

AL-TP-1992-0052

AD-A261 088



# INTEGRATED DEVELOPMENT SUPPORT ENVIRONMENT (IDSE)

Richard J. Mayer  
Martha S. Wells

KNOWLEDGE BASED SYSTEMS LABORATORY  
DEPARTMENT OF INDUSTRIAL ENGINEERING  
TEXAS A&M UNIVERSITY  
COLLEGE STATION, TX 77843

Michael K. Painter, Capt, USAF

HUMAN RESOURCES DIRECTORATE  
LOGISTICS RESEARCH DIVISION

NOVEMBER 1992

INTERIM TECHNICAL PAPER FOR PERIOD JANUARY 1990 - MARCH 1991

Approved for public release; distribution is unlimited.

93 3 1 026

AIR FORCE MATERIEL COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

ARMSTRONG  
LABORATORY

93-04256



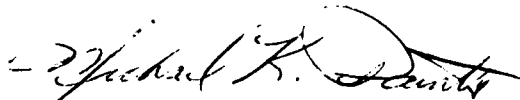
DTIC  
ELECTE  
MAR 02 1993  
S E D

## NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation, or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.



MICHAEL K. PAINTER, Capt, USAF  
Program Manager



BERTRAM W. CREAM, Chief  
Logistics Research Division

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>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.</small>				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE November 1992	3. REPORT TYPE AND DATES COVERED Interim - January 1990 to March 1991		
4. TITLE AND SUBTITLE Integrated Development Support Environment (IDSE)		5. FUNDING NUMBERS C - FQ7624-90-00010 PE - 63106F PR - 2940 TA - 01 WU - 15 <i>MEPR-</i>		
6. AUTHOR(S) Richard J. Mayer      Michael K. Painter, Capt, USAF Martha S. Wells				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Knowledge Based Systems Laboratory ✓ Department of Industrial Engineering Texas A&M University College Station, TX 77843		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Armstrong Laboratory Human Resources Directorate Logistics Research Division Wright-Patterson AFB, OH 45433-6573		10. SPONSORING/MONITORING AGENCY REPORT NUMBER AL-TP-1992-0052		
11. SUPPLEMENTARY NOTES Armstrong Laboratory Technical Monitor: Michael K. Painter, AL/HRGA, (513) 255-7775				
12a. DISTRIBUTION/AVAILABILITY STATEMENT  Approved for public release; distribution is unlimited		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) <p>Businesses today are rapidly recognizing that information assets are a resource that belongs to, and can be more effectively used by, the enterprise as a whole to realize major advancements in competitiveness. Attempts to leverage and reuse these assets, however, have long been plagued by the inability of our information systems to adapt and evolve gracefully to a changing environment. The ability to leverage corporate information assets therefore necessitates an altogether new way of thinking about, and developing, enterprise information systems. If information is to be controlled and managed as a global enterprise resource, then enterprise information systems will require the consistent, long-term involvement of large numbers of individuals, many of whom are not computer specialists. An Integrated Development Support Environment (IDSE) provides a system which supports user-driven <u>development</u> and <u>evolution</u> of information systems through graceful change to both (1) the information needs to be supported, and (2) the degree and types of automation. This report describes the design concepts for an IDSE centering around a <i>scalable</i> architectural design strategy enabling enterprise-level evaluation towards the desired levels of integration, automation, and sophistication.</p>				
14. SUBJECT TERMS architecture computer-aided systems engineering		continuous process improvement information engineering information systems	integrated information systems integration systems engineering	15. NUMBER OF PAGES 81
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR	

## Table of Contents

List of Figures.....	v
Preface .....	vi
Summary.....	vii
1.0 Introduction.....	1
2.0 Executive Overview.....	1
2.1 Motivation .....	2
2.2 Requirements.....	3
2.3 Key IDSE Architectural Needs and Concepts .....	5
2.3.1 IDSE Architectural Levels .....	6
2.4 Example Scenario of IDSE Application .....	12
3.0 IDSE Concepts .....	16
3.1 Basic Philosophy, Strategy, and Approach.....	16
3.2 Architectural Description of the Four IDSE Levels .....	17
3.2.1 Level 0 IDSE.....	17
3.2.1.1 User Interaction Services.....	20
3.2.1.2 Tool Sets and Data Integration.....	20
3.2.1.3 Minimum Artifact Repository.....	21
3.2.2 Level 1 IDSE.....	24
3.2.2.1 Increased Functionality of Level 0 Concepts in Level 1 .....	26
3.2.2.2 Level 1 Integration Services Concept.....	27
3.2.3 Level 2 IDSE.....	30
3.2.3.1 Object/Artifact Repository.....	32
3.2.3.2 ISyCL Loader and Reader .....	34
3.2.4 Level 3 IDSE.....	35
4.0 IDSE Usage and Application Scenario .....	38
5.0 Summary and Conclusions .....	49
5.1 Integrated Set of Development and Management Tools .....	50
5.2 Automated Support for Information Transfer .....	50
5.3 Management of the System Development Process.....	53
5.4 Management of Information System Design Artifacts.....	54
5.5 IDSE Internal Evolution Capabilities .....	55

<b>Bibliography.....</b>	<b>57</b>
--------------------------	-----------

## List of Figures

Figure 1	IDSE Architecture.....	7
Figure 2	IDSE Architectural Levels.....	9
Figure 3	IDSE Functionalities by Level.....	18
Figure 4	IDSE Level 0 Architecture .....	19
Figure 5	IDSE Level 1 Architecture .....	25
Figure 6	Integration Services Manager.....	28
Figure 7	Level 2 IDSE Architecture .....	31
Figure 8	Level 3 IDSE-Life Cycle Artifact Object Repository.....	36
Figure 9	Level 3 Information Transfer and Interpretation.....	37
Figure 10	Login Display for the IDSE.....	39
Figure 11	IDSE Options Available to a Modeler.....	40
Figure 12	IDEF3 Process Model for a Modeler Using a Level 0 IDSE .....	42
Figure 13	Check-In Procedure for IDSE Level 0.....	43
Figure 14	File Check-In Form.....	44
Figure 15	Level 0 Librarian Check-In Process .....	45
Figure 16	Requesting Data in a Level 0 IDSE .....	46
Figure 17	Check-Out Process.....	47
Figure 18	Browsing the Artifacts.....	48

DTIC QUALITY INSPECTED 1

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

## **Preface**

This paper describes the research accomplished at the Knowledge Based Systems Laboratory of the Department of Industrial Engineering at Texas A&M University. Funding for the Laboratory's research in Integrated Information System Development Methods and Tools has been provided by the Logistics Research Division of the Armstrong Laboratory (AL/HRG), Wright-Patterson Air Force Base, Ohio 45433, under the technical direction of USAF Captain Michael K. Painter, under subcontract with the NASA Research Institute for Computing and Information Systems (RICIS) Program at the University of Houston. The authors wish to acknowledge and extend a special thanks to the Integrated Development Support Environment (IDSE) design team whose names are listed below:

Keith A. Ackley

Thomas M. Blinn

Mark A. Brunsell

Louis P. Decker

Arthur A. Keen

Richard J. Mayer

Michael K. Painter, Captain, USAF

Martha S. Wells

## **Summary**

This report describes the basic functionality, architecture, and operational concepts of an Integrated Development Support Environment (IDSE). The requirement exists for a means of developing and maintaining information systems that allow the organization to control the system development life cycle. The primary focus of the IDSE research is to provide a system development environment that supports development and evolution of all artifacts of the information system life cycle from conception through retirement including component reusability, development process description, design artifact management and tracking, and user requirements. The IDSE will provide an environment for intelligent assistance in the design, development, and evolution of an enterprise information system through support for the following functions:

- 1) data transfer between tools and methods,
- 2) storage and retrieval of the system life-cycle data, information, and knowledge at both the object and artifact level,
- 3) communication between users of the system, and
- 4) version control and configuration management for the system life-cycle information.

The IDSE concept is structured around the idea of support sophistication levels. Each level represents a complete and viable development support environment, not a version of some other environment. Level 0 represents what we believe is a reasonable environment that would require little or no additional computer resources investment from of the implementing organization. Advancing from Level 0 to Level 3 will, with each step, provide increased integration support, increased automation for the system development process, increased tool sophistication, and correspondingly, will require increased financial and computer resources.



## **1.0 Introduction**

The Integrated Development Support Environment (IDSE) research was conducted as part of the Air Force Integrated Information Systems Evolution Environment (IISEE) program, which is focused on developing technology like theories, formalizations, frameworks, methods, automated tools, and environments for enabling or improving the process of planning, definition, development, and maintenance of evolutionary, integrated information systems. An IDSE is a system development environment consisting of:

1. system development tools,
2. a repository (storage area) of system development artifacts, and
3. integration services that provide automated support for system development efforts.

Other objectives of the IDSE research are to provide better support and tracking of information system evolution, better coordination between interdependent analysis and design activities, and up-front consideration of all life-cycle factors associated with a product. In other words, IDSE will be an automated environment that will support the functions and methods required for the definition, design, development, and maintenance of an evolving integrated information system. To control and manage this environment, procedures must be provided to control and manage the large amounts of data generated by the utilization of IDSE.

## **2.0 Executive Overview**

### **2.1 Motivation**

If the United States is to regain its leadership role in the world marketplace, industry needs a means of managing and controlling its information systems. Most organizations recognize this and acknowledge that information is second only to people in importance. However, development of a comprehensive information system that can control and manage the needs of an organization or even a product development effort is a complex process. Although information systems may vary greatly, many common characteristics are shared by all, including the following.

1. Many people are involved with the development of an information system, a significant number of whom are not computer specialists.

## *IDSE Concept of Operations*

2. The life cycles of information systems often extend beyond the working life of the original designers.
3. There are vast amounts of poorly maintained documentation.
4. The most costly phase in the life cycle is system maintenance.
5. The few automated tools that do exist address different phases of the life cycle and do not provide complete coverage or integrated support for the entire process. In the system development process, there are only "islands" of automated assistance; the data created in one phase must often be re-created if it is to be used in later phases.
6. The lessons learned and components created in one project are seldom carried over to others.
7. The existing project management tools do not address the problems of task assignment and accountability with any degree of completeness.
8. The systems produced often do not satisfy the customer's requirements.
9. No one person can understand all the details of a system because the size of the code in even one now involves hundreds of thousands of lines.

The requirement exists for a means of developing and maintaining information systems that allow the organization to control the system development life cycle. Furthermore, since the amount of information that the organization maintains is so large, the development of information systems is no longer a job performed by a few individuals. Developing an information system requires the coordinated effort of a large number of individuals, many of whom are not trained system development experts. A system development effort requires input from business managers, potential users, and customers as well as system designers, programmers, and technicians. What is needed is an information system development environment that provides the following.

1. A means to capture, represent, present, manipulate, and integrate information about the system definition and design, as well as the design of the process used to produce it and the modification history of the resulting artifact.
2. An integrated set of tools to be used by the individuals involved in the development process. The integration supported by these tools should enable a process of evolvable, tailorable, and universally automated tool integration. In the environment, integrating and accessing automated tools should be

## *IDSE Concept of Operations*

accomplished without extensive work. These tools must address all phases of the system life cycle.

3. A means for controlled sharing and tracking of design information. This will require design databases that allow the linking of design information and the interlinking of design information to requirements information.
4. A means for tracking design dependencies and change, as well as propagating their effects. The data storage facilities must provide for the linking of system needs and requirements to designs and design decisions.
5. A means for monitoring and controlling the development process.

The primary focus of the IDSE research is to provide a system development environment that supports development and evolution of all artifacts of the information system life cycle from conception through retirement, including component reusability, development process description, design artifact management and tracking, and user requirements.

## **2.2 Requirements**

If an IDSE is to enable development, management, and evolution of information systems, it must provide the following basic functionality.

1. **An integrated set of system development and management tools.** There are currently many non-integrated tools that address different aspects of the system development process.<sup>1</sup> Most of them adequately address the area of the system development process for which they were designed; however, they have been developed by different vendors and for the most part have different data and hardware requirements. An IDSE must provide a means whereby these tools can function together. Since the IDSE will be an evolving information system itself, it must provide the means whereby new tools and new versions of old tools can easily be incorporated into the environment without interfering with the execution of existing tools.
2. **Automated support for information transfer between tools.** This functionality must be provided by the IDSE because the tools will have different hardware

---

<sup>1</sup> There are many areas of the system development process that as yet have no automated support. Furthermore, in the area of project management and administration, the available tools are spectacularly inadequate.

### *IDSE Concept of Operations*

and software requirements. Furthermore, different tools will be addressing different aspects of the development of the same system, and the data produced by one will likely provide valuable input to another. Automated support for information transfer will shorten system development time.

3. **Management and control of the system development process.** For large, complex systems, the development process is equally complex. Generally, a great number of individuals, tools, and data are involved. An IDSE must provide a means for improved coordination between the various interdependent design activities, better support and tracking of system evolution, and maintenance considerations for all life-cycle artifacts associated with the development process. The implications of this requirement are that there must be (1) site-definable process definition (i.e., each site defines its own development process), (2) configuration management and version control for the target system, and (3) change control and notification mechanisms.

4. **Management of information system design artifacts.** IDSE primary functions are to improve system quality, reduce system development life-cycle cost, and shorten system development lead-times. Large, complex projects produce vast amounts of data. An IDSE must provide the means to describe and control it while providing for the creation and manipulation of data, access control for global data, management of system versions and configurations (software and hardware), and the management of repository backups and restores. The IDSE system must deal equally well with design, project management, and control data in a manner transparent to a user.

5. **IDSE administration.** An IDSE is expected to be an evolving information system. As such, it must provide for its own development, management, and evolution. This means that an IDSE must provide utilities that enable the storage, management, access, and manipulation of data dictionaries, user profiles, tool and workstation descriptions, control rules, and descriptions of other IDSE components and networks.

By meeting these requirements, IDSE will provide the automated environment needed to support the application of design procedures and methods for the creation and maintenance of an evolving system description. Most importantly, it will provide the capability to manage the repository for the evolving system description at both elemental object and artifact levels.

### **2.3 Key IDSE Architectural Needs and Concepts**

An IDSE must, by definition, be an integrated information system. The phrase "integrated information system" is used to refer to the information base and associated schemas which collectively model the Universe of Discourse<sup>2</sup> associated with an enterprise. An **evolving information system** is designed to allow for change over time and has mechanisms to facilitate those changes. Furthermore, an integrated information system must evolve from (and accommodate) existing traditional information systems. The IDSE will share components with the enterprise information system (EIS) it is supporting. For example, part of an EIS would be the integration mechanism that would allow the system changes to be integrated into the evolving system. A scaled version of this integration mechanism must also be a part of an IDSE, enabling the incorporation of new tools and allowing system development automation technology to be added seamlessly into an existing installation. In other words, the IDSE is an information system itself and will require utilities similar to the enterprise information systems it is being used to construct.

