC project committee be formed to deal with CASE tool integration. ANSI X3H4 stated that the CDIF and STARS efforts (as noted above and described in more detail below), coupled with the general model integration work being performed by the ANSI X3H4.6 IRDS Conceptual Schema Task Group and the proposed new X3H4 work effort on a CASE tool standard information model content module, was already sufficient to address CASE tool integration requirements. ANSI, however recently voted to form the new committee, stating that it felt the proposed work was closely related to, but sufficiently different from existing efforts, to merit a separate X3/SPARC project committee. There were a number of ANSI delegates who dissented. The new X3H6 CASE Tool Integration Models (CTIM) committee will concentrate on data control and integration (or at least, *interchange*) among CASE tools (i.e. within a common CASE tool environment). This will be achieved through a common framework (such as PCTE, described above), tool-level standards, and other means for easier tool-to-tool communication. Standards for processing modeling are also part of this effort.

The other ANSI group, EIA/CDIF, is already working on three major standards along these same lines: a [CASE-based] framework for modeling and extensibility; a transfer format definition for physical data transfer among CASE-tools; and a standardized CASE interchange meta-model, which would provide the underlying semantic basis for the interchange of model data among CASE tools. CDIF modeling efforts to-date have concentrated on Data Flow Diagrams (DFD's), although other aspects of processing modeling, particularly Finite State Machines (FSM's), are also be addressed [Note: Data Flow Diagrams and Finite State Machines are described, respectively, under Section 3.2., Modeling Methods and Tools, subsection on Process Dynamics/Simulation Modeling and subsection on the Yourdon Methodology]. Some prototype development efforts are also being performed as part of this project.

STARS is concerned with developing a broad software engineering environment (SEE), particularly directed at Ada-based software engineering for large-scale embedded systems. CASE tool integration, specifically including integration among existing state-of-the-art CASE tools, is only one aspect of the overall STARS work, but is an area of considerable short- to mid-term focus. However, in the long-run "CASE" tools in the STARS SEE may look substantially different than today's CASE tools. They will likely involve integration among Ada programming language generators, traditional databases, object-oriented databases, project management application systems, and various user interfaces, including modelers and other graphics-based facilities for capturing semantics via models, simulators and programming specifications [Gardner, 1991].

- ❑ Computer-based Patient Records (CPR) project, based on a Unified Medical Language System (UMLS) being developed at the National Library of Medicine (NLM). Several prototype artificial intelligence systems have been developed in recent years under R&D projects related to the medical field;
- ❑ Enterprise Integration Program (EIP) under the Air Force Wright Laboratory - Manufacturing Technology (ManTech) Directorate. This aspect of the EIP work is being



## ITKM State-of-the-Art Review

performed by the Microelectronics and Computer Technology Corporation (MCC) and an EIP Supplier Group including IBM, DEC and AT&T. The approach originally being advocated under this effort can be described as a network-based "federated systems approach" using the notion of pair-wise translation between enterprise modeling methodologies and tools [Microelectronics and Computer Technology Corporation (MCC), 1991]. Several workshops conducted among representatives from both the technology supplier community and industry have resulted in a shift towards a unification-based approach to enterprise integration, consistent with the work of ANSI X3H4 IRDS (as described above);

- Integrated Software Engineering Environments project at the National Institute of Standards and Technology (NIST);
- Intelligent Information Systems (IIS) project under the National Science Foundation;
- LILOG project led by IBM in Stuttgart, Germany and involving several universities in Germany. LILOG is based on sorted logic, which lends itself to the use of Conceptual Graphs as a "front-end" tool for capturing and presenting the semantics represented in the LILOG language;
- Object-oriented efforts such as those being conducted by the Object Management Group (OMG), as well as the ANSI Object-Oriented Database Task Group (OODBTG). OODBTG is working on an Object Data Management Reference Model and recommendations for standards in Object Information Management [American National Standards Institute (ANSI), 1991b]. Like the CASE tool integration work mentioned above, this work is specific to object-oriented tools and methods, but has some applicability to the general integration problem;
- Portable Common Interface Set (PCIS) project sponsored jointly by the Ada Joint Program Office (AJPO) and the Independent European Programme Group of the North Atlantic Treaty Organization (NATO)'s Ministry of Defense (MoD) [Microelectronics and Computer Technology Corporation (MCC), 1991]. The goal of this project is to support the creation of the next generation of Ada Programming Support Environment (APSE) technologies. PCIS would incorporate the PCTE standard (described under a separate subheading within this section), as well as other standards related to CASE tools and software development tools in general. The intent is to promote an international widely-used standard for software [development] environments;
- Integrated CASE or I-CASE, a proposed project sponsored by the Air Force Gunter Laboratory. I-CASE is currently in the Request For Information (RFI) phase of procurement. It is quite similar to STARS and PCIS (described above) in its overall objective; namely, to establish a standard software engineering environment (SEE) that supports a formal, repeatable software development process throughout the entire software development life-cycle. I-CASE will generate applications using Ada and relational database management systems (RDBMS) technology targeted for open systems environments. I-CASE will also provide an environment to migrate current MIS applications to an Ada/RDBMS implementation. The I-CASE architecture actually

specifies three distinct operating environments: the Software Engineering Environment (SEE), which facilitates the building and/or modification of applications using CASE tools, including an Ada code generator; the Operational Test Environment (OTE), which gets the code from the SEE and performs integration and acceptance testing; and the Application Execution Environment (AEE), which generates operational binary code which is then implemented in the target environment [Air Force Gunter Laboratory, 1991].

- Project 1175, Software Engineering Life-Cycle, of the Institute of Electrical and Electronics Engineers (IEEE). Like STARS, PCIS and I-CASE (described above) this project is directed at standards for a complete software engineering environment that addresses the entire software development life-cycle, but without the particular focus on Ada (as is the case with STARS, PCIS and I-CASE). The Department of Defense (DoD) - Office of the Secretary of Defense (OSD) is, however, actively involved in this work as well. One major area of current work effort is the development of modeling language meta-models for purposes of integrating various models used during systems analysis and design. Data Flow Diagrams (DFD) are a key element of the P1175 modeling life-cycle. The development of a DFD meta-model is being coordinated with the CDIF, IPO SUMM and ANSI IRDS efforts described above.
- Tacitus project at the Stanford Research Institute (SRI).

### 3.4. Repository/Data Dictionary Technologies

#### 3.4.1 Categories Of Data Dictionaries

The schemas of existing, legacy database management systems (DBMS's) such as the IMS hierarchical DBMS, relational DBMS's based on SQL, and Codasyl or network databases are in one sense "models" of enterprise data, but their audience is limited to database administrators, applications programs, and in some cases systems software such as DBMS report generators. Traditional application programs, report generators, etc. do not need to have explicit representation of the meaning of the data they access and manipulate, and so naturally DBMS schemas do not explicitly model the semantics of the data separately from its implementation form. In the case of the IMS DBMS, even the implementation form is not available in one place, as one must look at the COBOL program copylib (or copybook) members, as well as the IMS data structures and access commands, to see the complete implementation schema.

Some existing data management environments do segregate the implementation schema, at least to some degree, from the [logical] representation of the data elements that are being implemented. These two-schema environments employ a data directory as the mechanism for achieving this segregation. Wertz [1986] defines a *data directory* as a machine-readable definition of a computerized database. It contains the record structure and data elements from the database schema along with information about the size, format and, physical storage location of the data elements. Wertz goes on to define a *data dictionary* as a superset of a data directory that contains additional information, such as definitions, pertaining to the data elements. Data dictionaries may be classified into three categories based on their relationships to DBMS's and the capabilities that they exhibit:

## UKM State-of-the-Art Review

- Static or Passive — a data dictionary that must be maintained manually by a database administrator. Changes to the dictionary are not automatically ramified to application programs that use the dictionary, i.e. the database definitions used by programs are not directly coupled to a static data dictionary.
- Active — changes to the data dictionary must be made manually by a database administrator, but such changes may initiate a re-compile of the application program database definitions (i.e. the coupling is made closer through checks in the DBMS pre-compiler for changes to the data dictionary).
- Dynamic — changes to the data dictionary may be initiated automatically by the system itself (e.g. as the result of some processing someplace else in the system). These changes automatically trigger any necessary changes to application programs and the change is made transparently. However, to the degree that any semantics reside in the program logic itself and not in the data dictionary, a dynamic data dictionary still cannot validate whether the change will affect the semantic integrity of the program itself. In the long-run, dynamic data dictionaries may work best in artificial intelligence-based environments where program logic is encoded in the knowledgebase, along with the semantics of the information that is accessed, processed and/or produced by execution of the program logic.

Data dictionaries, whether they are static, active or dynamic, may still only be two-schema. Another kind of data dictionary, which is really more correctly considered a category unto itself, is a three-schema data dictionary. In the best case, a three-schema data dictionary would also be dynamic, i.e. it would exhibit the most advanced capabilities and relationships with respect to DBMS's. According to ANSI/SPARC and ISO [Tsichritzis and Klug, 1978; International Organization for Standardization (ISO), 1987a], three-schema databases must provide for the segregation of their representations into three distinct schemas: the External Schema (ES), which represents how information is acquired and presented to users via data acquisition and presentation devices such as computer terminals and various user interface mechanisms like windows and menus; the Conceptual Schema (CS), which represents the abstract meaning of information independent of its presentation or implementation form; and the Internal Schema (IS), which represents how information is stored as data by various implementation mechanisms such as file management systems or database management systems. Mappings must then be established between the Conceptual Schema and both the Internal and External Schemas. Segregation of the three schemas is important for purely representational reasons and for implementational efficiency when the objective is heterogeneous database integration.