The integration mechanism within both the EIS and the IDSE must ensure that all users who request logically equivalent data get the same data even if it is referred to by different names. Moreover, the integration mechanism should ensure that modifications to a piece of information in the system always follow the same process and conform to all predefined organizational constraints on information creation and modification. An example of such a constraint might be, "No employee may be his own supervisor." This constraint affects not only employee assignment, creation, and modification but project supervisor creation and modification as well. The integration mechanism must provide the means to enforce<sup>3</sup> this constraint and others defined on the information system.

Future information systems are expected to have different hardware and software operating the system. However, future design changes in the internal storage of these integrated

---

<sup>2</sup> A selected portion (the enterprise) of the real world. For instance, a university, the degrees granted by it, and all instances of those degrees would represent a universe of discourse. An integrated information system contains a representation of the universe of discourse. A description of the universe of discourse can be called the enterprise description. (ISO/TR 9007, *Information Processing Systems - Concepts and Terminology for the Conceptual Schema and the Information Base*, 1987.)

<sup>3</sup> "Enforce" is most likely too strong a term to use in many cases. The information system might allow certain modifications but issue a notification of a business rule violation.

### *IDSE Concept of Operations*

systems should not affect how the user works with it. In other words, the evolving information system environment must also be an integrated environment.

#### **2.3.1 IDSE Architectural Levels**

An IDSE must provide a platform for the delivery of intelligent tools in addition to the management and control of the enterprise information system education. To provide an environment for intelligent assistance in the design, development, and evolution of an enterprise information system, the IDSE must support the following:

1. data transfer between tools and methods,
2. storage and retrieval of the system life-cycle data, information, and knowledge at both the object and artifact level,
3. communication between system users, and
4. version control and configuration management for the system life-cycle information.

It must also provide a scalable, integrated environment that allows integration, data transfer services, and tailoring of implementation site control over the development process.

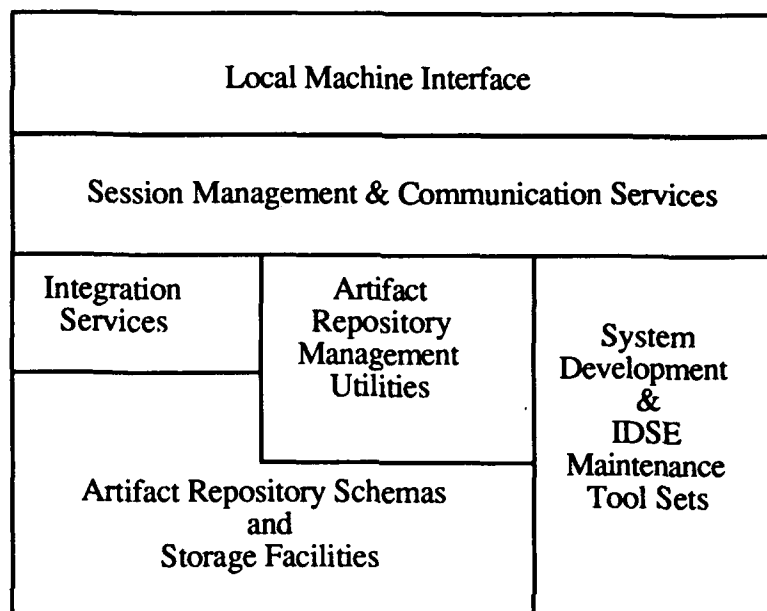
As conceptualized in Figure 1, an IDSE will consist of:

- Local Machine Interfaces,
- Session Management and Communication Services,
- Integration Services,
- Artifact Repository Schemas and Storage Facilities,
- Artifact Repository Management Utilities, and
- System Development and IDSE Maintenance Tool Sets.

Each computer within an IDSE implementation will have its own local interface to the system. Through this local interface, the user will connect to IDSE via a session manager, which will set up the user's work environment based upon his or her profile and the usage scenario for that session. The user profiles and usage scenarios are maintained by the System Definition Data. The session manager will have access to them via the repository management utilities. Because the object and artifact repositories are likely not to be completely stored on a single machine, the IDSE will provide networking capabilities by which remote users (programs and individuals) can communicate with these repositories and other users. The IDSE will support data and function integration through an

### *IDSE Concept of Operations*

integration services concept that requires minimal modifications to existing applications. This approach, described in detail in Section 3, IDSE Concepts, offers the potential to avoid many of the legacy system assimilation problems normally associated with the traditional International Organization for Standardization conceptual schema approach to integration.



**Figure 1. IDSE Architecture**

The life-cycle object/artifact repository provides for the storage and maintenance of the data, information, and knowledge associated with a system development effort. The complexity of the repository management utilities provided by the IDSE will be capable of tailoring to the support level required of the IDSE at a site. For a minimum IDSE, artifact repository management will be supported. To allow intelligent application of the integration services, repository management will be extended to include configuration control information and data evolution rules. Finally, to provide design rationale capture support and automated model interpretation and translation, support for object-level manipulation must be provided. In other words, objects (such as model elements), in addition to files, must be able to be manipulated. In general, the repository management utilities include those that allow users to obtain data from the repository and to insert objects/artifacts into it. Furthermore, the management facilities will provide for (1) data security by having encryption and decryption utilities, (2) version management and change

### *IDSE Concept of Operations*

control for design and other life-cycle data, (3) message management utilities, and (4) various other utilities required by any database management system.

An IDSE would not be complete without a set of automated system development support tools, not necessarily built by the developers of the IDSE. In general, these will be commercial, off-the-shelf (COTS) tools that provide automated support for some phase of a system development effort, from problem identification and design to system maintenance. This set of tools includes those for the development and maintenance of the IDSE itself.

Integration is not obtained without cost. The IDSE philosophy is that integration should be demand-driven, meaning that integration services would be provided only when there are sufficient needs and resources to accomplish integration. A design goal of the IDSE is to provide an architecture that can be scaled to facilitate the desired level of integration. The resources required to achieve an integration level must be justified by the value of the increased capabilities of personnel provided by it. In other words, the IDSE philosophy is to treat integration as a resource.

To provide for tailorability, we have developed a "level" concept for the packaging of the IDSE functionality. This concept defines four levels of IDSE technology (see Figure 2) which are not IDSE versions. Each higher level of the IDSE will have all of the functionality of the lower levels as well as additional integration capabilities. When developed, each level will be a fully functional system and may, in the course of its lifetime, exist in different versions.

The components featured at each IDSE level are illustrated in Figure 2, which also shows a comparison of the functionality provided at each level. The level at which an organization would implement depends on the following factors:

- 1) current computer resources,
- 2) size of the enterprise,
- 3) extent of the system development needs, and
- 4) implementation budget.

The Level 0 IDSE would offer the implementing organization the least expensive level of IDSE support. Moreover, it would require the smallest investment in both computer and networking resources. In fact, the goal is for a Level 0 implementation to be implemented with the current computer resources of the organization. For a small organization with limited system development needs, Level 0 could be the highest level ever implemented.



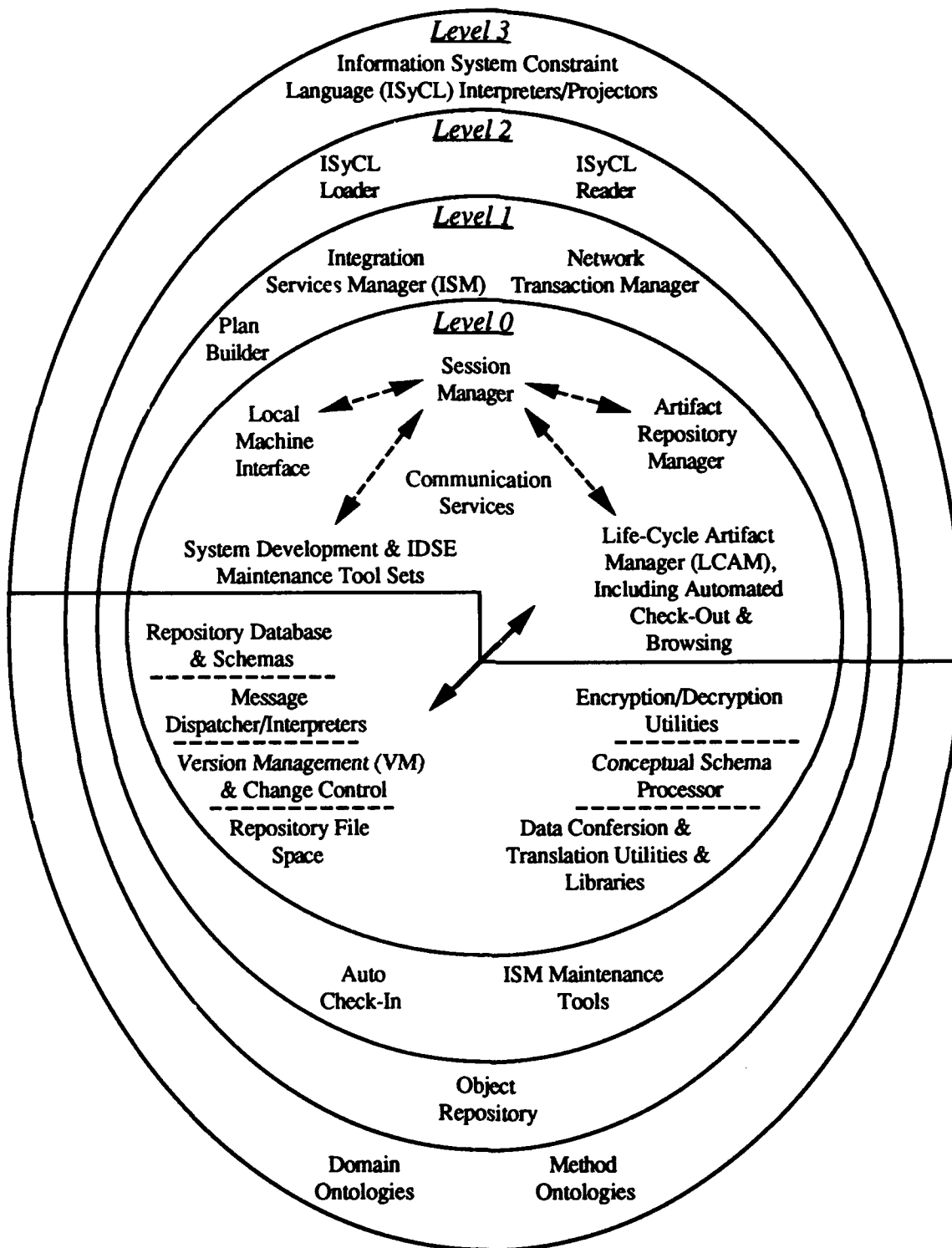


Figure 2. IDSE Architectural Levels

### *IDSE Concept of Operations*

In a Level 0 implementation, the IDSE would support an expandable set of COTS Computer-Aided Systems Engineering (CASE) and computer-aided software engineering (case) tools with outputs integrated via data converters and translation utilities. Implemented as a multi-schema architecture, it would support a minimal artifact repository (see Section 3.4). Associated with this level would be browsing capabilities and automated check-out functionality which would allow artifact use to be monitored and managed. At this level, the integration approach may be called the "automation support for manual control" because enforcement of check-in policies and constraints would rely heavily upon a human librarian to enforce the system development policies. Furthermore, the users would be much more actively involved in the data access and translation process.

A larger organization having a bigger budget and greater system development needs would want to implement one of the more advanced levels of IDSE. However, even with a larger organization, implementation of a Level 0 IDSE prior to one of the more advanced levels would make the transition to an integrated development environment less of a "future shock" to the affected system designers.

With each level, the extent of automated support for integration and control of the development process increases. A Level 1 implementation focuses primarily on automated support and coherence of data, function, and process integration. One of the primary capabilities provided by a Level 1 implementation is automated support for the check-in process, including automatic notification.<sup>4</sup> Enhanced networking capabilities will be required of the computing platform for a Level 1 implementation, which still provides all the manually exercised integration functionality of Level 0. However, integration will be enhanced with the inclusion of a component called the Integration Services Manager (ISM), which will allow the establishment of a Client/Server relationship between two tools and thereby provide for both data and function integration. The ISM will have access to a service list (advertised services) provided by the IDSE tools. When a user/tool (client) requires data created with another tool, an integration service request would be passed to the ISM. If this service is one of a list of services known to the ISM, then it would invoke the process(es) needed to provide the requested data. This process may be complicated by such issues as having to invoke remotely one or more tools to execute the request. If it is satisfied, the data would be transferred back to the client for use.

---

<sup>4</sup> Data check-in requires more advanced networking capabilities and more intelligence on the part of the processor performing the task. The activities associated with check-in are difficult and expensive to perform automatically.

### *IDSE Concept of Operations*

IDSE Level 2 implementation focuses on application-independent management of the life-cycle data. Integration will be enhanced at Level 2 with the inclusion of a Neutral Information Representation Scheme (NIRS) that enables information from models in various methods to be represented in a neutral format. With the inclusion of the NIRS, the IDSE can maintain and manage an Evolving System Description (ESD) at an object rather than an artifact level. It is expected that the NIRS would be implemented using the Information Systems Constraint Language (ISyCL). More information about ISyCL is in the ISyCL Technical Report [Decker 92]. At Level 2, the minimal artifact repository would be replaced with an Object Repository. A repository at this level would store the actual objects created in the system development process rather than just pointers to where artifacts containing those objects are stored. For example, specific requirement statements rather than a version of the requirements document would be managed. Each IDSE "Level 2 friendly" tool would have an ISyCL extractor that would convert models into an ISyCL-formatted NIRS representation. The ISM would have an additional utility (called an ISyCL Loader) which would place data that is in ISyCL format into the object repository. An ISyCL Reader will be provided to translate client-requested information from the object repository. To support the Loader and Reader, tool vendors must provide utilities that can translate the data their tools create into ISyCL code and interpret ISyCL data files.