### 3.4.2 Capabilities Of Data Dictionaries

A three-schema data dictionary should support representation of: file management systems and database management systems; application systems; programming languages; modeling languages; operating systems; presentation or display devices; and distributed system networks. Simple, static or active three-schema data dictionaries may be referred to as common data models. Data dictionaries that are more dynamic and have additional representational depth (i.e. they allow the representation of additional semantics about the information in them) are referred to as common semantic models or federated semantic database management systems. The

Information Resources Dictionary System (IRDS) II (described in Section 3.3., Model Extension and Integration Projects and Approaches, of this document) has as its objective the development of standards for three-schema, dynamic data dictionaries. Specifically, the Conceptual Schema of a three-schema, dynamic data dictionary should support such capabilities as:

- Representation of information entities;
- Representation of relations, including non-normalized relations and various kinds of rules (e.g. business rules, global or common rules, rules specific to an entity, rules applied to other rules);
- Re-representation of other representational forms, including hierarchical, relational and network/Codasyl DBMS's;
- Explicit, unlimited definition of types or classes;
- Inheritance from [super]classes to subclasses, including both single and multiple inheritance;
- Collection or grouping of entities (e.g. data elements into records, and records into groups of records);
- Representation of processes:
  - Representation of processes in the same representation system as static information entities;
  - Fully declarative representation of processes;
  - Dynamic execution of processes from within the representation system;
  - Declarative representations of update propagation strategies (e.g. immediate, on request, periodic, push, pull, etc.);
  - Incremental and dynamic binding;
  - Procedural attachment (i.e. custom routines executable by a database manager facility).
- Supports representation of modes of intension (e.g. belief states, time, probability, etc.).

The External Schema of a three-schema, dynamic data dictionary should support representation of:

- Aliases for data elements;
- Data element data types and formats;
- Derived data elements;
- Display devices;
- Windowing environments;
- Users (both human users and application programs as "users"):
  - Classes of users;
  - Individual users;
  - Skill levels and preferences of human users.

The Internal Schema of a three-schema, dynamic data dictionary should support representation of:

- Physical storage location of a particular datum in a particular file or database management system (e.g. a specific file operating under a particular file management system, or a specific database operating under a particular DBMS);

## FKM State-of-the-Art Review

- Actual bit block form in which a datum is stored;
- Operating system under which the file management system or DBMS operates;
- Computer system or systems on which the datum resides;
- Network protocol(s) which must be used to communicate between the host system and the computer system(s) on which the datum resides;
- Representation system in which the data dictionary is implemented is itself explicitly represented in the three schemas of the data dictionary (i.e. the system is self-applicable).

Each of the schemas in a three-schema, dynamic data dictionary is explicitly represented and directly accessible for data administration and application programming. There are a number of underlying primitives that are either directly represented in a three-schema, dynamic data dictionary or constructable from other primitives. These primitives include: keys (including primary, secondary, *n*-ary, foreign); functions; record types; set types; arrays; bags (i.e. unordered sets); sequence; union; intersection; and projection (including directly defined attributes and attributes that are derived from classes — partially and combinatorily — at the time of instantiation). Real-time, on-line changes may be made to three-schema, dynamic data dictionaries and explicit versions of the schemas are maintained. In the longer-run, these data dictionaries will also be self-extensible, i.e. dynamic changes may be made to schemas by the system itself (i.e. by execution of procedures or rules for change processes represented in the three schemas of the data dictionary itself).

In addition to representing their own underlying DBMS, three-schema, dynamic data dictionaries support subsumption of foreign DBMS's (i.e. DBMS's other than the DBMS used by the data dictionary itself). This means that arbitrary foreign DBMS's must be fully representable within the three schemas of the data dictionary. Utilities should be included for automatic acquisition of foreign DBMS data structures and commands, both in terms of syntax and semantics. This may be accomplished by a hard-wired parser, but more advanced, flexible systems would use a dynamic parser which draws upon a meta-schematic representation of the foreign DBMS's. Subsumption in its most advanced form also involves the ability to perform dynamic, distributed updates to the databases of the foreign DBMS's which have been subsumed.

Data in databases represented in a three-schema, dynamic data dictionary retains its independence. That means changes to the re-represented database schemas (both the structure of the databases and their contents) do not require a database unload and reload. Data persistence is explicitly represented, as is temporal versioning of the data. Non-destructive updating of databases may also be a capability of advanced three-schema, dynamic data dictionaries.

Three-schema, dynamic data dictionaries are portable and fully-distributable. "Portable" means that the data dictionary implements the Open Systems Interconnect (OSI) Reference Model and is compliant to that model at all layers [of interface]. Such data dictionaries run on heterogeneous hardware, under various operating systems and using heterogeneous user interface mechanisms. Their data access mechanism is uniform regardless of memory devices. "Distributed" means that the data dictionary allows access to multiple, geographically-dispersed databases, and that the dictionary itself is distributable. Distribution services are provided by heterogeneous network protocols such as TCP/IP and SNA LU6.2.

### 3.4.3 Commercially Available Data Dictionary Technologies

Various static and active data dictionaries are currently available in the commercial software marketplace. Several are mentioned in the summary of industry survey results presented

in this document, specifically in Section 3.3.10., Re-Engineering and Repositories/Data Dictionaries. Most are provided by software vendors who also market a DBMS, particularly relational DBMS's (since they lend themselves somewhat better to data dictionary capabilities than existing hierarchical databases, and since they are much more common in industry than network/Codasyl databases). Others have been developed in-house by various companies for their internal use. There is only one common data model that is currently known to be marketed commercially. That is the Common Data Model (CDM)\*Plus® from Control Data Corporation. Various companies in industry are also attempting to implement three-schema concepts using existing data dictionary software or customized variations of such software. Most major vendors of DBMS technology are actively developing three-schema data dictionaries. The original IRDS I (Information Resources Dictionary System I) effort, and related extensions conducted more recently (i.e. A Tools Integration Standard or ATIS), have developed standards for initial implementations of static, active and even limited dynamic, three-schema data dictionary technology using traditional relational (i.e. SQL-based) approaches combined with [limited] aspects of the object-oriented paradigm. IRDS I [American National Standards Institute (ANSI), 1988] is based on the ISO concept of "level-pairs" which allows for levels of abstraction for database representation. Proprietary DBMS's can still have their own internal implementations, but can interchange database files with other DBMS's using a [relational] SQL-based neutral exchange format — and without losing important syntax and semantics about the data structures being exchanged. ATIS [American National Standards Institute (ANSI), 1991a] is an extension to IRDS I that involves a [limited] object-oriented approach to the integration of DBMS tools, again using SQL, but adding aspects of the object-oriented paradigm. In fact, ATIS has been described by some as an attempt to bring the advantages of the object-oriented paradigm to the existing SQL-based DBMS environment. The object-oriented modeling paradigm is described under a separate subheading within Section 3.2., Modeling Methods and Tools, of this document. ATIS provides a set of standard services interfaces that support schema-driven dispatching of behavior. ATIS provides an interface to and defines a data model for an initial Information Resources Dictionary (IRD) or "repository". The ATIS interface and data model support the services interfaces defined under IRDS I and, to some degree, the expanded services interfaces emerging under IRDS II. Several products that implement the IRDS I standard and/or ATIS have been announced, and in some cases are already in use in industry. The IRDS II effort is currently working on a specification of standards for a more advanced three-schema, dynamic data dictionary. IRDS II is described in more detail under a separate subheading within Section 3.3., Model Extension and Integration Projects and Approaches, of this document. There are currently no three-schema, dynamic data dictionaries available commercially.

## ITKM State-of-the-Art Review

---

AD/Cycle, Business Systems Planning, DB2, DL/1, HIPO, IBM and IMS are registered trademarks of International Business Machines Corporation.

ASA and Automata are trademarks or registered trademarks of Verilog, S.A.

C++ is a registered trademark of AT&T Bell Laboratories.

Cadillac and Seville are registered trademarks of General Motors Corporation.

CASE\* and ORACLE are registered trademarks of Oracle Corporation.

CDM\*Plus, IAST and PC-IAST are registered trademarks or trademarks of Control Data Corporation.

Design/CPN and Design/IDEF are registered trademarks of Meta Software Corporation.

Hypercard and Macintosh are registered trademarks of Apple Computers, Inc.

IDEFINE-0 and IDEFQual are trademarks of Wizdoms Systems, Inc.

Information Access Vehicle (IAV) is a trademark of Ontek Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

Ontos is a registered trademark of Ontologic, Inc.

Ptech is a trademark of Associative Design Technology (U.S.), Ltd.

RDD-100 is a registered trademark of Ascent Logic, Inc.

RIDI\* is a trademark of IntelliBase, Inc.

SADT is a registered trademark of SofTech, Inc.

Virtual Software Factory is a registered trademark of Systematica, Ltd.

## **4. Supporting Data**

### **4.1. Overview Of Data Gathering Approach**

Ontek gathered data regarding state-of-the-art methods and tools for enterprise analysis and modeling from numerous organizations and individuals in industry and academia, as well as from computer hardware and software developers and vendors. Ontek's three corporate sponsors (The Aluminum Company of America, Westinghouse Electric Corporation and Northrop Corporation) were included among the sources from industry, as were members of Ontek's original Industry Advisory Board. Ontek also coordinated with the American National Standards Institute (ANSI) X3H4 committee working on the Information Resources Dictionary System (IRDS) Phase II project, the IDEF Users' Group and other professional associations regarding the opinions and insights of their members. Extensive searches of literature were conducted using the Defense Technical Information Center (DTIC) and other sources of reference materials.