Level 3 IDSE implementations would provide the highest integration level (semantic interpretation of model-acquired data). The object repository in Level 3 would provide a facility for interpreting the meaning of the various model element instances (and complete models) submitted to the Object repository. This functionality will be provided via the use of method and domain ontologies, which are knowledge bases containing the meta-level descriptions of objects, relationships, and constraints pertaining to a particular domain. For each domain or method of interest, there must be a separate ontology. In Level 3, a method ontology would be used in conjunction with a domain ontology to guide information transfer from a model into the object repository through the use of the ISyCL interpreter for that method and particular domain ontology. The ontologies (domain and method) could then be used (with an ISyCL projector) to take information found in the object repository and generate a partial model in a different method. With these additions, the site implementing a Level 3 IDSE will support automatic model translation, provide data consistency across the entire system development process, and maintain traceability links between data artifacts.

## **2.4 Example Scenario of IDSE Application**

Just like any information system, the IDSE will be an evolving system. It is expected that IDSE technology application would be implemented at a site in at least four stages (corresponding to the implementation of the four levels described above). The four levels do not represent versions; rather, each is a fully functional production system. With each stage in the assimilation process, the introduction of a new level would provide an organization with increased automated assistance in their system definition, development, or maintenance activities. The following scenario provides a concept of how an imaginary chemical company called Forefront Chemical, Inc., would likely implement this IDSE technology.

Consider a typical situation for the employees of the Forefront Chemical, Inc., Information and Systems Group. As one of the largest U.S. chemical firms, Forefront is a forerunner in not only chemical technology but also in information systems technology. Where other companies are buckling under the weight of their information systems, Forefront is actually decreasing query times and most importantly, decreasing costs. What put Forefront ahead of its competitors? It has successfully leveraged technology developed under the U.S. Air Force IISEE project.

Of course, there was a great deal of grumbling at first. Information systems are terrible beasts to manage. When the keepers of the beasts heard of plans to implement an entirely new system for evolution of the enterprise information systems, the obvious questions were raised. What is wrong with the current system? How can we afford to mess around with such an integral part of our business? Where are we going to find the resources to accomplish such a project?

Luckily, corporate management had already seen the proverbial handwriting on the wall. They knew that information was their second most critical resource behind its people. They also knew that the quantity of information to be handled was increasing far too rapidly for their current systems to evolve without major overhauls. Intense world market competition was forcing quicker responses to market demands and causing profit margins to decrease while manpower and raw material costs kept increasing. The current systems were founded on the following outdated principles: computing costs were expensive; computer systems were centralized; storage was limited; and the company had the resources to develop custom applications.

### *IDSE Concept of Operations*

No one can argue that information systems must be handled gently. Cut off the information flow and the company stops dead in its tracks. The Technology Planning Group had assured management that the new IDSE could be brought on-line independent of the EIISs, incrementally improving portions of the systems as needed. A thorough security plan had been developed to protect the information systems from corruption.

Once the decision had been made to implement IDSE, the grumbling stopped since no one had time to worry about it. The Forefront developers took the specification provided by the researchers and broke into teams. One concentrated on IDSE implementation strategies, and the other took on the task of training the company engineers and management in function modeling (IDEF0) and process description capture (IDEF3). A third team set out to bring current information models up to date and construct new enterprise models. Yet another team worked with corporate management to determine policies regarding project planning. For instance, no project received funding without a complete activity model. There were still other teams that interfaced with industry standards organizations and consortiums.

Then the changes began to appear. As one walked past a conference room, the presenter would usually be referring to an IDEF0 or IDEF3 model of some task. Portable computers had become so inexpensive and powerful that conference tables included network connections to the corporation model databases. At lunch, discussions of how particular constraints had been identified could be heard. Even weekly progress reports had become more organized since project members could refer to their tasks in the project framework.

The IDSE was yet to be completed, though. The models were still not integrated. Human librarians still checked models in and out and oversaw the evolution of system artifacts. Users needed to be notified of what services were available since the automated services manager was still under development. Tools were not integrated because they were still being enhanced to support remote invocation and cooperation with the IDSE Artifact Manager. Common information between models still had to be transferred and maintained by hand. The advent of Dynamic Data Exchange (DDE) on Personal Computers (PCs) and "Hot Links" on Macintosh computers had shown people what they could expect from their modeling tools. Consequently, there was some grumbling when models had to be modified independently. Still, all in all, the system was gaining speed, and corporate management began to see dividends from their investment.

A while later, the IDSE Implementation Group brought on-line what they called the "Level One" implementation of the IDSE. It was revolutionary. People no longer needed to be experts to get around the system. The International Organization for Standardization (ISO)

## *IDSE Concept of Operations*

provided transparent access to models no matter where they were. Check-in and check-out of models was handled automatically. No one had to wait for a reply from the system librarian in order to obtain artifacts. Either they were available, or the name and mailbox of the present holder were returned. Of course, the corporation voice mail system became more widely used because people no longer had to use "SneakerNet" to hunt down models.

The Integration Services Planner (ISP) was now working in the background, automatically generating plans when services were requested. Users no longer needed to remember the chain of steps required either to translate data from one format to another or to extract useful parts of a model for use in another. For instance, if an individual wanted the list of concepts from an IDEF0 model for use in an IDEF1 model, all the user needed to supply was the name of the model. The ISP determined the steps necessary to obtain the IDEF0 model, invoke the tool to extract the concepts (the IDEF0 tool had this capability), move the concept list to the user's directory, and invoke the IDEF1 tool to read the concept list and allow the user to specify how to incorporate the concepts into the IDEF1 model. System gurus still liked to show that they knew how to do it on their own, but management frowned upon those who made manual errors.

The model integration became automated, and the ISM now allowed various forms of it. The simplest form was called "Get and Forget." In other words, a portion of one model was copied to another, but the connection was lost. Thus, if changes were made to one, they would not be reflected in the other. ISO also provided "Get and Remember." In this case, the links were remembered, and changes in one model were reflected in the other. Certain tools that had not been modified to exist within the IDSE could not provide this capability. They were either enhanced by their vendors or left by the wayside.

The response to the move to Level One caught the IDSE Implementation Group by surprise. They were not prepared for the demands users placed on the system. Since people now depended on the ISM to perform tasks unaided, they spent more of their time on modeling and work and less on learning how to use the system. People became spoiled quickly. If the system failed to do something the user felt possible, a service request was filed immediately (voice mail, of course). The IDSE Implementation Group was forced to reduce its efforts on advancing the state of the IDSE and concentrate on supplying the services the users identified as essential to their effective completion of tasks. Eventually, the requests leveled off and new technology levels were addressed.

What more could one ask? The enterprise information systems were already radically improved over the former implementations. Still, there were a lot of people who spent time trying to incorporate the new models into information systems implementations. Since the

### *IDSE Concept of Operations*

tools for modeling allowed the users to model so much more quickly, the workload on the information system implementors went through the roof. There needed to be a better method for incorporating model changes into the Evolving System Description (the collection of models used to implement the enterprise information systems).

The IDSE Implementation Group was already hard at work on this problem with standards organizations and tool vendors to iron out the standard for the ISyCL that would act as the Neutral Information Representation structure of the IDSE. Once the standard was defined, tool vendors incorporated ISyCL code generation and parsing into their tools. Furthermore, researchers had for some time been looking into expert systems for translating between models in different methods. ISyCL allowed many of the concepts discovered in the research to be implemented in the IDSE.

Parallel with this effort, a group was formed to define ontologies (using IDEF5) for various enterprise parts so that models with common elements could be identified automatically. This effort was very difficult since a company the size of Forefront has so many people with different backgrounds and word usages. As many conflicts as possible were resolved, and overall interdepartmental communication was improved greatly.

Just recently, Level Two went on-line. It did not have nearly the impact on users as Level One, but the corporation is seeing drastic improvements in its information quality. No longer are there discrepancies between overlapping models; confusing terminology has been reduced and the information systems implementations are more stable now that ISyCL descriptions are used to generate the actual database representations. No longer is there fear that the information systems will not be able to keep up. It is common to fear the unknown, but now that the IDSE provides such a clear view of the information structure, fears can be assuaged quickly.

There are still advances to be made. It is rumored that soon it will be possible to use tools as information system views since expert systems will be available to translate between models in different methods. Thus, the system will identify which concepts in an IDEF0 model are also represented as entity classes in an IDEF1 model, and so forth.

Though future advances are always appreciated, most people are just happy to perform their jobs more effectively. Forefront Chemicals, Inc., is effectively competing in a tough world market.

It is expected that most organizations would assimilate the IDSE technology in a manner similar to that depicted in this scenario. Even if it were currently available, few organizations are likely to introduce IDSE technology with a Level 2 or 3 implementation

## *IDSE Concept of Operations*

because cost-effective application at these levels requires cultural changes (just as the implementation of a Manufacturing Resource Planning (MRP) system requires in a manufacturing environment). We anticipate that IDSE assimilation will start at Level 0 or Level 1 because of the lower implementation cost and the transfer of an automated development environment that is palatable to the users. Furthermore, many smaller organizations will very likely not require the more advanced functionality. In the remainder of this report, we will examine the various IDSE concepts in greater detail.

### **3.0 IDSE Concepts**

#### **3.1 Basic Philosophy, Strategy, and Approach**

The primary IDSE goal is to provide automated support for information system developers which can be applied in any industry. Normally, information system development is considered a task performed by data processing and computer specialists, but this is no longer true. Enterprises are rapidly recognizing that the information they maintain is a resource belonging to the enterprise as a whole. As such, it is not the domain of, nor should it be controlled by, any particular department. If the information is to be controlled and managed as a global enterprise resource, then the development and evolution of the enterprise information systems will require many individuals who are not computer specialists. Thus, the IDSE must support multidisciplined development teams with varying degrees of computer skills.

To support these multidisciplined system development teams, an IDSE must provide an environment containing a larger number of automated tools that address issues from data collection and analysis to system maintenance. It must also support the automated transfer of information developed by one group of individuals to a form usable by other individuals working on other phases of the development.

This section will address those IDSE concepts of particular importance in satisfying the five basic requirements first listed in Section 2.2. These requirements include the following:

1. an integrated set of system development and management tools,
2. automated support for information transfer between tools,
3. management and control of the system development process,
4. management of information system design artifacts, and



## **5. IDSE administration.**

These requirements can be addressed by an environment that provides at least a minimal Life-Cycle Artifact Repository (LCAR), which is a facility that provides for the insertion, storage, management, and linkage of the life-cycle artifacts (data items) developed or used in a project. It will include not only the database(s) that hold the artifacts, but also the computer programs and utilities necessary to provide this functionality. Such an LCAR capability is central to each IDSE level. The sophisticated functionality of Level 2 and Level 3 requires the augmentation of the LCAR with an object-level manager. This combination will be called the Life-Cycle Object Repository (LCOR) and will provide the needed storage and management utilities necessary for implementing control at the atomic-object level (rather than the document level) for the massive amounts of information generated in a system development effort.

We will begin our discussion of IDSE concepts with a presentation of the conceptual architectures of each of the four IDSE levels, followed by discussions of the integration concepts to be implemented in the IDSE. These concepts include 1) the IDSE implementation of complex objects, 2) the interrelated notions of version management, configuration management, and change control, 3) model development support, and 4) the provision for user views and roles. Finally, we will cover tool requirements and a system development framework. Throughout these discussions, we wish to emphasize that many of these concepts will be implemented through the functionality provided in the LCAR or LCOR.

## **3.2 Architectural Description of the Four IDSE Levels**

This section will explain the architectures of the four IDSE levels. Each one provides increased integration support and automation for the system development process. The different degrees of integration functionality provided at each level of the IDSE are shown in Figure 3.

### **3.2.1 Level 0 IDSE**

The architecture of an IDSE<sup>5</sup> Level 0 implementation addresses each IDSE requirement, but in many aspects it does so by following a stringent set of site-specific manual procedures.

---

<sup>5</sup> It is not expected that any basic characteristics of Level 0 will change, even in later versions

## *IDSE Concept of Operations*

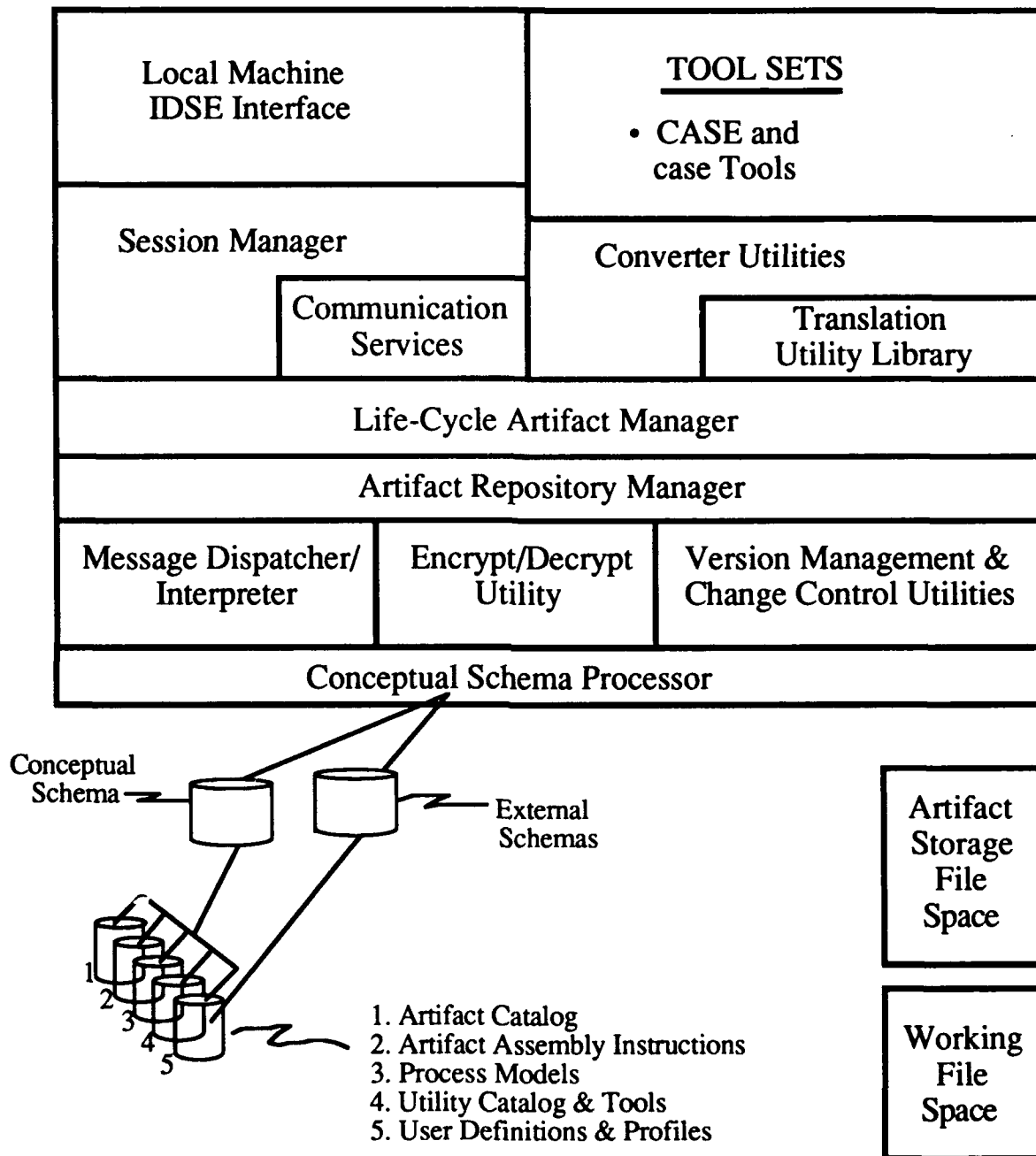
For this reason, we say that it provides primarily a manual approach to integration. Another way of looking at Level 0 is that it requires the minimum investment in computer and financial resources by the enterprise in which the technology is implemented.

Level 0	Level 1	Level 2	Level 3
1. Device-Independent User Interface.	All features of a Level 0 IDSE but with increased functionality (i.e., more automated). New features found in a Level 1 IDSE are:	All features and functionality of Levels 0 and 1. New/improved features in Level 2 are:	All features and functionality of Levels 0, 1, and 2. New/improved features in Level 2 are:
2. Session Management Facilities.	1. Integration Services Manager.	1. Object Repository is included, providing facilities for storing an ESD.	1. Domain Ontology Knowledge Bases.
3. Communication Services.	2. Integration Services Plan Builder.	2. Provision for a Neutral Information Representation Scheme (NIRS) via the inclusion of ISyCL.	2. Method Ontology Knowledge Bases.
4. System Development & IDSE Maintenance Tool Sets.	3. Plan Executor.	3. ISyCL Loader for inclusion of ISyCL representation in the ESD.	3. ISyCL Interpreter for translating ISyCL representations of system models into the the ESD using the method and domain-specific ontology knowledge bases.
5. Artifact Repository Manager.	4. Integration Services maintenance tools included in the IDSE set of maintenance tools.	4. ISyCL Reader for the automatic translation of information in the ESD into a particular method.	4. ISyCL Projector for translating ISyCL representations in the ESD into partial or complete models using the method and domain-specific ontology knowledge bases.
6. LCA Manager, including automated check-out & browsing.	5. Network Transaction Manager.	5. Increases in automation provided in the area of Version/Configuration Management & Change Control.	
7. Repository Database & Schemas.	6. LCA Manager will have automated check-in because of the increased network capability.		
8. Message Dispatcher/Interpreters.			
9. Encryption/Decryption Utilities.			
10. Version Management & Change Control.			
11. Conceptual Schema Processor.			
12. Repository File Space.			
13. Data Conversion, Translation Utilities & Libraries.			

**Figure 3. IDSE Functionalities by Level**

## IDSE Concept of Operations

Specifically, the architecture of Level 0 meets the IDSE requirements with the features presented Figure 4.



**Figure 4. IDSE Level 0 Architecture**

### **3.2.1.1 User Interaction Services**

The Local Machine Interface, the Session Manager, and the Communication Services of a Level 0 implementation provide and manage the user interaction services. Each machine will contain a device-independent IDSE interface. Regardless of the machine type, it will appear the same, and from it the user will access the session management facilities and other services under the control of a session manager, who will set up the user's work environment based on the user profile and usage scenario for the session. The session manager provides the user with access to the commands executed by the Life-Cycle Artifact Manager (LCAM) or system librarian. Level 0 will provide rudimentary communication services that will not be extensive but will provide for data-level integration and allow some communication among the user, artifact repository, and system librarian.

### **3.2.1.2 Tool Sets and Data Integration**

One IDSE requirement is providing "an integrated set of system development and management tools" that will consist of both CASE (Computer-Aided System Engineering) and case (computer-aided software engineering) tools. Upper-CASE tools are automated tools that support methods, documentation, or decision procedures in the early stages of the system life cycle (normally, the needs analysis stage through the detailed design stage). Project planning, management, and control tools are normally considered upper-CASE tools. Meanwhile, lower-case tools support the construction, testing, integration, maintenance, or reverse engineering of the software itself. In this document, we use the acronym CaSe when we want to refer to both upper-CASE and lower-case technology. Some of these tools will be developed with the IDSE. However, most will be vendor-supplied. An underlying assumption of IDSE design is that these tools will be operating on multiple, heterogeneous hardware and database platforms. In a Level 0 installation, there is no assumption that the tools have been designed by the vendors against any open-architecture standard, nor that they are integrated in any manner.

A Level 0 IDSE implementation will provide basic life-cycle data integration support as well as facilities for communication between the tools and the artifact repository. To provide basic integration capability, Level 0 will provide for data transfer between tools via the Converter Utilities and the Translation Utility Library (see Figure 4). These utilities support integration between these largely closed architecture tools at a file-transfer level. The strategy pursued at Level 0 is to encourage the tool vendors to adopt ASCII file model-

data-exchange conventions and provide utilities with tools that read and produce such file versions of their tool data. The National Institute of Standards and Technology (NIST) Interim Graphics Exchange Specification (IGES) initiative has successfully employed a similar strategy for data exchange between heterogeneous Computer-Aided Design (CAD) systems. For instance, a developer of an IDEF0 tool could choose to produce a text file of the objects that form relations between activities in an IDEF0 model (called concepts). Another vendor could then modify its tool to accept as input a text file of IDEF0 concepts. For example, an Evolving Natural Language Information Model (ENALIM) tool vendor may modify its tool to accept text file input of IDEF0 concepts. The Translation Utility Library IDSE component (Figure 4) would maintain a list of such data providers along with the type (and format) of the data that they provide (e.g., the IDEF0 of Vendor 1 can produce model IDEF0 objects in the standard model data exchange format). Moreover, the Translation Utility Library would maintain information on the available consumers for such data. That is, it would contain information such as, "the IDEF1X tool from Vendor 2 can consume IDEF0 concepts provided in the standard model data exchange format." The Converter Utilities will provide the requester with the instructions necessary to perform the required translations.

### **3.2.1.3 Minimum Artifact Repository**

A Level 0 (as well as a Level 1) implementation would include a minimum artifact repository. We use the term "minimum artifact repository" because the repository Artifact Catalog database will actually contain only pointers to the objects created in the system development process. The actual objects themselves will be stored in a file space reserved for the IDSE. Models and documents will be stored in the reserved file space as nondivisible entities. For example, storage will be provided for a requirements document but possibly not for individual requirements. Similarly, a model will be maintained as a single object, not as individual model elements. Along with other identifying information, the pointer objects maintained in the Artifact Catalog will contain a brief description of the artifact to which they point. This description will be available for the user who browses the repository.

The LCAM is responsible for providing the artifact-browsing and check-out functions. It is also responsible for execution of user commands and requests pertaining to the artifact repository. In a minimum LCAR, browsing the repository will mean that the user will actually be looking at brief descriptions of the artifacts stored in the repository file space. The descriptions will contain information about the version of the artifact and a description

## *IDSE Concept of Operations*

of the data it contains, allowing the user who browses the artifacts in the repository to collect information about the artifact without ever looking at it. If additional details are required as the user browses, the artifact would have to be *checked-out* and the appropriate tool invoked to examine the artifact (model or document).

The check-out operation is automated in the sense that it does not require intervention by the librarian. In the Level 0 LCAM version, the check-out procedure would result in a record being written to a audit-trail file/database. This record is for keeping track of who is modifying or reading the file. Furthermore, it will be used by the librarian to confirm that the user who checks out a file actually has the proper authority. In a Level 0 implementation, check-out will not necessarily result in file transfer across the network. Although the LCAM will be capable of handling the browsing and check-out functions, check-in is a much more complicated procedure.

When an object is checked in, there are completeness checks that must be performed. In some cases, they may require several days to complete. For instance, if an IDEF0 model is to be checked in for review, it must first pass a review process. Moreover, the check-in process will also likely invoke the version management and configuration control process. In some cases, the version of the object must be changed or the system configuration will have to be modified by the object being checked in. In a Level 0 implementation, the integration and communication services will not be robust enough to manage automatic check-in, which will be a manual procedure handled by a project or site librarian. The Level 0 LCAM would include the functionality necessary to support this project librarian activity. For example, Level 0 would provide support for a local site to define and manage its own system development process models maintained in the repository database. The project librarian would be able to obtain the procedures for document/model check-in from these process models. In a Level 1 IDSE, automated check-in would be supported from these same process models.

The Artifact Repository Manager (ARM) provides the management functions required to maintain the artifact repository, such as backup and retrieval utilities. It also contains access functions to the Message Dispatcher/Interpreter utility, Encrypt/Decrypt utility, Version Management, Change Control Utilities, and IDSE maintenance utilities. The Message Dispatcher/Interpreter would accept user messages (via the Session Manager) and place them in the appropriate message log for processing. This message log is accessible by the project librarian. The Encrypt/Decrypt Utility provides the librarian with the utilities to provide security from unauthorized file modification. These utilities have the necessary functionality to encrypt the names and store them as hidden files in repository-controlled

directories. Version Management and Change Control Utilities contain the functionality the ARM would require to properly manage the object and artifact version control. These utilities would also manage changes/updates that must be made to previously approved designs or models. The human librarian could access these utilities from the LCAM for manually executing the versioning and change control procedures defined in the project process model. Furthermore, those functions necessary to add tool types and other new types to the artifact repository will also exist within the ARM. Finally, the ARM contains the utilities necessary for capturing the IDSE Level 0 Conceptual Schema.

The repository databases include the Artifact Catalog, Artifact Assembly Instructions, etc. (Figure 4). The Artifact Catalog contains the pointers to the system development objects stored in the Artifact Storage File Space. For example, if the object pointer in the artifact catalog is to a model, the following information would represent the model object.

- Name: Name of the model.
- Type: Method used to construct the model.
- Location: Path name to the directory in which the model is stored.
- Tool vendor: Name of the tool used to construct the model.
- Tool machine: Machine on which the tool runs.
- Tool version: Version of the tool used to construct the model.
- Description: Brief description of the model (2 or 3 lines).
- Project: Project for which the model was constructed.
- Author: Chief author of the model, i.e., the creator.
- Version: Version of the model.
- State: Model state, i.e., in design, released, etc.
- Date created: Date the model was first saved.
- Date last modified: Last date on which the model was checked out for modification.

The Artifact Assembly Instructions will be procedures used to construct complex artifacts not explicitly stored. One example is a report consisting of several different documents and models. The artifact assembly instructions would contain the procedure to be followed to gather the required information and construct the report. In the Level 0 implementation, this assembly would not be automated. Instead, the user will be given a required file list and told how to construct the report.

### *IDSE Concept of Operations*

The Utility Catalog and tools database will maintain information about the various tools and utilities available in the IDSE. The information type maintained in this database will be about the machines on which the various tools will run and the latest version of those tools.

The user definitions and profiles are maintained in the User Definitions Database. These user definitions identify the legal users of the system and the roles they can play. The user roles identify for the session manager which set of commands, and in general what information, a particular user is entitled to access. In higher-level implementations, user roles will be used to interpret the meanings of statements or commands a user enters. Utilities in the utilities catalog will be provided to modify the user definitions and profiles. These will be used by the session manager to determine user access. Moreover, the conceptual schema processor will insure that data access and control are properly enforced and that data integration and conversion will occur as expected.

Finally, associated with a minimum artifact repository will be artifact storage file space and working file space. The artifact storage file space will be used for secure storage of the life-cycle artifacts, which include such objects as database files and flat files of the models created by the IDSE tools. Any documents created in the course of a system development (e.g., needs and requirements documents, management reports, etc.) are also managed in the artifact storage. The working file space is used for storing mail files and message logs, which will be maintained for the messages sent by the librarian to members of the development team and other parties. Temporary files used to manage such objects as models or documents awaiting review (or some other check-in activity) will also be maintained in the working file space.

Perhaps the most important Level 0 aspect is that it allows for multiple vendor tools on multiple hardware platforms. Furthermore, it supplies a minimum amount of tool integration with the translator utilities. There is provision for increased system development automation, and the Level 0 IDSE is extensible (new tools can be added). Level 0 does provide for increased control over a system development effort; however, because the primary control over the development is in the hands of the project librarian, it is referred to as a manual form of integration.