### **4.2. Survey**

As part of the review and assessment of the State-of-the-Art (SOTA) of existing methods and tools, a survey was conducted of: users of modeling methodologies and tools in the aerospace/defense and commercial manufacturing industry and other industries heavily dependent on information processing; consulting and systems integration companies who provide enterprise analysis and modeling and/or systems development services to industry; developers and vendors of methods and tools utilized for enterprise analysis and modeling and/or systems development; academicians in universities or related research facilities; and enterprises that are combinations of any of the other categories. The results of the survey were used as input to and drivers for many of the requirements expressed in the complementary ITKM Needs Analysis Document (entitled "Needs Analysis and Requirements Specification For an Integration Toolkit and Methods (ITKM)"). For a more complete and detailed description of the survey results, the reader is referred to Section 2., Summary Of Industry Survey Results, as well as a version of the survey form populated with the actual results (which appears as Attachment A to this document).

### **4.3. Coordination With Other Related Organizations and Projects**

In addition to the survey discussed in Section 4.2., data regarding the state-of-the-art was gathered based on experience from real-world, enterprise integration projects being conducted by Ontek's corporate sponsors (the Aluminum Company of America (Alcoa), Northrop Corporation - Aircraft Division, and Westinghouse Electric Corporation - Electronic Systems Group). Specifically, these include: the Automated Airframe Assembly Program (AAAP), Northrop Aircraft Division Strategic Architecture Project (NADSARP) and the Systems Engineering Environment (SEE) project at Northrop Corporation; the Computer-Integrated Enterprise (CIE), [proposed] Enterprise Integration Program (EIP) and Design, Engineering and Manufacturing Operations Information System (DEMOIS) projects at Westinghouse Corporation; and the IMS Data Directory/Dictionary and Alcoa Information Architecture (AIA) projects at Alcoa.



## UKM State-of-the-Art Review

Data was also gathered through participation in various industry standards efforts, consortia, professional societies and users' groups in which existing methods and technologies, problems, requirements and potential solutions are shared and discussed. Specifically, these activities include close coordination with:

- The Integrated Computer-Aided Manufacturing (ICAM) Definition (IDEF) methodology Users' Group (UG). Ontek has attended many of the IDEF UG conferences and workshops over the past few years and actively participates in the Research and Development Forums;
- The American National Standards Institute (ANSI) X3H4 Information Resources Dictionary System (IRDS) Phase II project. Ontek is an active participant and voting member of the ANSI X3H4 IRDS II Committee. Ontek is particularly active in the X3H4.6 IRDS Conceptual Schema Task Group and contributes to the technical reports produced by this group;
- The IBM GUIDE Users' Group. Ontek, in conjunction with Alcoa, has participated and presented at a number of GUIDE meetings, primarily regarding the issues of knowledge representation and legacy database integration and access; and
- The American Society of Engineering Management/National Council On Systems Engineering (ASEM/NCOSE). Ontek, in conjunction with Northrop Corporation, has been actively involved in the definition of the systems engineering methodology and process, and the determination of requirements for information technologies to support systems engineering. An Ontek representative recently published a technical paper on this subject and presented the work at a recent national conference.

More general coordination, liaison or harmonization activities with other organizations or projects also provided some additional source material. These included: the Initial Graphics Exchange Standard (IGES)/Product Definition Exchange using STEP (PDES) Organization or IPO [Note: STEP stands for Standard for The Exchange of Product data, and is the international counterpart to the U.S. PDES project], particularly the Dictionary Methodology Committee working on the Semantic Unification Meta-Model; the Defense Advanced Research Projects Agency (DARPA) and Stanford Research Institute (SRI) regarding the Shared Reusable Knowledgebase (SRKB) project; and the Computer-Aided Acquisition and Logistics Support (CALs)/Concurrent Engineering (CE) initiative, regarding Contractor Integrated Technical Information System (CITIS) and CALs Phase II requirements and state-of-the-art methods and technologies recommended to meet the requirements.

## 4.4. Reference Materials

### 4.4.1. Bibliography

Abrial, J.R., and Klimbie, J.W. and Koffemen, K.L. (Eds.), 1974. Data Semantics. *Data Base Management*, 1-59. North-Holland Publishing Company, Amsterdam, The Netherlands.

Air Force Gunter Laboratory, 1991. Request For Information (RFI) for Integrated-Computer Aided Software Engineering (I-CASE), version 12/18/91. Air Force Gunter Laboratory, Gunter Air Force Station, Alabama, U.S.A.

Albano, A., DeAntonellis, V., and DiLeva, A., 1985. Computer-Aided Database Design. North-Holland Publishing Company, Amsterdam, The Netherlands.

Alcoa, 1989. Quality Improvement Handbook. The Aluminum Company of America, Pittsburgh, Pennsylvania, U.S.A.

Alford, M., 1991. Strengthening the Systems Engineering Process. Proceedings of the *Joint Conference of the American Society for Engineering Management (ASEM) and the National Council on Systems Engineering (NCOSE)*, Chattanooga, Tennessee, U.S.A., October 1991.

American National Standards Institute (ANSI), 1988. Information Resource Dictionary System (IRDS). ANSI X3.138-1988. American National Standards Institute - X3H4 Information Processing Systems Committee, IRDS Working Group, U.S.A.

American National Standards Institute (ANSI), 1991a. Information Resource Dictionary System (IRDS) - A Tool Integration Standard (ATIS), ANSI X3H4 Working Draft, October 1991. American National Standards Institute - X3H4 Information Processing Systems Committee, IRDS Working Group, U.S.A.

American National Standards Institute (ANSI), 1991b. Final Report Of the OODBTG, X3/SPARC/DBSSG/OODBTG, September 17, 1991. American National Standards Institute - Object-Oriented Database Task Group, U.S.A.

Associative Design Technology, 1990a. Ptech: The Object-Oriented CASE Environment. *Introductory Presentation*. Associative Design Technology (US) Ltd., Westborough, Massachusetts, U.S.A.

Associative Design Technology, 1990b. Ptech Events and Other Objects, Vol. 2, No. 3, September 1990. Associative Design Technology (US) Ltd., Westborough, Massachusetts, U.S.A.

Blanchard, B. and Fabrycky, W., 1990. Systems Engineering and Analysis, Second Edition. Prentice Hall, Inc., Englewood Cliffs, New Jersey, U.S.A.

Bracchi, G., Paolini, P., and Pelagatti, G., 1976. Binary Logical Associations in Data Modeling. *Modeling in Database Management Systems, Conference Proceedings of IFIP TC2*. Freudenstadt. North-Holland Publishing Company, Amsterdam, The Netherlands.

UKM State-of-the-Art Review

Burkhart, R., Dickson, S., Hanna, J., Perez, S., Sarris, A., Singh, M., Sowa, J. and Sundberg, C., 1991. ISO/IEC JTC1/SC21/WG3 N. Information Resources Dictionary System (IRDS) Conceptual Schema Working Paper. American National Standards Institute (ANSI) X3H4 IRDS, U.S.A.

Ceri, S. (Ed.), 1983. Methodology and Tools For Database Design. North-Holland Publishing Company, Amsterdam, The Netherlands.

Chen, P.P., 1976. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, Vol. 1, No. 1, 9-36.

Codd, E.F., 1979. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions On Database Systems*, Vol. 4, No. 4, 397-434.

Cox, B., 1987. Object-Oriented Programming. Addison Wesley Publishing Company, Reading, Massachusetts, U.S.A.

D. Appleton Company (DACOM), 1985. Information Modeling Manual - IDEF1 Extended, AFWAL-TR-86-4006. Air Force Materials Laboratory, ManTech Division, CIM Branch, Integrated Information Support System (IISS) Program, Volume V - Common Data Model Subsystem, Pt.4.

D. Appleton Company (DACOM), 1988. IDEF1X ENHANCEMENTS - Detailed Descriptions of Candidate Enhancements, Document #D-697-88-01. Submitted by DACOM to Control Data Corporation (CDC) Manufacturing Technology/CIM Division for delivery to the Air Force Materials Laboratory, ManTech Division, CIM Branch, under the Air Force Data Automation Processor (DAPro) Program, May 16, 1988.

Davis, H., 1991. ECMA/TC33-TGEP/91/57, memo and presentation materials entitled *Harmonization of Standards*. European Computer Manufacturers Association (ECMA).

Department of Defense (DoD), 1991. Pre-Coordination Draft of MilStd 499B, Systems Engineering. Aeronautical Systems Division (ASD/ENS), Wright-Patterson Air Force Base, Ohio, U.S.A.

Dement, C.W. and Woodruff, P.W., 1988. The 42 Memo: Translating IMS To Relational Databases. Ontek Corporation, Laguna Hills, California, U.S.A.

DiLeva, A. and Giolito, P., 1986. Data Models and Process Models For Computer-Integrated Manufacturing Systems. *Proceedings of First International Conference on Applications of Artificial Intelligence to Engineering Problems*, 513-526.