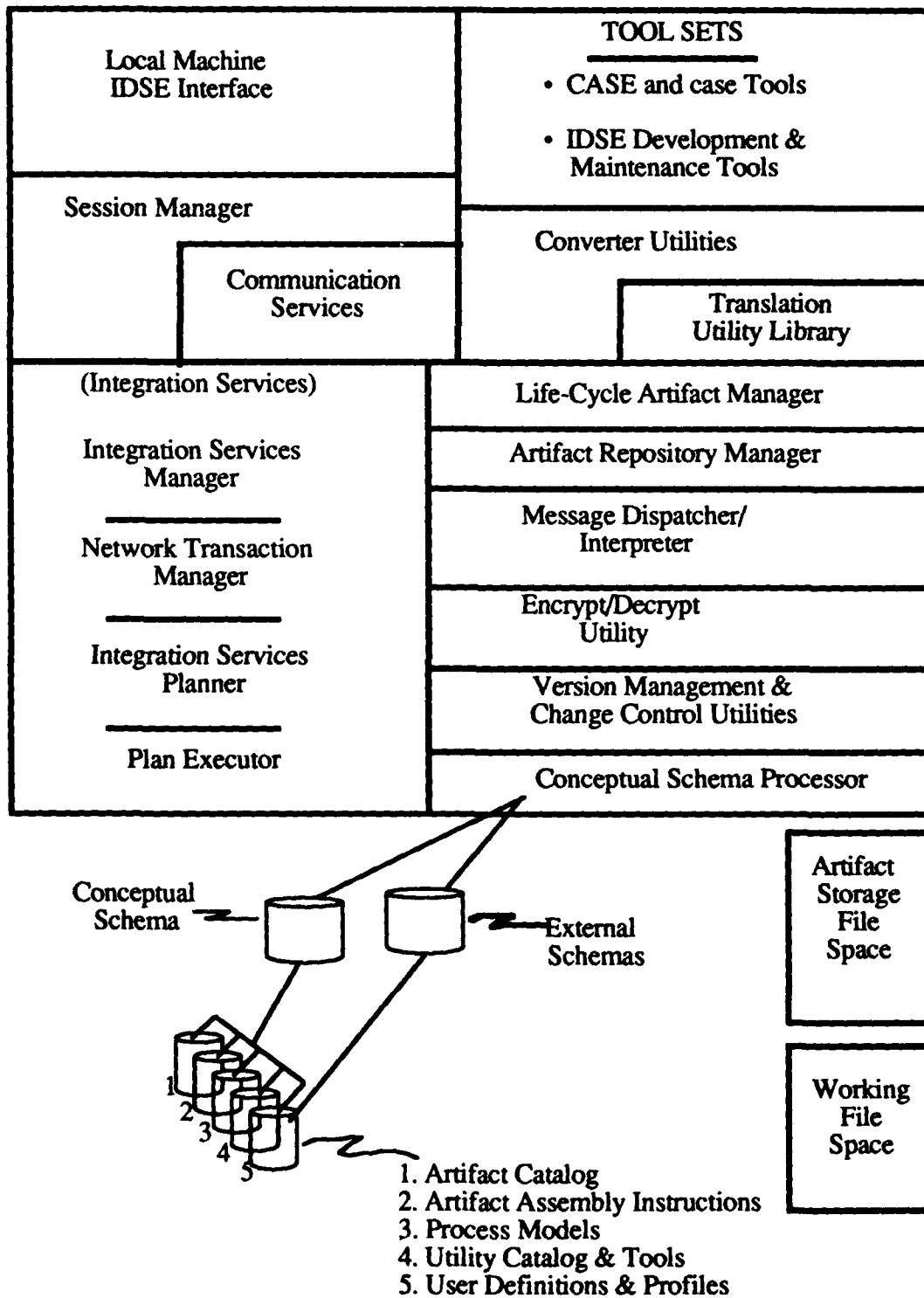
#### **3.2.2 Level 1 IDSE**

Those sites that implement a Level 1 IDSE will gain increased integration capability while still retaining all of the functionality of Level 0. In a Level 1 implementation, support will be provided for both data and function integration through the establishment of a



## IDSE Concept of Operations

client/server relationship between tools. Furthermore, Level 1 will provide more robust communication services. Figure 5 shows the architecture of the Level 1 IDSE.



**Figure 5. IDSE Level 1 Architecture**

## *IDSE Concept of Operations*

Compared to Level 0, Level 1 IDSE has the following:

1. All of the functionality of Level 0:
  - device-independent user interface,
  - Session Management facilities,
  - Translation Requirement Server and the Translation Utility Library,
  - Artifact Repository Manager,
  - Encrypt/Decrypt Utility,
  - Version Management and Change Control Utilities,
  - The Conceptual Schema Processor,
  - repository database and schemas, and
  - Artifact Repository file space.
2. Increased functionality in:
  - Communication Services,
  - Message Dispatcher/Interpreter,
  - Tool Sets and
  - Life Cycle Artifact Manager.
3. New functionality in integration services.

### **3.2.2.1 Increased Functionality of Level 0 Concepts in Level 1**

For most notable Level 0 IDSE users, the Level 1 change is that the LCAM will automatically handle the notifications and check-in tasks of the life-cycle data management and control. In Level 0, the LCAM provided browsing and check-out, but check-in was handled as a message to or some other communication form with the project librarian. With the improved network/communication services provided with Level 1, the check-in process is an automatic procedure. In Level 1, an IDSE will provide automated enforcement of the design/development policies and rules of the organization or project. This means that the LCAM will have to interpret a project development process defined in the process model. Although the process models must be understandable by the human, in Level 1, it will be possible for the LCAM utilities to determine the procedures to follow in a project by processing the process models. The Level 1 LCAM will follow the procedure outlined in the process schema (the stored version of the process model) by interpreting the

project process model just as the human librarian did in Level 0. For example, in the case of a model check-in-for-review, it would electronically ship the model to the reviewers and then wait for the comments to return. After a predetermined period of time, the LCAM will begin to query the reviewers for the return of their reviews. This ability for (1) long term transactions, (2) model based processing of user requests, and (3) the automatic notification facility is not available in the Level 0 IDSE. This additional functionality is provided by the Message Dispatcher/Interpreter and LCAM in Level 1.

### **3.2.2.2 Level 1 Integration Services Concept**

One of the notable aspects of the various IDSE technology levels is that each one offers more advanced integration while continuing to offer all the functionality available at the lower levels. In Level 1, the integration increase is offered via the Integration Services Manager (ISM) and Integration Services Planner (ISP).

The ISM will be responsible for automating much of the manual integration provided by the Converter Utility in Level 0. A motivating concept behind ISM is that an installation should not be forced into some preconceived notion of what future information integration needs will be. The ISM will be a flexible, evolvable approach to provide information integration for an IDSE.

The ISM will allow the IDSE to focus on providing the integration of tools<sup>6</sup> and the information created by them by establishing a client/server relationship between two of them. In a system development effort, information created in each stage will be required by later stages. Generally, it is true that the different developer tools do not require all the information created by other tools to perform their downstream functions. However, if the tool user had access to some of the data created by a previous task, it would decrease the amount of time required for the current task. With today's technology, this data often has to be re-created or at least reentered. With a services-based approach to integration, this redundant data entry would, over time, be eliminated.

One of the design goals of the services-based approach is to minimize the modifications vendors will have to make to their tools in order to accommodate the integration needs of a particular site. By logging the services required at a site, a vendor can provide just those

---

<sup>6</sup> The word "tool" is used to refer to any of a number of automated system development modeling method programs or automated project management aids.

## IDSE Concept of Operations

required services to be responsive to the needs of that site. Since most tools address different areas of the development process, only a portion of the data created by a tool will be needed by tools used in other project phases. The ISM will know what services are provided by each IDSE tool and how to invoke them. For instance, when building an IDEF1 model, the modeler might want to use the concepts/relations from an associated IDEF0 model as input to the IDEF1 modeling activity. In a Level 1 IDSE, the user of the IDEF1 tool should be able to request the IDEF0 concepts from the desired model. The ISM would then determine if such a service is available. If not, it would invoke the ISP to determine if a set of services could be assembled into a plan to accomplish the desired results. Such a plan might call for remotely invoking the tool that created the IDEF0 model, loading the model, and creating a file that contains the desired data.

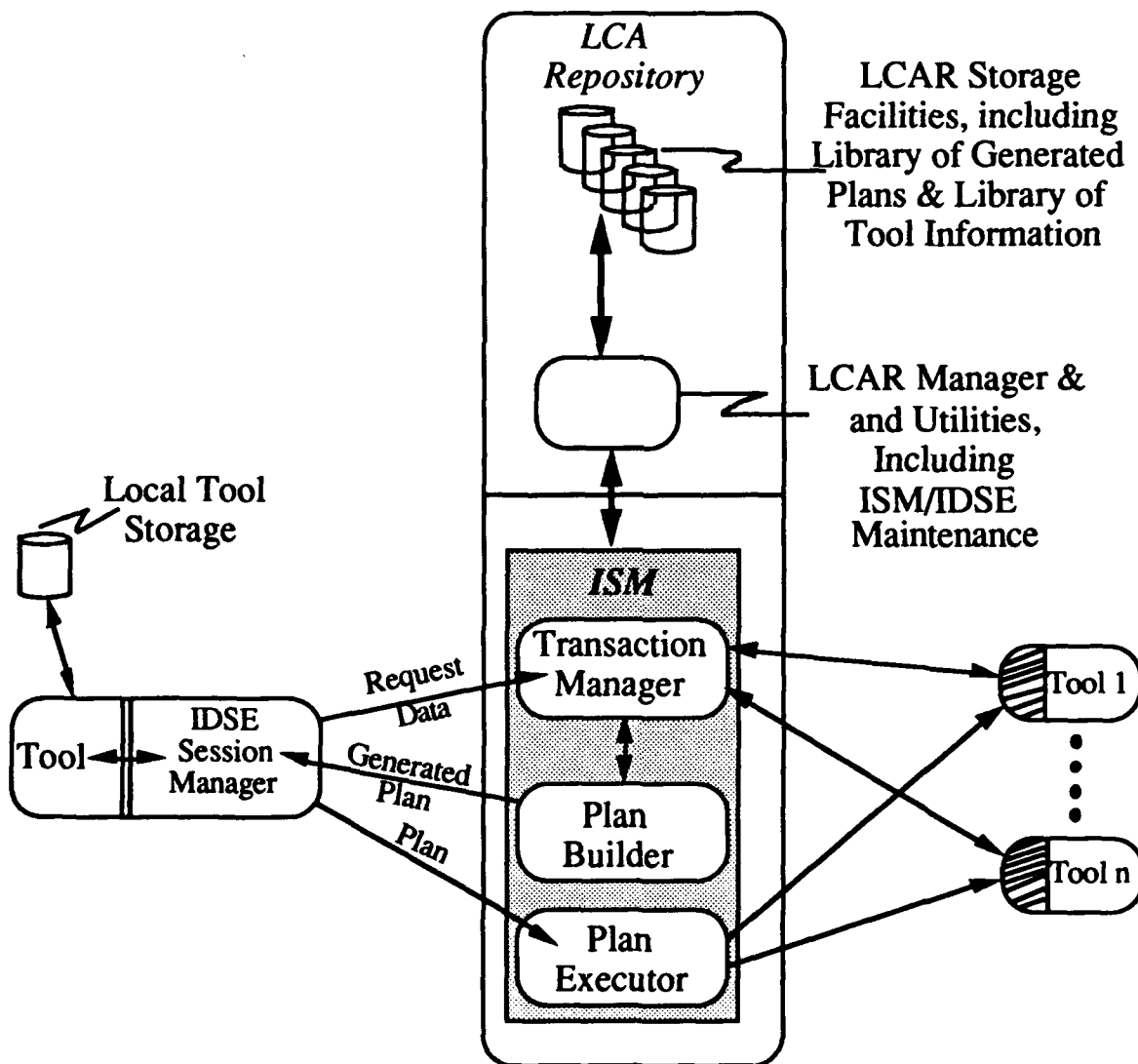


Figure 6. Integration Services Manager

Figure 6 shows IDSE utilities (beyond the ISM/ISP components) that will be required to provide these enhanced integration services. These utilities include Network Transaction Manager, Plan Builder, and Plan Executor. In addition to these utilities, IDSE integration services will also include ISM maintenance tools, a library of pregenerated plans, and a library of tool information. The ISM maintenance tools will be the utilities provided to allow IDSE sites to extend the integration services that they provide. These utilities will be used to maintain a library of service plans and the database of service advertisements. The library of plans catalogues frequently used service plans at the site. The database of service information consists of the information about the data or function services available in the system. It will include such information as tool identifier, tool vendor, tool version and type, and the format of the data either required or produced. The library and the database of advertised services will be used by the ISM to provide requested services.

A data or function service request will be communicated via the Session Manager and the Network Transaction Manager to the ISM, which will interpret the integration request and pass it to the Plan Builder. It will be the Plan Builder's responsibility to determine if the user's request can be satisfied. Quite often, the service will be as simple as a data format translation. In such a case, all the planner would have to do is match the request format against that of the list of available services. However, there may not be a simple match, in which case the planner will attempt to construct a sequence of translations using the available services. Once a valid plan for the service request has been built or extracted from the library, the plan is passed to the Plan Executor, who will execute the plan and return the results to the requestor. If for any reason (such as inability to access one of the required tools in the plan) the plan cannot be executed, it will be passed back to the user with an explanation. For example, not all the tools at a site will be "attached"<sup>7</sup> to the IDSE. In the case of an "unattached" tool, the ISM would determine that a particular tool produces some of the data required to execute a plan but would be unable to initiate remotely the execution of that tool. In this case, that step of the plan would be passed to the user with instructions describing the actions to take to execute that step. The user will perform the necessary actions and then instruct the Plan Executor to continue.

---

<sup>7</sup> Tools may be either attached or unattached. Attached tools are those that have been modified by the tool vendor to exist in the IDSE's integrated environment and to take advantage of all of the integration services available. Unattached tools may be used in the system development effort but are not available from within the integration services environment.

### *IDSE Concept of Operations*

This services approach to integration is dependent on (1) service advertisements (advertisement), (2) service protocol specifications, and (3) service contract specifications. Special utilities are needed in the Level 1 IDSE to capture this data. A service advertisement is a high-level description of services provided by the individual IDSE tools. An example is "that a particular tool will convert some of its data into a form suitable for use by another tool." The protocol for a service provides all the information needed by the Plan Executor to execute a tool remotely, and if necessary, to provide the advertised service, which contains such information as what version must be used, the invocation requirements of that tool, and the execution environment requirements for that tool. The contract specification will contain the data format and the content description of both the input to and the output from a service. The IDSE Level 1 will contain utilities for registering new services with the ISM and for describing and storing these three types of specifications in the artifact repository. The three languages for expressing these specifications will be discussed in more detail in Section 3.3.3.

A Level 1 implementation of the IDSE provides the advanced integration features of the Integration Services Manager without losing any of the Level 0 functionality. Moreover, the more robust communication/ networking facilities available to a Level 1 IDSE enables enhanced control over the development process through automation of many of the configuration control functions described in the Level 0 discussion above.

#### **3.2.3 Level 2 IDSE**

The Level 2 IDSE is focused on providing advanced support in the following areas:

- object-level management of the Evolving System Description,
- improved Version Management and Change Control Utilities, and
- increased set of integrated tools.

In Figure 7, we see that two new components have been added in the Level 2 architecture to provide this advanced functionality: the ISyCL Loader and ISyCL Reader.