Edwards, D.D. and Mayer, R.J., 1989. IDEF4 Technical Report. Prepared for the Air Force Human Resources Laboratory - Logistics and Human Factors Division, Wright-Patterson Air Force Base, Ohio and the National Aeronautics and Space Administration RISC Program at the University of Houston, Houston, Texas. Knowledge Based Systems, Inc., College Station, Texas, U.S.A.

Elam, J., Henderson, J.C., Miller, L.W., 1980. Model Management Systems: An Approach To Decision Support In Complex Organizations. *Proceedings of the First International Conference on Information Systems*, 98-110.

ESPRIT Consortium AMICE (Ed.), 1989. Open System Architecture for CIM. Research Reports ESPRIT, Project 688, AMICE, Volume 1. Springer-Verlag, Berlin, Germany.

Farah, B.N. and Kusiak, A. (Ed.), 1988. CIM Databases. *Artificial Intelligence: Implications For Computer Integrated Manufacturing*, 12:435-457. IFS (Publications) Ltd., Kempston, Bedford, United Kingdom, and Springer-Verlag, Berlin, Federal Republic of Germany and New York, New York, U.S.A.

Fraser, M.D., Kumar, K., and Vaishnavi, V., 1991. Informal and Formal Requirements Specification Languages: Bridging the Gap. *IEEE Transactions on Software Engineering*, Vol. 17, No. 5, May 1991, 454-466.

Freeman, P., 1987. Software Perspectives: The System Is the Message, 5.2:125. Addison Wesley Publishing Company, Reading, Massachusetts, U.S.A.

Fulton, J.A., 1990. SUMM Semantic Constraints: Formal Definitions of Semantic Constraints for the STEP Unification Meta-Model (SUMM). The Boeing Company, Seattle, Washington, U.S.A.

Fulton, J.A., 1991a. Architecture for Standard Models. Presentation to the GUIDE 79 IBM Users' Group on March 20, 1991. IGES/PDES Organization (IPO) and the International Organization for Standardization (ISO).

Fulton, J.A., 1991b. The STEP Unification Meta-Model. Presentation to the GUIDE 79 IBM Users' Group IGES/PDES Organization (IPO) and the International Organization for Standardization (ISO) in Anaheim, California, March 20, 1991.

Gadre, S., 1987. Building an Enterprise and Information Model. *Computer Applications in Spacecraft Design and Operation, Conference Proceedings of COMPAS '87*, Paris, France.

Gardner, J. (Ed.), 1991. *STARS Newsletter*, June 1991. Unisys Defense Systems, Inc., Arlington, Virginia, U.S.A.

Given, R., McAllester, D., and Zalondek, K., 1991. Ontic91: Language Specification and User's Manual. Draft 3, July 22, 1991. Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts, U.S.A.

Griskey, T.C., Sarris, A.K., Santos, H., and MacAllister, I., 1991. A White-Paper on Systems Engineering in PACIS. Northrop Corporation - Aircraft Division, Hawthorne, California, U.S.A. and Ontek Corporation, Laguna Hills, California, U.S.A.

ITKM State-of-the-Art Review

Gruber, T., 1991. Ontolingua: A Mechanism To Support Portable Ontologies, Draft of September 13, 1991. Knowledge Systems Laboratory, Stanford University, Palo Alto, California, U.S.A.

GUIDE, 1991. Data Modeling Extensions. GUIDE Project AE-5200P (formerly DP-1853) Project Paper Draft. GUIDE IBM Users' Group.

Hatley, D. and Pirbhai, I., 1988. Strategies for Real-Time System Specification. Dorset House Publishing Co., Inc., New York, New York, U.S.A.

Hauser, J.R. and Clausing, D., 1988. The House of Quality. *Harvard Business Review*, May/June 1988, 63-73.

Hill, R.R. Jr., 1991. Decision Support Environment for Concurrent Engineering Requirements. Final Technical Paper for Period June 1989 - November 1990. Air Force Systems Command, Human Resources Laboratory, Logistics and Human Factors Division, Wright-Patterson Air Force Base, Ohio, U.S.A.

Hull, R. and King, B., 1987. Semantic Data Modeling. *ACM Computing Surveys*, Vol. 19, No. 3, 200-260.

IGES/PDES Organization (IPO) - Dictionary Methodology Committee (DMC), 1991. The Semantic Unification Meta-Model: Technical Approach, Version 0, Release 6, Working Draft 3, October 7, 1991. IGES/PDES Organization (IPO), U.S.A.

International Organization for Standardization (ISO), 1987a. Information Processing Systems — Concepts and Terminology for the Conceptual Schema and the Information Base. ISO/TR 9007 (E). International Organization for Standardization (ISO).

International Organization for Standardization (ISO), 1987b. Information Processing Systems — Concepts and Terminology for the Conceptual Schema and the Information Base, ISO/TR 9007 (E), 4:63-70. International Organization for Standardization (ISO).

Jorgenson, B.R., 1991a. Business Model Glossary. The Boeing Company, Seattle, Washington, U.S.A.

Jorgenson, B.R., 1991b. Conceptual Schema System Architecture. The Boeing Company, Seattle, Washington, U.S.A.

Jorgenson, B.R. and Walter, P.A., 1991. Business Entity Relationship Model (BERM). The Boeing Company, Seattle, Washington, U.S.A.

Keuffel, W., (1991). House of Structure. *UNIX Review*, Vol. 9 No. 2.

Leite, J.C.S.d.P., and Freeman, P.A., 1991. Requirements Validation Through Viewpoint Resolution. *IEEE Transactions on Software Engineering*, Vol. 17, No. 12, December 1991, 1253-1269.

Lenat, D., Prakash, M. and Shepherd, M., 1986. CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. *The AI Magazine*, Winter 1986.

Martin, J., 1990. Information Engineering Book II: Planning and Analysis. Prentice Hall, Inc., Englewood Cliffs, New Jersey, U.S.A.

Mayer, R.J. and Painter, M.K., 1991. The IDEF Suite of Methods For System Development and Evolution. Prepared for the Air Force Human Resources Laboratory - Logistics and Human Factors Division, Wright-Patterson Air Force Base, Ohio. Knowledge Based Systems, Inc., College Station, Texas, U.S.A.

Meta Software Corporation, 1989. Design/CPN: A Tool Package Supporting the Use of Colored Petri Nets. Meta Software Corporation, Cambridge, Massachusetts, U.S.A.

Microelectronics and Computer Technology Corporation (MCC), 1991. Technical Issues of Enterprise Integration, MCC Technical Report, CAD-349-91. Microelectronic and Computer Technology Corporation (MCC), Austin, Texas, U.S.A.

Miller, G.A., 1956. The Magical Number Seven, Plus or Minus Two: Some Limits On Our Capacity For Processing Information. *Psychological Review*, 63: 81-97.

Miller, R.E., 1973. A Comparison Of Some Theoretical Models Of Parallel Computation. *IEEE Transactions On Computers*, C-22(8).

Novak, J. and Gowin, D. B., 1984. Learning How to Learn. Cambridge University Press, Cambridge, United Kingdom.

Quillian, M.R. and Minsky, M. (Ed.), 1968. Semantic Memory. *Semantic Information Processing*, MIT Press, Cambridge, Massachusetts, U.S.A.

Radeke, E., 1991. DaDaMo: The DASSY Data Model. CADLAB, Paderborn, Germany.

Roques, P., 1991. Industrial Applications of ASA (IDEF0 and Automata) in Europe. Proceedings of the IDEF Users' Group Conference on *Advancing Modeling Methods Technology*. Fort Worth, Texas, U.S.A.

Ross, R.G., 1987. Entity Modeling: Techniques and Application. Database Research Group, Inc., Boston, Massachusetts, U.S.A.

Sarris, A.K., 1988. Observations on the ICAM Definition (IDEF) Modeling Language. ONTEK/TR-88/004. Prepared by Ontek Corporation, Laguna Hills, California, U.S.A. for Northrop Corporation - Aircraft Division and the United States Air Force, Manufacturing Technology Directorate, Integration Division, CIM Branch under the Automated Airframe Assembly Program (AAP), Phase IIA -Advanced Technology Development, Contract #F33615-87-C-5217.

ITKM State-of-the-Art Review

Sarris, A.K., 1991. Systems Engineering Role in Process Management. Proceedings of the *Joint Conference of the American Society for Engineering Management (ASEM) and the National Council on Systems Engineering (NCOSE)*, Chattanooga, Tennessee, U.S.A., October 1991.

Schenck, D.A. and Spiby, P., 1991. EXPRESS Language Reference Manual, Working Draft, January 10, 1991. International Organization for Standardization.

Senko, M.E., 1977. Conceptual Schema, Abstract Data Structures, Enterprise Descriptions. *International Computing Symposium*. North-Holland Publishing Company, Amsterdam, The Netherlands.

Sloan, D., 1991. Using IDEF to Define Systems Requirements. Proceedings of the IDEF Users' Group Conference on *Advancing Modeling Methods Technology*. IDEF Users' Group, Fort Worth, Texas, U.S.A.; and *The IDEFine Family*, Wizdom Systems, Inc. product literature. Wizdom Systems, Inc., Naperville, Illinois, U.S.A.

Sowa, J.F., 1984. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, Massachusetts, U.S.A.

Sowa, J.F., 1991a. Matching Logical Structure to Linguistic Structure. Presentation to American National Standards Institute (ANSI), X3H4 Information Resources Dictionary System (IRDS) Project Committee, Conceptual Schema Task Group in Seattle, Washington, U.S.A., August 1991.

Sowa, J.F., 1991b. Towards the Expressive Power of Natural Language. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, edited by J.F. Sowa, Morgan Kaufmann Publishers, San Mateo, California, U.S.A., pp. 157-189.

Sowa, J.F., 1991c. Lexical Structures and Conceptual Structures. To appear in *Semantics in the Lexicon*, edited by J. Pustejovsky, Kluwer Academic Publishers.

Sowa, J., 1991d. Petri Nets For Modeling Concurrency. Presentation to the ANSI X3H4.6 IRDS Conceptual Schema Task Group, October 1991, Gaithersburg, Maryland, U.S.A.

Steel, T.B. Jr., 1978. A Modest Proposal For a Conceptual Schema Language. *Conference Proceedings of SHARE LI, Volume 1*. SHARE Inc. Chicago, Illinois, U.S.A.

Su, S.Y.W., 1983. SAM: A Semantic Association Model for Corporate and Scientific-Statistical Databases. *Information Sciences*, Vol. 29, 151-199.

Su, S.Y.W., 1986. Modeling Integrated Manufacturing Data with SAM. *IEEE Computer*, Vol. 19, No. 7, 34-49.

Sullivan, L.P., 1989. Policy Management Through Quality Function Deployment. *Quality Progress*, June 1989, 18-20.

Tsichritzis, D.C. and Klug, A., (Ed.), 1978. "The ANSI/X3/SPARC DBMS Framework", *Information Systems*, Vol. 3, 173-191.

Tyler, J., 1990. A Planning Model for Unifying Information Modeling Languages for Product Data Exchange Specification (PDES). NISTIR 90-4234. U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Center for Manufacturing Engineering, Gaithersburg, Maryland, U.S.A.

Vernadat, F. and Kusiak, A. (Ed.), 1988. CIM Databases. *Artificial Intelligence: Implications For Computer Integrated Manufacturing*, 7:270-279. IFS (Publications) Ltd., Kempston, Bedford, United Kingdom, and Springer-Verlag, Berlin, Federal Republic of Germany and New York, New York, U.S.A.

Wertz, C.J., 1986. *The Data Dictionary: Concepts and Uses*. QED Information Sciences, Inc., Wellesly, Massachusetts, U.S.A.

Yourdon, Inc., 1980. *Structured Analysis/Design Workshop*. Yourdon Press, New York, New York, U.S.A.

Zachman, J.A., 1987. A Framework For Information Systems Architecture. *IBM Systems Journal*, Vol. 26, No. 3. International Business Machines Corporation. White Plains, New York, U.S.A.

#### 4.4.2. Other Source Documents

Brodie, M., Mylopoulos, J., and Schmidt, J.W. (Ed.), 1984. *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer-Verlag, New York, New York, U.S.A.

Creasy, P. and Moulin, B., 1991. Approaches to Data Conceptual Modelling Using Conceptual Graphs. Proceedings of *6th Annual Workshop on Conceptual Graphs*. University of Queensland, Department of Computer Science, Queensland, Australia and Université Laval, Département d'informatique, Ste Foy, Québec, Canada.

Cyre, W., 1991. Integrating Knowledge in Digital Systems Specifications using Conceptual Graphs. Proceedings of *6th Annual Workshop on Conceptual Graphs*. Virginia Polytechnic Institute and State University, The Bradley Department of Electrical Engineering, Blacksburg, Virginia, U.S.A.

Davis, C., Jajodia, S., Ng, P., and Yeh, R.T. (Ed.), 1983. *Entity-Relationship Approach to Software Engineering*. North-Holland Elsevier Science Publishers B. V., Amsterdam, The Netherlands.

Esch, J.W., 1991. Visualizing Temporal Intervals Represented as Conceptual Graphs. Proceedings of *6th Annual Workshop on Conceptual Graphs*. Unisys, St. Paul, Minnesota.

Feldman, C.G., Preston, R.R., and Thompson, P.S., 1991. Activity and Information Modeling Methodology Assessment. Task Order Final Report for Period May 1990 - November 1990.



ITKM State-of-the-Art Review

WL-TR-91-8012. Air Force Systems Command, Wright Laboratory, Manufacturing Technology Directorate, Dayton, Ohio, U.S.A.

Gray, L.C. and Bonnell, R.D., 1991. A Comprehensive Conceptual Analysis using ER and Conceptual Graphs. Proceedings of *6th Annual Workshop on Conceptual Graphs*. University of South Carolina, Department of Electrical and Computer Engineering, Columbia, South Carolina, U.S.A.

IGES/PDES Organization (IPO), 1991. Summary and Notes of the Joint Meeting of the ISO TC184/SC4 and the IGES/PDES Organization, San Diego, California, April 3-12, 1991. IGES/PDES Organization (IPO) and the International Organization for Standardization (ISO).

Kocura, P., Ho, K.K., Moorhouse, D., and Sharpe, G., 1991. *Proceedings of 6th Annual Workshop on Conceptual Graphs*. Loughborough University of Technology, Department of Computer Studies, Loughborough, Leicestershire, United Kingdom.

Kusiak, A. (Ed.), 1988. Artificial Intelligence: Implications for CIM. IFS (Publications) Ltd., Kempston, Bedford, United Kingdom, and Springer-Verlag, Berlin, Germany and New York, New York, U.S.A.

Meersman, R.A., Shi, Zh., and Kung, C-H. (Ed.), 1990. Knowledge Representation in Databases, Expert Systems, and Natural Language. *Artificial Intelligence in Databases and Information Systems (DS-3)*, 17-50. North Holland Publishing Co., Amsterdam, The Netherlands. Based on an Invited Talk at the IFIP Conference on the Role of Artificial Intelligence in Databases and Information Systems in Guangzhou, China, July 1988.

Rothenberg, J., 1989. Object-Oriented Simulation: Where Do We Go from Here? The Rand Corporation, Santa Monica, California, U.S.A.

Rothenberg, J., 1989. The Nature of Modeling. The Rand Corporation, Santa Monica, California, U.S.A.

Sarris, A.K., et al, 1988. Strategic Analysis and Planning Issues of Inter-Organizational Computer Integrated Manufacturing. AFWAL-TR-88-4122. Prepared by Ontek Corporation, Laguna Hills, California, U.S.A. for Control Data Corporation and the United States Air Force, Manufacturing Technology Directorate, Integration Division, CIM Branch under the Data Automation Processor (DAPro) Program, Contract #F33600-87-C-0464.

Schmidt, J. and Kocura, P., 1991. Open Systems: Towards Synergy in Conceptual Graph Processor Construction. Proceedings of *6th Annual Workshop on Conceptual Graphs*. Czech Technical University, Department of Computers, Prague, Czechoslovakia and Loughborough University of Technology, Department of Computer Studies, Loughborough, Leicestershire, United Kingdom.

Smith, D.W., 1989. The ABC's of Representing Representation. ONTEK/WP-89/019. Ontek Corporation, Laguna Hills, California, U.S.A.

Sowa, J.F., 1990a. Crystallizing Theories out of Knowledge Soup. *Intelligent Systems: State of the Art and Future Directions*, 456-487, edited by Z.W. Ras and M. Zemankova. Ellis Horwood Publishers, New York, New York, U.S.A.

Sowa, J.F., 1990b. Finding Structure in Knowledge Soup. Proceedings of *Info Japan '90*. Information Processing Society of Japan, Tokyo, Japan, Vol. 2. 245-252.

## Attachment A

### Populated Survey of Enterprise Modeling Methodologies and Tools

- 25 Responses (complete or nearly complete)
  - 14 Industry User
  - 3 Consulting (including views representative of their customer sites)
  - 2 Academia
  - 2 Vendor
  - 2 Blind
  - 2 Both Vendor and Industry User

**Note:** Where the number of responses exceeds the numbers of respondents in any category, a given respondent may have provided more than one answer. For example, multiple modeling methods and tools are typically used within a single enterprise. A question about modeling methods might have had 10 respondents who identified 15 methods in use in their various enterprises.

1. Which of the following kinds or aspects of enterprise modeling are currently performed in your enterprise? (check all that apply)

- 24 Functional Modeling
  - 19 Node Tree/Functional Hierarchy Modeling
  - 21 Modeling of Elements Related to Functions
    - 21 Data Entities (as Inputs and Outputs)
    - 18 Organizational Entities
    - 17 Systems/Mechanisms
    - 10 Policies/Procedures
    - 12 Cost (in Dollars or Other Measures)
    - 14 Schedule/Milestones
- 24 Data Modeling
  - 18 Data Flow Modeling
- 19 Information/Semantic Modeling
  - 5 Propositions or Assertions
  - 15 Classification of Information (e.g. "Kind of", "Part of", etc.)
  - 14 Partitioning of Information into ANSI/X3/SPARC Three-Schema Architecture
    - 11 External/Presentation Schema (e.g. Aliases, Screen/Report Formats)
    - 13 Conceptual Schema
    - 12 Internal/Implementation Schema
      - 10 Domain and Range of Values
      - 11 Physical Storage Location
- 13 Declarative Process Modeling, including State/Event Modeling
  - One respondent noted that all process modeling was currently experimental at their site.*
  - 7 Temporal Process Flow
  - 12 Control Flow
    - 9 Branching Conditions

- 9 State Tables/Decision Tables
- 6 Causality/Pre-Requisite Conditions
- 4 Effect/Ramification
  - 4 Propagation Rules
- 7 Scenario Modeling
- 10 Simulation
- 6 Process Model Execution
  - 4 Directly Executable (i.e. Objective, Declarative) Process Models
- 10 Modalization (e.g. User Views; Time, including Start, End and Duration; Qualitative and Quantitative Factors; Beliefs; Probability; etc.)
 