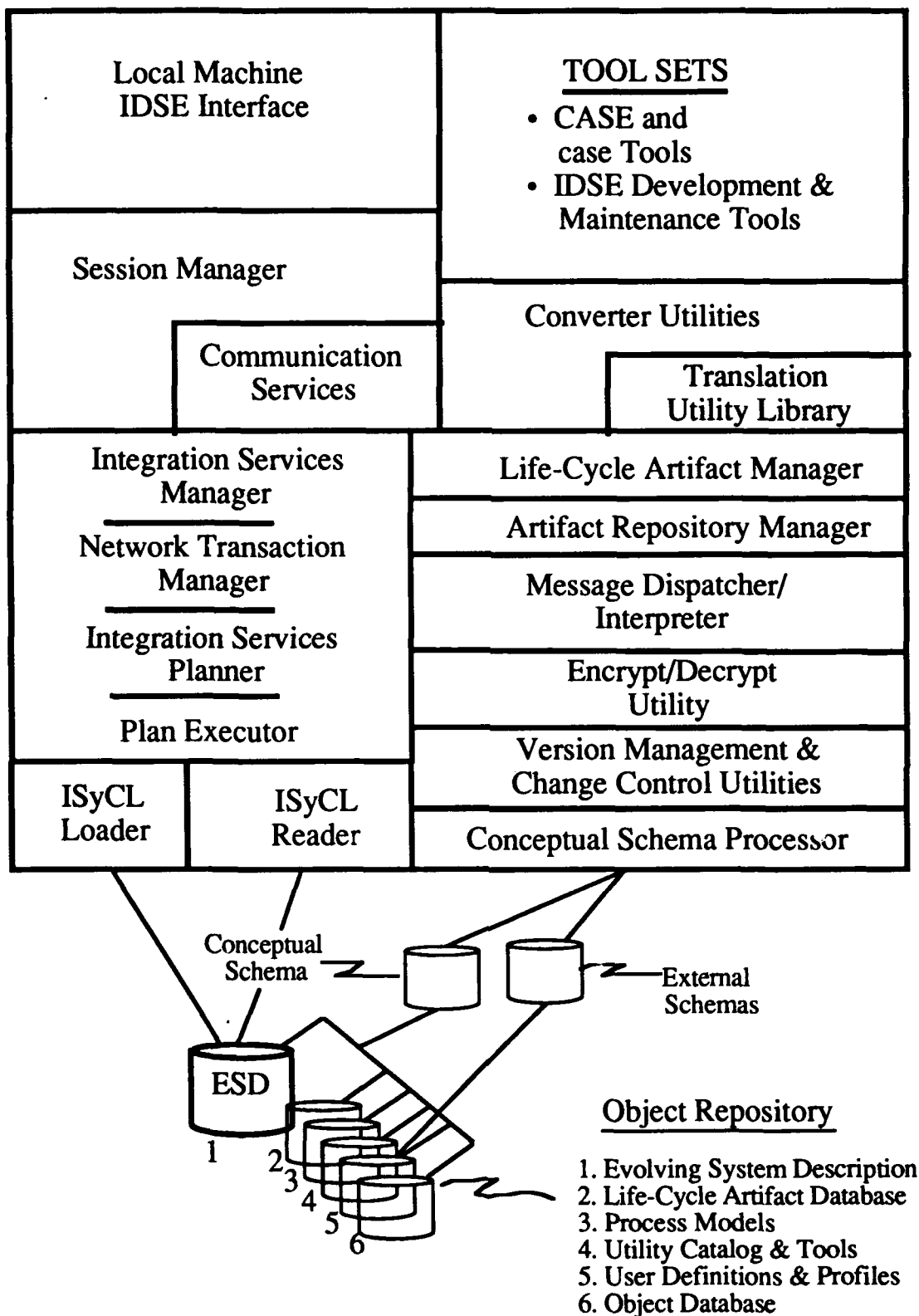


Figure 7. Level 2 IDSE Architecture

### **3.2.3.1 Object/Artifact Repository**

In Levels 0 and 1, IDSE was supported by an Artifact Repository, but with a Level 2 implementation, the capabilities of data management changes granularity from files to objects. The Level 2 IDSE LCAM and ARM will manage a repository that contains the actual information objects created in the system development process. It will be responsible for storing and maintaining the individual information objects making up the artifacts that are created in the development effort. Thus, a Level 2 IDSE would manage the individual model elements of approved/released<sup>8</sup> system models. It will also manage such items as individual requirements statements, design rationale statements, etc. Such an object repository would support better control, sharing, and analysis of these items. Site-specific change control policies can be implemented, which will provide the automatic propagation of requirements and design changes. In this environment, when a requirement is modified, the system will be able to notify the user immediately of the other design objects affected. Finally, the Level 2 repository will provide for a more automated approach to the difficult issues of version control and configuration management. The inclusion of ISyCL descriptions will be the primary reason this object representation and tighter system development control will be possible.

ISyCL has been designed to meet the definitional and knowledge representation needs of system design, from physical data description (low level) to the requirements of an analyst (moderately detailed) and the description of the business rules (high level).<sup>9</sup> ISyCL provides an object-centered approach to model elements and provides facilities for attaching constraints to them. It is designed to support all types of individuals involved in a system design effort, from the area expert to the software designer. ISyCL is a family of languages. To address the variety of system development needs, ISyCL is organized into layers and slices. Each layer is designed to support one of the types of individuals who would be involved in an information system development effort. Furthermore, because it is one language, constructs expressed in one of the higher-level layers can be easily expressed in lower layers. Expressive power increases as one moves down through the

---

<sup>8</sup> "Approved" and "released" are terms used to indicate the state of a model. The meaning of these terms will be entirely defined by the site that implements the IDSE.

<sup>9</sup> Decker, Louis P., and Mayer, Richard J., ISyCL Technical Report, AL-TP-1992-0019, Armstrong Laboratory, September 1992.



layers at the cost of expressive clarity. Each slice is designed to address a particular method, and slice is composed of a number of the previously described layers.

The current ISyCL design has identified four layers, each one addressing a different but vital group of individuals in an enterprise that will influence the information system design.

**1. Area Expert Layer:** The individual that this layer is designed for is focused on ensuring that the information represented in the particular information system model is consistent with his or her expectations.

**2. Analyst Layer:** The analyst would use this layer to attach constraints to the models and model elements which would have otherwise been expressed as text. In the system development process, graphical modeling methods are used to describe various system aspects. These languages derive much of their power from the fact that they provide a graphical picture of some aspect of the system. To maintain readability, they must limit the amount of information in a diagram. Thus, some of the constraints discovered during the model development cannot be expressed in the method syntax. This information is generally expressed in the text associated with the model. The analyst layer provides a means of capturing these additional constraints and linking them to the various model elements in a manner that can be interpreted by both humans and computers.

**3. Information Systems Design Layer:** This layer is designed to facilitate the transition of system models into the actual implementation of an information system. Since the implementation of an information system will be largely influenced by programming issues, this layer of ISyCL provides a more "programming language" type of syntax. It is expected that the information systems designer will use the information obtained by the analyst and implement the actual information system using this layer.

**4. Method Formalization Layer:** This is a special layer used for defining the "slices" of ISyCL that support the different constructs required to support the various modeling methods associated with the system design process. Each of the "slices" defines the constructs required by a given modeling method, one per method.

The primary functional addition of the Level 2 IDSE will be the provision of support for the capture and management of fine-grained information about the Evolving System Description. The Object Repository will store and maintain the individual development

artifacts (such as model elements) in this description. ISyCL provides the neutral representation language for storing these artifacts.

### **3.2.3.2 ISyCL Loader and Reader**

A Level 2 IDSE implementation will have a set of ISyCL Loaders and Readers that enable the construction of the object-based artifact repository. These utilities are considered a part of the integration services provided by the IDSE. An ISyCL Loader translates an ISyCL data file created by a tool into the ESD. The ISyCL Reader produces an ISyCL representation of a model from the data stored in the ESD.

Each tool that supports Level 2 must be able to generate an ISyCL representation of models created with that tool. It must also support the attachment of ISyCL constraints to models. In other words, a text editor must be supplied in which constraints can be written which will be appended to the ISyCL representation of a produced model. The tool is not required to do anything with these constraints other than append them to the ISyCL dump.

Due to the ISM and ISP, the typical user will likely never worry about creation of ISyCL representations. The user would simply direct the ISM to save a model to the ESD. Given that the user has the authority to modify the ESD, the ISM would invoke the tool to produce the ISyCL representation and then direct the ISyCL Loader to modify the ESD. The only ISyCL statements the modeler will see are the constraints he or she creates in the model construction process.

It is important to keep in mind the ISyCL layers. If the individual creating the model is not an analyst, then the constraints added to the model will be the English-like, area expert layer constraints. It is likely that an analyst would review a model created by an area expert and elaborate on the constraints (using the analyst layer of ISyCL). The analyst would probably then need to submit the model for release approval. Finally, whoever was given the necessary authority would save the model to the ESD. If the ESD was going to be used to enhance the information system, then a systems programmer would become involved. He or she would start with the analyst layer ISyCL representation of the necessary portion of the ESD and add the necessary detail to create a systems layer ISyCL representation that could be compiled to produce the schemas and control logic for the information system using the enterprise's Database Management System (DBMS).

The ISyCL Loader will place the model objects into the ESD. In order for a second tool to access this data, an ISyCL Reader must be provided to interpret/translate the neutral description of the ESD into information that is usable and understandable by that tool. This

### *IDSE Concept of Operations*

use of the ISyCL and the Loader and Reader will eliminate the need for tool format translators that were a part of Level 0. However, it will also mean that all the “attached” tools within the Level 2 implementation must generate and parse ISyCL model representations constructed using the method they automate.

#### **3.2.4 Level 3 IDSE**

The fourth IDSE implementation level (Level 3) is built on the same architecture as Level 2, with some minor modifications (Figure 8).

The other IDSE levels provide facilities that enable model elements to be stored, examined, and transferred between various tools. On the other hand, they do not have facilities for interpreting the meaning of the model elements. The Level 3 IDSE implementation provides this functionality through the use of Domain and Method Ontologies. With the inclusion of this functionality, the Level 3 IDSE technology provides support for (1) automatic model translation, (2) data consistency across the entire system development process, and (3) maintenance of traceability links between data artifacts.

An ontology, simply stated, is the theory of “what there is” in a particular domain. It is a knowledge base that contains information about the concepts, relationships, and situations in a domain. In particular, it contains the terminology information used to describe these concepts, relationships, and situations. As seen in Figure 8, there are two types of ontologies needed for the Level 3 IDSE: Method Ontologies and Domain Ontologies. A Method Ontology is a meta-level description of the objects, relationships, and constraints pertaining to a particular method. For example, an IDEF1 Method Ontology would contain objects such as Entity Classes, Attribute Classes, and Link Classes. It would also capture IDEF1 constraints such as the no-repeat and no-null rules. There must be a Method Ontology for each method supported by the Level 3 IDSE. A Domain Ontology is a meta-level description of the universe of discourse relating to a specific domain. For example, if one is producing automobiles, the Domain Ontology might include such objects as Engine Blocks, Transmissions, and Steering Wheels, as well as descriptions of the relationships between them. These ontologies will be used to generate interpretations of model objects, from one modeling method used under the IDSE to facilitating automatic generation of another model type.

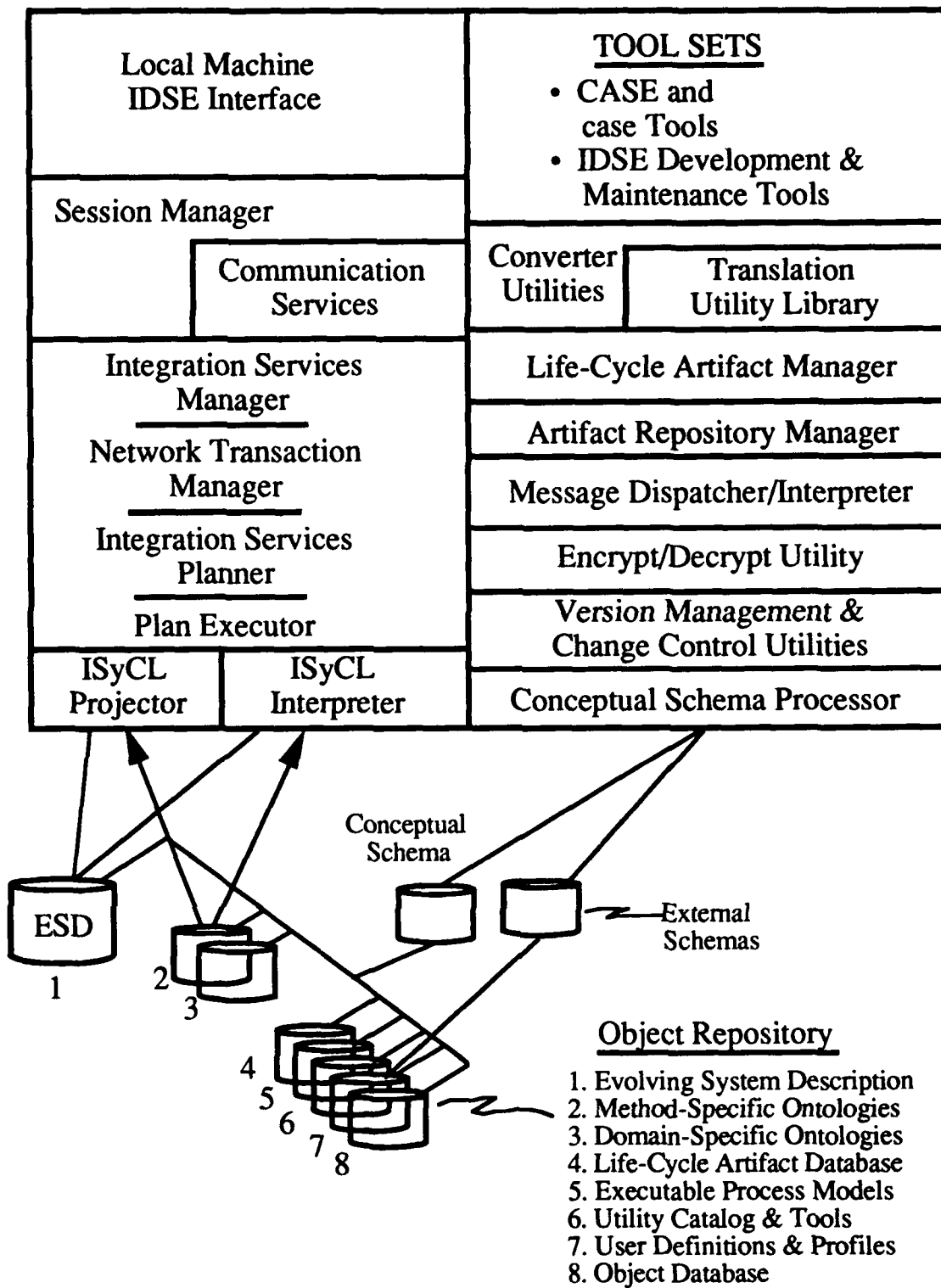
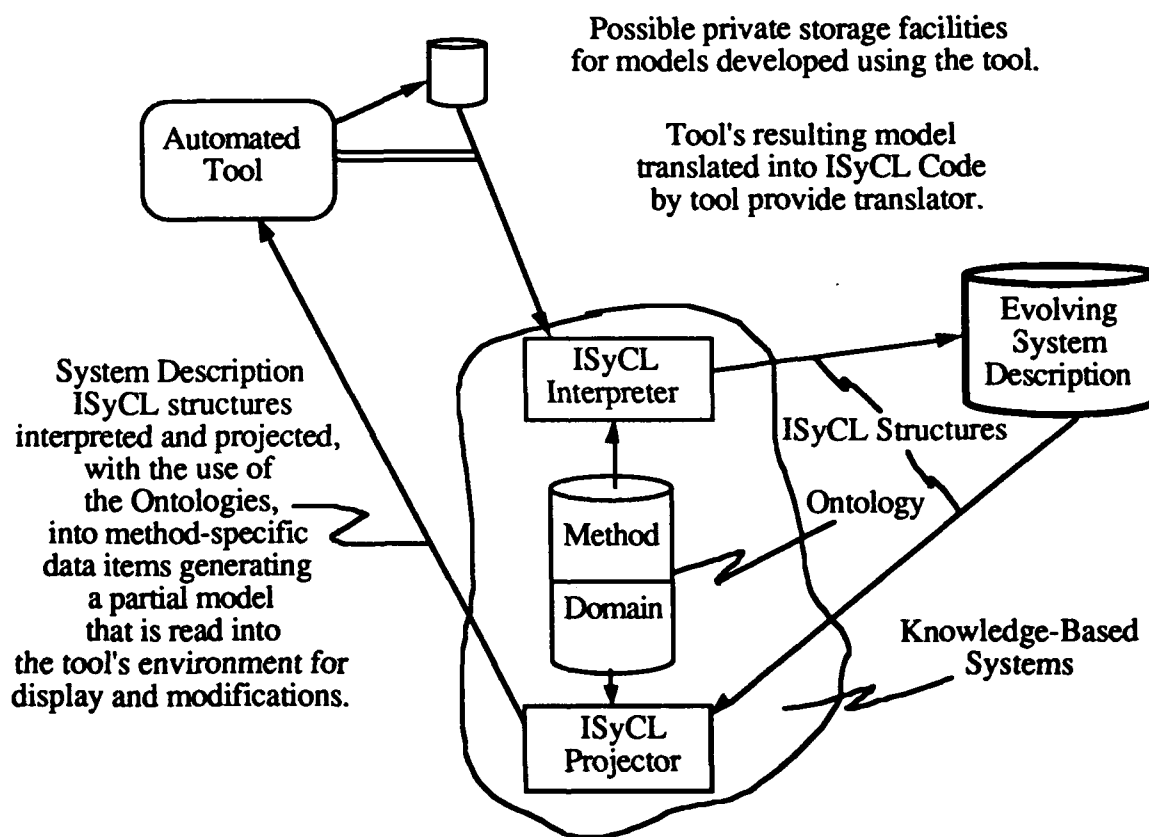