*One respondent noted "user views only"; others noted time, probability and other scheduling factors for simulation, as well as informal ways to qualify other model data.*

  - 9 Modalization of Information
  - 9 Modalization of Processes

2. Within each of the following groups, please check the statement which best describes your current enterprise modeling environment.

Organizations:

- 2 Only one organization is responsible for all enterprise modeling;
- 2 One organization is responsible for most enterprise modeling — with the exception of data/information modeling, which is performed by a separate organization such as Data Management, Database Management, Data Administration or Database Administration;
- 10 One or two organizations serve as lead technical organizations for modeling, but various end-user organizations perform a substantial amount of the modeling;
- 8 Various end-user organizations are responsible for and perform their own modeling;
- 1 *"Various ~~end-user~~ organizations are responsible for and perform their own modeling";*
- 1 Most enterprise modeling is performed by outside contractors/consultants;
- 1 *"One or two organizations try to do some modeling."*

Techniques:

- 10 Enterprise modeling is performed using an object-oriented paradigm encompassing such techniques as information hiding (i.e. encapsulation) and messaging;
- 18 Enterprise modeling is performed based on traditional structured analysis techniques.
  - 4 Enterprise modeling is only performed top-down;
  - 0 Enterprise modeling is only performed bottom-up;
  - 14 Enterprise modeling is performed using some combination of both top-down and bottom-up approaches.

*Six respondents noted that both object-oriented and traditional structured methods are being used. Two of those stated that they are phasing out traditional structured methods in favor of object-oriented methods. One said they were trying to integrate the two methods. One noted that there are two modeling organizations working in different areas and with differing objectives — one using the traditional structured methods and the other using an object-oriented approach. Two had no additional comments*

Multiple Model Elements:

- 13 Various models of different aspects of a target domain are typically developed concurrently;

*One respondent noted that this is done iteratively; there is a rough sequence each pass through the models, but the sequence is repeated several times and there is often parallelism.*

- 8 Models of different aspects of a target domain are typically produced separately, in sequence (e.g. first functional models, then data models, then process flow models, etc.).

The sequence typically followed is: *Function-Process Flow-Data (2); Process-Data (1); Data-Function (1); Function-Data-Process Flow (1); Function-Data-Process Flow-Simulation (1); Varies (1); Not specified (1).*

*One respondent noted that they are currently only doing data modeling in their enterprise and, therefore, did not mark either of the responses above.*

#### Modeling Form:

- 9 Graphics-based models are generally utilized over text-based models;  
2 Text-based models are generally utilized over graphics-based models;  
12 A mixture of graphics and text-based models is utilized for most modeling;  
3 Graphics-based models are utilized for some kinds of modeling and text for others; it largely depends on what is being modeled.

• If graphics-based models are used for modeling some aspects of a domain and text for others, please indicate which kinds of models from 1. above are typically developed using graphics and which using text:

*Graphics: Processes; Functions (Detailed Functions); Relationships (Simple).*

*Text: 3-Schema Mappings; Critical Success Factors/Goals/Objectives; Costs; Organizations Mapped to Functions; Functions (High-Level, Business Functions); Business Rules and Other Semantics.*

#### Model Media:

- 1 Most models are created only on paper (i.e. in hard-copy form).  
7 Most models are created in digital form as word processing text files and/or CAD or drawing tool files.  
11 Most models are created using automated modeling tools, and are stored and retrieved from files or databases using the modeling tools.  
*Three respondents who noted that most models are in word processing or CAD form also noted that at least some models are in databases created by automated tools.*  
9 Models are usually created using a mixture of media.

3. Is enterprise modeling performed within the context or guidelines of an information framework or architecture? (circle one): 15 Yes 9 No 1 "Not really sure what a framework is"

*Of respondents answering "no", two were working on an architecture.*

If "Yes", is the framework/architecture home-grown or one developed by John Zachman, James Martin, Christopher Alexander or other outside parties?

*Home Grown (7), including three noted as based on industry leaders such as Zachman, CIM/OSA and Air Force Enterprise Integration Framework; Zachman (4); James Martin (3); CASE (1); USAF FoF 1105 and SDM (1); USAF IDS (1); DACOM's Three-Architecture (1); David Taylor Object-*

*Oriented (1); Robert Binder/Yourdon Structured Analysis (1); PDES/STEP (1).*

4. Please answer the following questions regarding the kinds of modeling performed for enterprise business processes, particularly decision-making processes.

How is the importance or value of the process captured in a model?

*It isn't/don't do (9); processes are mapped to CSF/goals/objectives (3); based on cost (3); weighted values or numerical ranking (3), including one noting that prioritization takes place within scenarios (3); text descriptions (2); based on schedule (1); processes mapped to other architecture components (1); identify business opportunities (1); graph actions to organizations (1); subject matter experts (1); data collection forms (1); process analysis (1).*

How is the information required for decision-making modeled?

*Is isn't/don't do (11); IDEF<sub>0</sub> functions with ICOM's (3), including one using graphics to specifically highlight those ICOM's used for decision-making; decision trees, tables and other algorithms, including conditionals (3); Entity-Attribute-Relationship (E-A-R) models/IDEF<sub>111X</sub> (2); information needs and "gaps" are modeled (1); weighted values or numerical ranking (1); in text form based on IBM enterprise model (1); model impact of organizations on processes (1); no standard procedure/ad hoc (1).*

Are organizations modeled (for example, as suppliers, owners or users/customers of data or as parties responsible for performing processes)?

*Not/not modeled (10); IDEF<sub>0</sub> mechanisms (4); as suppliers, users and/or customers of data using such techniques as SIPOC models (3); yes, no explanation (3); E-A-R/IDEF<sub>111X</sub> entities (2); as parties responsible for performing processes (2); graph actions to organizations (1); sometimes (1); both inside and outside of company (1).*

What methodologies are used, for example, are Total Quality Management-based methodologies such as Supplier-Input-Process-Output-Customer (SIPOC) and/or Quality Functional Deployment/House of Quality utilized?

*None (8), including one noting that this kind of modeling is not viewed as important in their enterprise; QFD (7), including one stating it is still being reviewed; IDEF<sub>0</sub> including functions and ICOM's (6), including several noting extensions or modifications; E-A-R/IDEF<sub>111X</sub> (5); SIPOC (3); Crosby and other TQM (2); benchmarking (1); levels of satisfaction (1); Mellor-Schlaer object-oriented information engineering (1); activity-based costing (1); CASE (1); Precise (1); SDM (1); dynamic modeling (1); Yourdon (1); GE Rumbaugh (1).*

What requirements for modeling this aspect of the enterprise are not being met or met well by existing enterprise modeling methodologies and/or tools?

*Integration (5)*

- *Model integration, esp. between function/process models and data/information models*
- *Technical standards for model interchange between different models and tools*

*Implementation/Execution (5)*

- *Relationship of models to implementation ("object-orientation seems to deal with this better than most traditional modeling tools")*



automatically linked to entities and attributes in an IDEF<sub>1X</sub> glossary or dictionary (2); unspecified CASE tools (2); Oracle Corporation's CASE\* (1); manually, based on USAF 1104 (1); USAF/IDS Integrated Resource Control System (IRCS), including an active data dictionary (1); SmallTalk (1); DACOM's The Integration Kernel (1); Accelerator (1); GE-Rumbaugh technique with Object - Maker for object model to dynamic model integration and object methods to functional model integration (1); tool known as Gadrian (1); Texas Instruments' planning methodology using matrices and business rules (1); functional models define activities and data interfaces which are manually translated into data flow and simulation models, performance data is overlayed on the functional model and transferred to the simulation models.

If "No" (including one "Rarely" response), do you have a need for model integration? (circle one):

8 Yes — Between/among which kinds of models (using the categories from 1. above)?

Function/process models to data/information models, esp. IDEF<sub>0</sub> to IDEF<sub>11X</sub> (4); between vendor tools, even of the same model type (2); among whole suite of models including: environment, decision, organization, process, information, systems, goals, events and data (1).

1 No — How do you handle data that may be duplicated or inter-related between/among models?

The one "no" respondent is currently only performing data modeling and therefore does not have to deal with model integration at this time. Two other respondents, while marking "yes", noted that they were working with object-oriented modeling techniques and that they believed those techniques might mitigate the need for model integration as "object-oriented models are sort of already integrated". One respondent noted that they currently duplicate data in models "like crazy". Two respondents noted that they currently have to manually interpret and check models for duplication, and where there are duplications, they must ensure that the duplicated data is handled consistently.

6. When a model is developed by more than one modeler, is coordination addressed by a library or configuration management procedure that is an inherent part of the methodology, or by external procedures? (check one): 12 Part of methodology 12 External to the methodology

Several respondents using multiple CASE tools noted that each tool has its own configuration management methods and capabilities. Either one specific CASE tool must be chosen as the baseline and used for all configuration management of all models, or humans must perform integration and configuration management among the various CASE tools. One respondent explicitly mentioned the usefulness of the IDEF methodology's author/library cycle and "kits" for configuration management.

7. Once a model has been developed, does the model get used for purposes other than its originally-intended purpose? (check one):

1 Always 8 Frequently 8 Occasionally 7 Rarely 0 Never

8. Once a model has been developed, is it maintained to reflect changes in the real-world domain represented by the model? (check one): 8 Yes 5 Sometimes ("depends on value") 11 No



If "Yes", how is correspondence ensured and verified?

*Manually, through periodic review, re-evaluation, comparison, audit or other formal validation of models to the real-world or to implemented systems (4); using repository services or other automated configuration checking, including the IDS IRCS, Binder method/tools, Taylor method/SmallTalk tool and Knowledgeware Encyclopedia-based verification (4); through software Corrective Action Procedures including those specified in USAF 1104 and/or 1501, e.g. when the software is formally changed, models or specifications are also changed (3).*

9. How rigid or flexible is the application of most modeling methodologies utilized in your environment (e.g. is the "3-to-6 rule" always obeyed in IDEF models)? (check one):

2 Very Rigid 3 Somewhat Rigid 13 Somewhat Flexible 6 Very Flexible

*Vendors were the most adamant about rigidity, followed by consultants and academicians. Respondents from industry were almost universal in noting their application of methodologies as being somewhat or very flexible.*

10. Are existing models (whether produced internally or as part of government or industry programs) used as the basis for new models? (check one):

0 Always 16 Frequently 5 Occasionally 3 Rarely 1 Never

*Many respondents stating "occasionally" or "rarely" indicated that this was due to the lack of availability of existing models, not because of a lack of interest. One respondent stated that they has specifically used ICAM's MFGD.*

Would a library of subject area models which could be easily copied and customized be of use? (check one):

15 Definitely 6 Probably 3 Possibly 0 Doubtful 0 No

11. Which of the following modeling methodologies, techniques, approaches or languages do you currently utilize in-house? (check all that apply):

- 8 Binary and Elementary N-ary Relationship Modeling
- 4 Nijssen or Natural-language Information Analysis Methodology (NIAM)
- 4 Object-Role (Relation) Modeling
- 2 Precedence Temporal Modeling (PTM)
- 2 Semantic Association Model (SAM)/Object-oriented Semantic Association Model (OSAM)
- 0 Structures Inheritance Network (SI-Net)

12 Computer-Aided Software Engineering (CASE), including Pseudo-Code and/or Problem Statement Language(s) (PSL)

- 15 Entity-Attribute-Relationship (EAR) Modeling (other than IDEF1/IX as listed below)
- 2 Extended-Entity-Relationship (EER) Modeling (e.g. M\*, DATAID, BERM)
- 2 Bachman Object Role Modeling

- 7 Express/Express-G (please check here if only used in context of PDES/STEP efforts): 2
- 9 Flowcharting
- 7 Hierarchical Input-Process-Output (HIPO) Modeling
- 16 Integrated Computer-Aided Manufacturing (ICAM) DEFinition Methodology (IDEF)
  - 16 IDEF0 Functional Modeling Methodology
    - 12 IDEF0 Node Tree/Functional Hierarchy Modeling
    - 14 Full IDEF0 — Input, Control, Output, Mechanism (ICOM) Modeling
  - 2 IDEF1 Data Modeling Methodology
  - 11 IDEF1X Data Modeling Methodology
  - 0 IDEF2 System Dynamics/Simulation Modeling Methodology
  - 4 *SLAM (2), TESS (1) and SAINT (1) in lieu of IDEF2*

*Regarding IDEF3 and IDEF4, please answer the following (check if true):*

- 7 I have seen a proposed specification for IDEF3 (Process Flow/Object State Modeling)
- 2 Based on our review of the specification, we would consider using IDEF3 at our site, but are not using it currently.
- 1 IDEF3 is currently being used in our enterprise modeling environment.
  
- 5 I have seen a proposed specification for IDEF4 (Object-Oriented Design).
- 1 Based on our review of the specification, we would consider using IDEF4 at our site, but are not using it currently.
- 0 IDEF4 is currently being used in our enterprise modeling environment.
  
- 1 *"Never heard of either IDEF3 or IDEF4. Where can I get information?"*
- 5 Interpreted Predicate Logic (IPL)
  - 4 Conceptual Graphs
- 8 Petri Net/Critical Path Method (CPM)
- 5 Requirements-Driven Systems Engineering (including Requirements Allocation and Traceability)
- 5 Structured Analysis and Design Technique (SADT)
  - 3 Acti-grams
  - 2 Data-grams *"Rarely" (1)*
- 9 Yourdon
  - 9 Data Flow Diagrams (DFD)/"Bubble Charts"/"Spider Diagrams"
  - 4 Mini-specs
  - 7 Data Dictionary
- 7 Other (please specify):

*Extended IDEF<sub>0</sub> to denote state changes such as "notification", "approval", etc.(1); Gane and Sarson*

*DFD (1); Oracle CASE\* (1); home-grown "classification" tool (1); GE-Rumbaugh Object Management Technique (1) with Object-Maker; home-grown tool called SADDAM (1); IDE (1).*

12. Are there any modeling methodologies, techniques, approaches or languages which were utilized for projects within the last 2 years, but which are no longer utilized? (circle one):

3 Yes    21 No

If "Yes", which? (please identify the methodology(ies), technique(s), approach(es) or language(s) and indicate briefly the reasons why it is no longer utilized):

*Cadre's Teamwork CASE method and tool — too weak in area of data modeling (1); IEF — too costly, tool not UNIX-based (1); not specified (1).*

13. Are you currently using any automated software tools to assist with the creation, storage, retrieval and/or graphical presentation of enterprise models? (circle one):    17 Yes    5 No

If "Yes", which? (please identify the tool, the software vendor and the kind(s) of modeling [from 1. above] for which the tool is utilized):

*Knowledgeware's ADW/IEW, including Encyclopedia (5); Meta Software's Design/IDEF for IDEF<sub>0</sub> and IDEF<sub>1X</sub> (4); Oracle Corporation's CASE\* (3); home-grown (3); various unspecified CASE tools (3); Accelerator (3); Logicwork's Erwin for IDEF<sub>1X</sub> (2); Wizdom Systems' IDEFine 0 and/or 1X (2); Gane and Sarson's DFD (2); DEFT (2); Cadre's Teamwork (2); Janus/Leverage IDEF<sub>1X</sub> (2); ModelPro (2); DataPro (1); Mark IV Limited's Object-Maker with GE-Rumbaugh Object Management Technique (1) for objects, dynamics/control and functions (1); Smalltalk (1); Lansa 4GL (1); Yourdon (1); Control Data Corporation's PC-IAST for NIAM modeling (1); Bachman (1); Systems Architect CASE tool (1); Texas Instruments' CASE tool (1); IBM's IS-Mod for process modeling (1); QFD Matrices (1); Uniface (1); Interleaf (1); Sybase (1); Knowledge Consultant's GADRAN (1); Eclectic Solutions' COINS; unspecified IDEF tool (1).*

14. Are there any automated modeling tools which were utilized for projects within the last 2 years, but which are no longer utilized? (circle one):    6 Yes    16 No

If "Yes", which? (please identify the tool, the software vendor and the kind(s) of modeling [from 1. above] for which the tool was utilized, and indicate briefly the reasons why the tool is no longer utilized):

*Accelerator — poor model integration (1); Meta Software's Design/IDEF — no reason given (1); Cadre's Teamwork CASE tool — too weak in area of data modeling (1); IEF — tool not UNIX-based (1); unspecified CASE tool — required too much graphical layout (1); several unspecified tools — size of models were too limited.*

15. Is analysis or "re-engineering" of existing, or "legacy" databases and/or application programs part of your current enterprise modeling process? (circle one):    15 Yes    7 No

If "Yes", please describe the methodology used. If this capability is the result of using an

automated tool, please identify and briefly describe the tool:

*Manually, based on USAF ICAM "As Is" and "To Be" architectures, 1501 Systems Development Methodology (SDM) and/or IDS Software Engineering Methodology (SEM) (5); manually, no standard methodology used/ad hoc/"whatever is handy — we'll use it"(4); ANSI/SPARC 3-schema-based approach using mappings from Conceptual Schema to Internal Schemas of legacy databases (3); ; Bachman (2), including use for re-engineering of data design architecture into high-level logical elements; home-grown, including one based on object-oriented messaging techniques (2); DACOM's Leverage (1); Cadre's Teamwork (1).*

If "No", do you plan to make this part of your enterprise modeling process in the future? (please answer "yes" or "no" and explain). (circle one): 3 Yes 3 No

*"Yes" respondents stated that it is an important part of future architectures or systems development life-cycle methodologies. "No" respondents stated this is not a part of their current plans.*

16. Do logical models and/or formal schemas currently exist for your legacy databases? (circle one and comment): 11 Yes 11 No

*Comments from "yes" respondents: those that exist are only there as a result of re-engineering efforts to-date (3); mostly formal schemas, few logical models (1); yes, but existing logical model and schemas are inadequate to reflect the semantics of existing database contents (1); both logical models and formal schemas exist (1).*

17. What database management systems (e.g. IMS, DB2, Oracle, etc.) do you currently utilize?

*DB2 (13); Oracle (11); IMS (9); Ingres (5); Rdb (3); DMS (2); RAMIS (2); Foxpro Foxbase (2); Sybase (1); dBASE (1); 4th Dimension (2); TSO (1); Image (1); AS400 (1); unspecified SQL (1); home-grown object-oriented (1).*

18. Do you presently have a data directory or dictionary? (circle one; if "Yes", please specify):

15 Yes 9 No

*Comments from "yes" respondents: Home-grown — more passive directories rather than active dictionaries — one is built on 4th Dimension and is based on data from functional models, one migrating to CASE tool-based dictionary (6); Knowledgeware CASE Dictionary (1)/Knowledgeware Encyclopedia, including one coupled with Object-Maker Data Dictionary (2); CDM\*Plus (1); IDMS coupled with Cadre's Teamwork (1); DACOM's The Integration Kernel (1); IBM's A/D Cycle Repository Manager (1); IBM's CIM/CDF (1); USAF IDS IRCS (1); UCC-100 (1); Oracle's CASE\* Dictionary (1); in some cases - it depends on organizational environment (1).*

19. Is a formal systems development life-cycle methodology followed during software design, development and support? (circle one): 15 Yes 7 No

*One "yes" respondent specifically noted that although a life-cycle systems development methodology exists, it is inconsistently applied.*

If "Yes", is the software development methodology home-grown or one such as: the ICAM Systems Development Life-Cycle; the DoD Automated Data Systems Documentation Standards (DOD 7935.1-S); Software Engineering Institute (SEI) methodologies; or other similar industry-wide methodologies?