Figure 8. Level 3 IDSE Life-Cycle Artifact Object Repository

### IDSE Concept of Operations

A Method Ontology and Domain Ontology used together will guide the information transfer from a model into the Object Repository via the ISyCL Interpreter (Figure 9). The interpretation process takes a model element, the type of model, and the purpose of that model (*as-is* or *to-be*) and creates assertions about the subject area being modeled which could be inferred based on the Method and Domain Ontologies. These assertions are then added to the repository and existing facts, and assertions relating to that element are examined to ensure consistency. In the reverse direction, the ISyCL Projector, which uses a particular Method Ontology and the Domain Ontology, can generate a partial model in the particular method given the existing information in the Object Repository. These knowledge-based systems are composed of the ISyCL Interpreter and Projector, and the Ontologies will provide IDSE users with a truly integrated information system development environment in which information developed in one phase of a project can be automatically used to decrease the design and development time on other phases of the project.



**Figure 9. Level 3 Information Transfer and Interpretation**

## **4.0 IDSE Usage and Application Scenario**

IDSE is an environment designed to aid and support those individuals who are responsible for developing information systems. Its features include system development tools, data and function integration utilities, and management and storage facilities for life-cycle artifacts. The features of this environment have been discussed in detail in previous sections of this report, but to a system user the most important issue is how to interact with the system and what it will do in order to execute the commands received. In other words, how does this IDSE work?

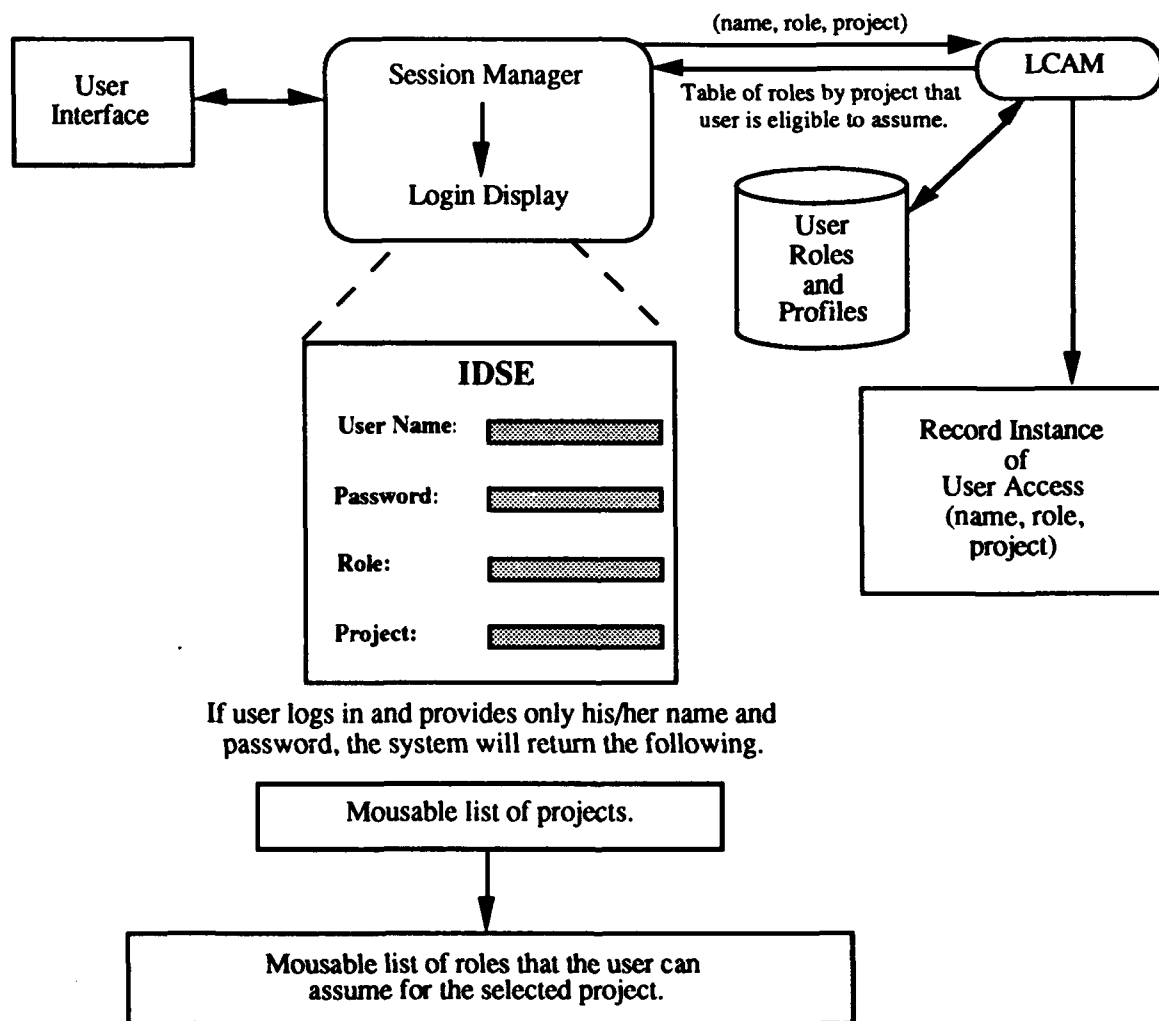
session and procedures that the user would wish to perform. In this section, we will examine the IDSE from a Level 0 user's point of view. That is, we will provide some insight into what a user could expect to see, what commands he or she would expect to use, how the environment provides for the execution of those commands, and what the system will do to provide a minimum integration development support environment (IDSE Level 0).

Perhaps the easiest way to understand the functionality of a system is to explain a typical session. The IDSE user would login to the IDSE from his/her personal computer. At this point, the session manager would assume control from the computer operating system and would maintain control until the user exits the IDSE session.

Figure 10 depicts what a typical login screen would look like. It shows that the user must enter his or her name, password, role, and project. A name and password are used for IDSE access control. The "role" is what the user intends to work in. It is not unusual for a user to assume several different roles in a development effort. For example, an individual may be both a designer and a programmer. When the person assumes the role of designer he or she requires one type of tools, and as a programmer, another. Input by the person acting as chief administrator would have one meaning to the system, and similar input by the same individual acting as a system analyst would assume a different meaning. Thus, it is important that the system know what role the individual is assuming. The system must also know what project the person will be working on. An individual may at any one time be performing tasks on several different projects and possibly assume different roles in each effort. By knowing what project the person is working on, the system will know which files he has access to, and with an installed System Development Framework (SDF), it will know where within the process his or her efforts are to be focused.

## IDSE Concept of Operations

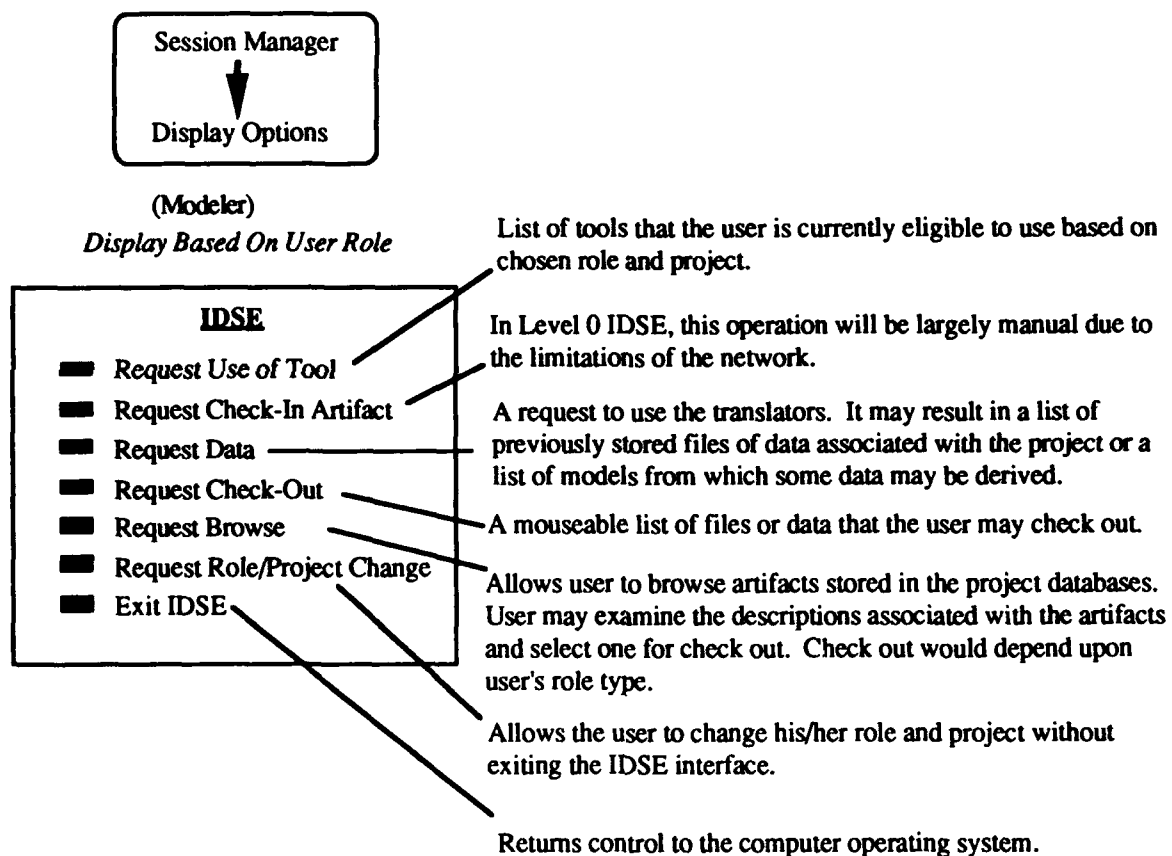
The user may choose only to enter his or her name and password. The system will take this information and, through the LCAM, select from the User Roles and Profiles definitions the projects and roles the user can access. This information is returned to the session manager in the form of a table (Figure 10). A list of projects will be displayed for the user to choose from. Based on the chosen project, the user will then be allowed to choose from a list of role types that he or she can assume for that project. In any event (entering all information or choosing the selection route), the session manager will maintain this table of projects and roles until the IDSE session has ended. Moreover, it will maintain the role and project information until either the session ends or the user elects to change them. This role/project information is used by the system to determine such things as what files the person can work with.



**Figure 10. Login Display for the IDSE**

## IDSE Concept of Operations

Once the user name, role, and project are established, the session manager will display a list of available usage options in that context. Figure 11 shows the display type presented to the individual assuming the role of modeler. From this, we see that a modeler may make one of several requests: use some modeling tool, check in a model, request data transfer from some other model, check out a model, browse the stored artifacts, change projects and/or role type, or quit. This initial display of options would be quite similar for other role types and may even be identical. The difference would be in the results of selections from the list. For example, in this screen, selection of the "use tool" request would display a list of modeling tools that the modeler (based on name and project) can use.



**Figure 11. IDSE Options Available to a Modeler**

The IDEF3 process model in Figure 12 shows what would happen when one of the options from the menu depicted in Figure 11 is selected. The process model actually describes the processes that would occur in a Level 0 implementation of the IDSE; however, its level is high enough to allow almost any automation degree. At Level 0, requesting a tool will produce a display of tool descriptions. The user would indicate which tool to use, but, for



most tools, would have to exit the IDSE session to actually use it. The purpose of indicating a selection would be purely for bookkeeping purposes. The system can record the tool choice, user, role, and project. A record of this type would allow project directors to keep track of tool usage and to determine if the prescribed tools are being used. At this level, automatic enforcement of tool usage is not possible.

Check-in is the most manual procedure in an IDSE Level 0 implementation, but there is a procedure to be followed which must be enforced by the librarian. The check-in procedure, as it would be displayed to the librarian, is pictured in Figure 13.