*Home-grown, including one based on the "spiral model" and object concepts, and another based on a number of industry-leading SDLC's such as ICAM (10); ICAM (2); MilStd 2167 (1); Binder SDM (1); SEI (1).*

20. How is configuration management performed across the software life-cycle?

*Home-grown or customized automated systems (5); "it isn't" or "poorly" (4); home-grown manual procedures (3); automated repository technology (2); automated version control, in one case coupled with a formal software release cycle (2); manual procedures based on ICAM SDM/CM (1); per MilStd's, but varies on a contract basis (1); check-in/check-out library system using PanValet (1); formal program and database definitions with simple versioning (1); SCLM (1); varies (1).*

21. How is correspondence maintained between:

a) Logical data model, physical database design and actual database schema?

*It isn't (10), comments: "it isn't in our IBM mainframe environment", "it isn't really — manually at best", "haven't been, but starting to happen"; automated repository technology/stored 3-schema mappings/data dictionary, including home-grown, Knowledgeware and Oracle CASE dictionaries (6); manually, using ICAM configuration management procedures, formal design documentation, structured walk-throughs and other internal database management and configuration management procedures (5); through software Corrective Action Procedures, e.g. when the actual database schema is formally changed, logical and physical models are changed, too (1).*

b) Logical process model (e.g. flowchart, system flow specification), programming specification and actual program code?

*It isn't (11), comments: "we go from formal requirements specifications to code, without formal programming specifications, etc."; automated repository technology/stored 3-schema mappings/data dictionary, including home-grown (2); through software Corrective Action Procedures, e.g. when the actual code is formally changed, logical process models and programming specifications are changed, too (2); structured analysis and testing, included integrated systems testing - in one case a Problem Definition Language (PDL), mini-specs, action-diagrams and implementation rules are all linked together (2); using IDEF3 specifications (1); "wishful thinking plus debugging" (1).*

22. Please prioritize the following criteria related to modeling for enterprise integration. Rank from 1st (highest importance) to 10th (lowest importance):

*Numbers shown are averages based on the rankings and number of responses for each criterium.*

Supports modeling of one aspect or a few aspects at a time; allows segregation of model elements (e.g. has a separate form or user interface for process flow versus control flow) (7.53)

Models of one element integrate with models of other elements (e.g. information models can be integrated with process flow models) (4.65)

Models translate easily into program code and/or database schemas (5.24)

Assists with re-engineering of legacy databases and/or application programs (5.27)

Supports multiple forms of modeling for the same aspect or element (e.g. information entities may be modeled graphically with EAR models or in text form using a PSL) (7.53)

Supports enterprise modeling in the context of an information framework or architecture and/or as part of a systems development life-cycle methodology (3.80)

Model captures lots of information/semantics (4.39)

Automated tools are available to assist with modeling (5.14)

Based on standards and/or commonly used in industry (6.14)

Models are easy to create and maintain (4.41)

*Expressed in order of highest to lowest priority, the criteria appear as follows:*

- 1 Supports enterprise modeling in the context of an information framework or architecture and/or as part of a systems development life-cycle methodology
- 2 Model captures lots of information/semantics
- 3 Models are easy to create and maintain
- 4 Models of one element integrate with models of other elements (e.g. information models can be integrated with process flow models)
- 5 Automated tools are available to assist with modeling
- 6 Models translate easily into program code and/or database schemas
- 7 Assists with re-engineering of legacy databases and/or application programs
- 8 Based on standards and/or commonly used in industry
- 9 Supports modeling of one aspect or a few aspects at a time; allows segregation of model elements (e.g. has a separate form or user interface for process flow versus control flow)
- 9 Supports multiple forms of modeling for the same aspect or element (e.g. information entities may be modeled graphically with EAR models or in text form using a PSL)

The highest-ranking criterium was ranked #1 more times than any other criterium (8 times). Its ranking was pulled down only slightly by a few mid-level rankings and a couple of low-level rankings. The second highest criterium varied from the highest by .59, marking a significantly clear break with the highest ranked criterium. #2 was fairly consistently ranked at 2 or 3. The third highest varied from the second by only a small fraction (.02), meaning they were effectively ranked the same.

The fourth and fifth ranked were reasonably well spaced in relation to the third ranked criterium, to one another, and to the sixth ranked criterium. They form a consistent middle or transitional level.

The sixth and seventh criteria varied by only .03, and like criteria #2 and #3, can be considered effectively equivalent in ranking. Internally, the rankings split into two groups in both cases. Apparently, either executable models and re-engineering are considered immediate "hot buttons" or have not yet identified as major aspects of enterprise analysis in support of enterprise integration.

The eighth ranked criterium was situated in its own statistical layer between #6 and #7 and the two criteria tied for last place. There was some variability in rankings with a few respondents considering this to be quite important. However, there was a strong grouping at the lower end of the ranking scale. The higher rankings were largely given by vendors and consultants.

The last two criteria, which tied for ninth place with exactly the same number of points, were consistently regarded as least important. The one concerned with multiple modeling forms for the same modeling element received the largest number of tenth place rankings. Since these two criteria were ranked the lowest, this would seem to indicate that most modelers are not concerned that models try to capture too much data at once (in one modeling form), and they are apparently satisfied to have one good modeling method and form for capturing a particular type of enterprise data (for example, E-A-R models for data modeling). However, some additional, somewhat different observations may be made regarding these last two criteria and their rankings. First, the finding that modelers are not particularly concerned about methods and forms which clearly segregate and limit the number of different types of data captured in a single method and form might arise from existing practice. Other survey results indicate that existing methods and forms are if anything too segregated, as there was a major need expressed for model integration. Also, more complex forms of enterprise modeling are only being performed in selected cases within selected enterprises. As these more advanced, deeper kinds of modeling methods and forms become more widely used, modelers may find them to be too bundled and may force the developer/vendors of these new methods and tools to segregate the information captured or presented into more workable "chunks", similar to the existing segregations between functional models, process models, data models and basic information models. Also, modelers might be satisfied with one method and form for capturing a particular type of data, but probably only if that method and form is the one they use in their enterprise. Since other survey results indicate that the choice of methods and forms for particular types of modeling varies greatly from one enterprise to another, tool developers/vendors will still be required to offer modelers a choice of several methods and forms for the same type of information. However, this is apparently a developer/vendor issue, not a modeler (as user) issue.

23. What is the single biggest problem with enterprise modeling in your current environment?

- Lacks top-management support, needs to be oriented more to business needs and language, mgmt. wants to know how modeling relates to productivity improvement and cost reduction (6)
- Lacks uniformity across the enterprise, inconsistently applied, need standardization (including

*standardization of modeling objects) (5)*

- *Not doing enough of it (3)*
- *Need more direct link between models and code, maintain the model, not the code (3)*
- *Lack of expert modelers, need better trained modelers, need to be trained in many different methods and tools (3)*
- *Too costly (2)*
- *Difficult to scope because the domain is too big, difficult to reach a consensus on the definition of enterprise modeling and a complete set of analysis goals (2)*
- *Current methods and tools don't support simulation and "what-if" analysis for decision-support (1)*
- *Takes too much time (1)*
- *Difficult to switch from traditional structured analysis methods to object-oriented approach (1)*
- *Requires a cultural/paradigm shift (1)*
- *Need better automated tools (1)*
- *Methods still need improvement (1)*



**SUPPLEMENTARY**

**INFORMATION**



DEPARTMENT OF THE AIR FORCE

WRIGHT LABORATORY (AFSC)  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6523

ERRATA

AD-A 255 547

ERRATA

3 Nov 92

REPLY TO  
ATTN OF DOOS (DSN: 785-5197)

SUBJECT: Notice of Changes in Technical Report(s) ~~AD-A255028~~ and ~~AD-A255547~~ (WL-TR-92-8027 and WL-TR-92-8048)

TO: Defense Technical Information Center  
ATTN: DTIC-FDR  
Cameron Station  
Alexandria VA 22304-6145


Please change subject report(s) as follows:


Please note that the contract # should be F33615-91-C-5722. On the DTIC fm 50s and SF 298/DD 1473 we showed F33615-90-C-5722 by mistake.

ERRATA:

AD-A 255 547

ERRATA

  
WM F. WHALEN  
Chief, Tech Editing & STINFO Branch  
Resource Management

  
AD-A 255 547