As we can see, there are several operations that must be performed when any artifact (model, document, etc.) is checked in. First, the file must be made available to the librarian ("Place File in Receiving"). This may entail simply writing it to a diskette, or with sufficiently advanced communication facilities, it may be transferred over a network to a special working directory. The user must register the file "Complete the Check-in Form." A sample check-in form appears in Figure 14. This form may be mailed via the communication network to the librarian or printed and hand-delivered. Basically, what this form contains is the information that is needed to describe the artifact in the LCAR and information that will assist the librarian in determining the version and status of the file.

One action associated with the check-in procedure not indicated in the IDEF3 model in Figure 13 is the user notification to the librarian that there is a file ready for check-in. This omission is intentional because in a Level 0 IDSE, this notification will very likely be handled without the aid of the communication network.

Figure 15 provides an overall description of this check-in process. In this process, after determining that the user checking in the file has the authority to do so, the librarian accesses special commands that will allow the extraction of project check-in procedures from the system development models for the project site. These may be in the form of IDEF3 process models or may be as simple as a list of actions the librarian must execute.

The librarian will be responsible for versioning of the artifacts (Unit of Behavior (UOB) 6 in Figure 15), executing file encryption utilities (UOB 11), ensuring that all review procedures are followed (UOB 12), creating new artifact pointers (UOB 7), and the execution of the commands that will store files in secure directories (UOB 13). Most of the details concerning when to version and who does the reviews will be contained in the process descriptions. The actual objects versioned in the LCAR are not the files themselves. The objects will in most cases be pointers to the stored files. The pointers contain the pathname by which the file may be obtained. Thus, different versions of one of

## IDSE Concept of Operations

these pointers will contain different pathnames, and therefore, effect version control of the file. File encryption actions will be similarly modified in the minimum LCAR.

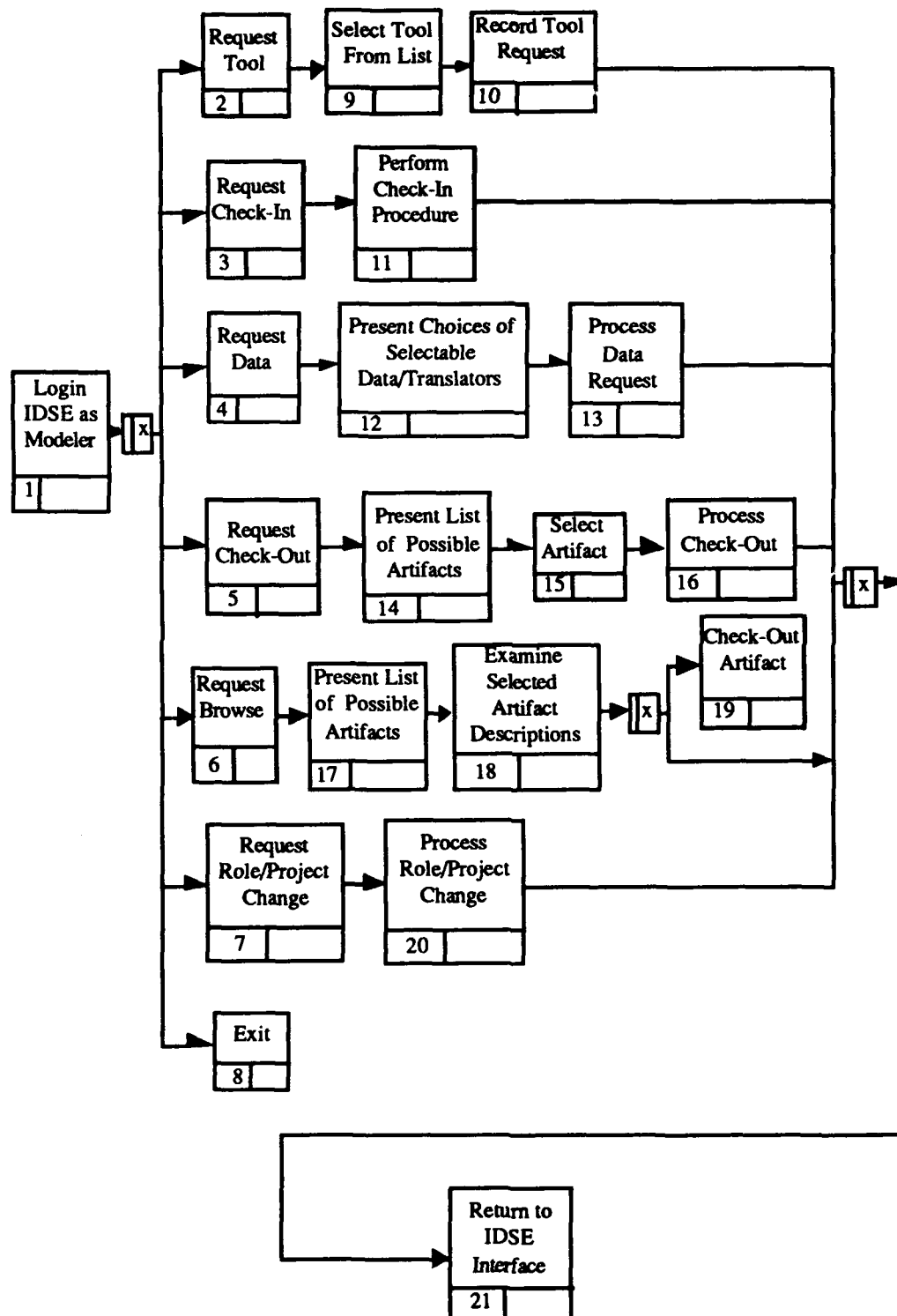


Figure 12. IDEF3 Process Model for a Modeler Using a Level 0 IDSE

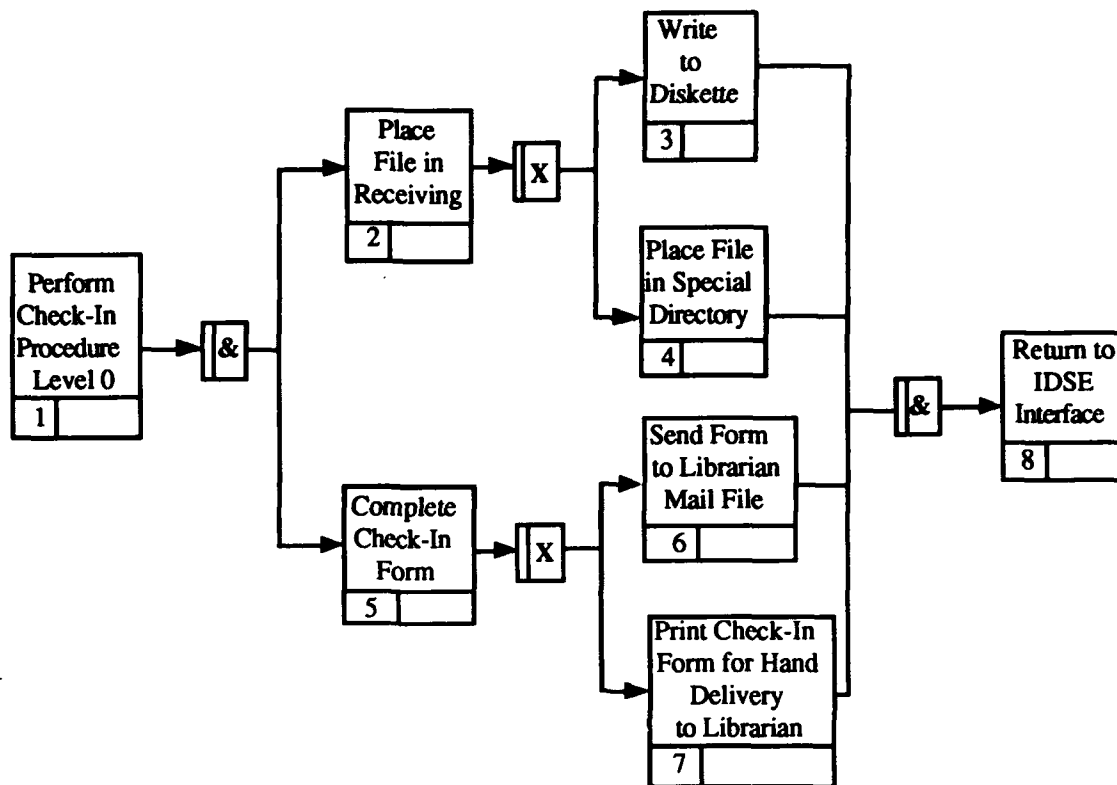


Figure 13. Check-In Procedure for IDSE Level 0

The primary integration support of a Level 0 implementation will be provided by the translators (the Request Data process in Figure 11). Figure 16 illustrates how the data translation/conversion is expected to work. When the user "Requests Data" from the IDSE, he or she will be given information on how to obtain and use the requested data. These translators will exist primarily for the purpose of taking some of the data that was produced when one model is built and using it as input into the production of some other model developed using another method. It is an accepted fact that data produced by the different modeling methods is related. Also, not all of the data produced by one method is required or even usable in another. A tool for which a translator has been written will, in addition to producing its normal model data, be capable of producing a separate data file in a standard model data exchange format. For a tool to use this data, it must be modified slightly to allow input other than the normal keyboard input. This data will very likely never result in a complete model of the second type. It will provide:

- decreased development time for a model that uses input in the form of data gathered from a model (in a different method) developed in an earlier phase (or for that matter another project), and

- an assurance that models and systems are more consistent than those developed without this data sharing.

**Check-In Form**

User Name: **Filled in based on login data.**

Name of Project: **Filled in based on login data.**

User Role: **Filled in based on login data.**

Name of File:

File Location:

State of Completion:

Purpose of Check-In:

Type of Data:

Tool Used:

Tool Version:

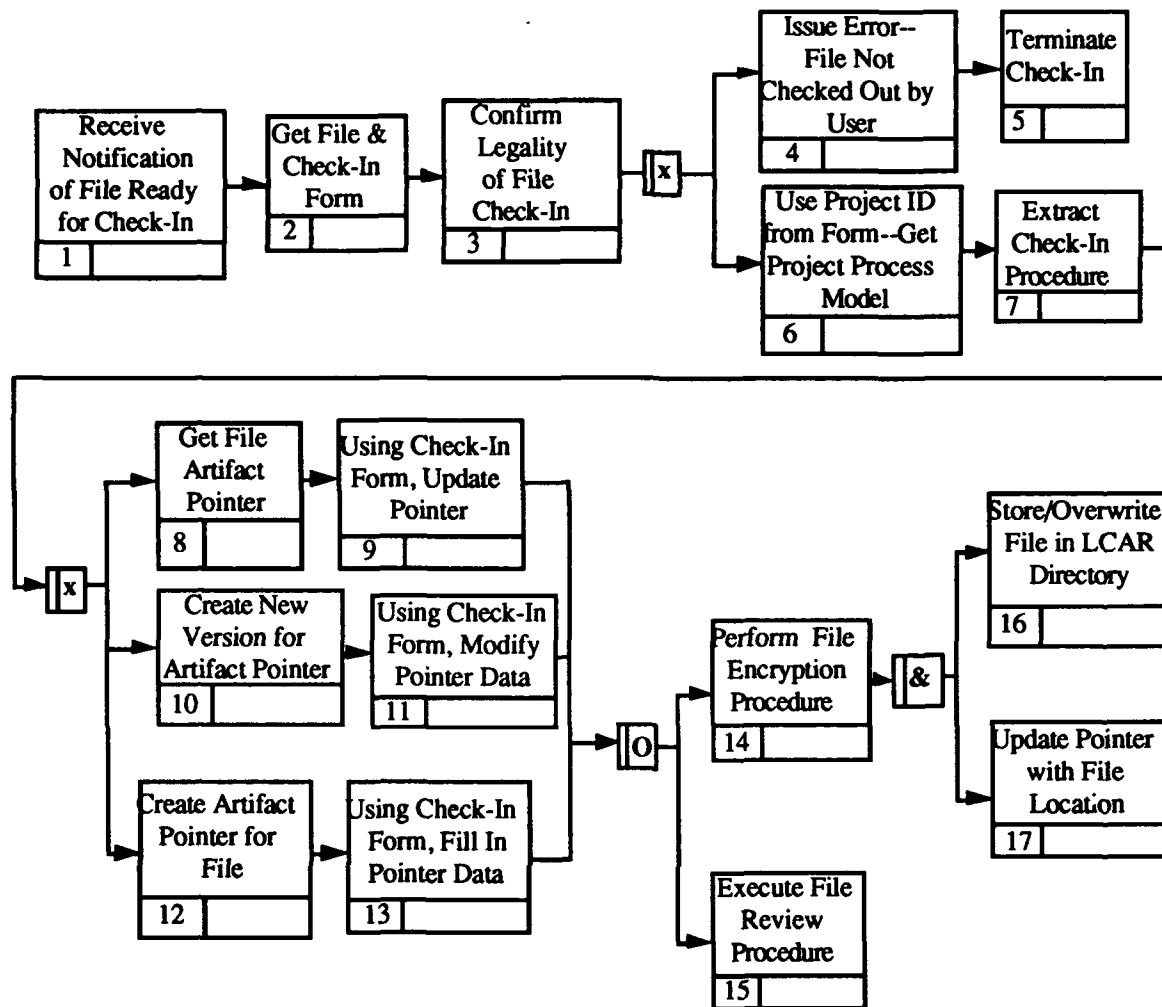
Description of File Contents:

This description should contain information such as the type of model (e.g., IDEF1), the type of document (e.g., status report), or contents of a list (e.g., list of requirements), etc. Moreover, the description should be detailed enough so that someone browsing life-cycle artifacts would have a clear understanding of the data located by the artifact pointer.

**Figure 14. File Check-In Form**

Although the data transfer from one method to another is primarily a manual process even with a Level 0 implementation, the IDSE users are provided with assistance that has hitherto not been available.

The request for file check-out is a much more automated process than checking in. As shown in Figure 17, after requesting the "file check-out" option, the user will be presented with a list of the artifacts that he or she is allowed to check out. This list is built by the system using the stored user name, role, and project. From this list, the user will select the file and version of that file he or she wishes to check out.



**Figure 15. Level 0 Librarian Check-In Process**

The automated check-out process will be as follows.

1. Get purpose of the check-out from the user.
2. Confirm that the file can be checked out (i.e., not already checked out to some other user), and if so, continue.
3. Make a record of the check-out containing information such as:

file name:	Inserted by the system.
file version:	Inserted by the system.
permanent location:	Provided by the system from artifact pointer.

## IDSE Concept of Operations

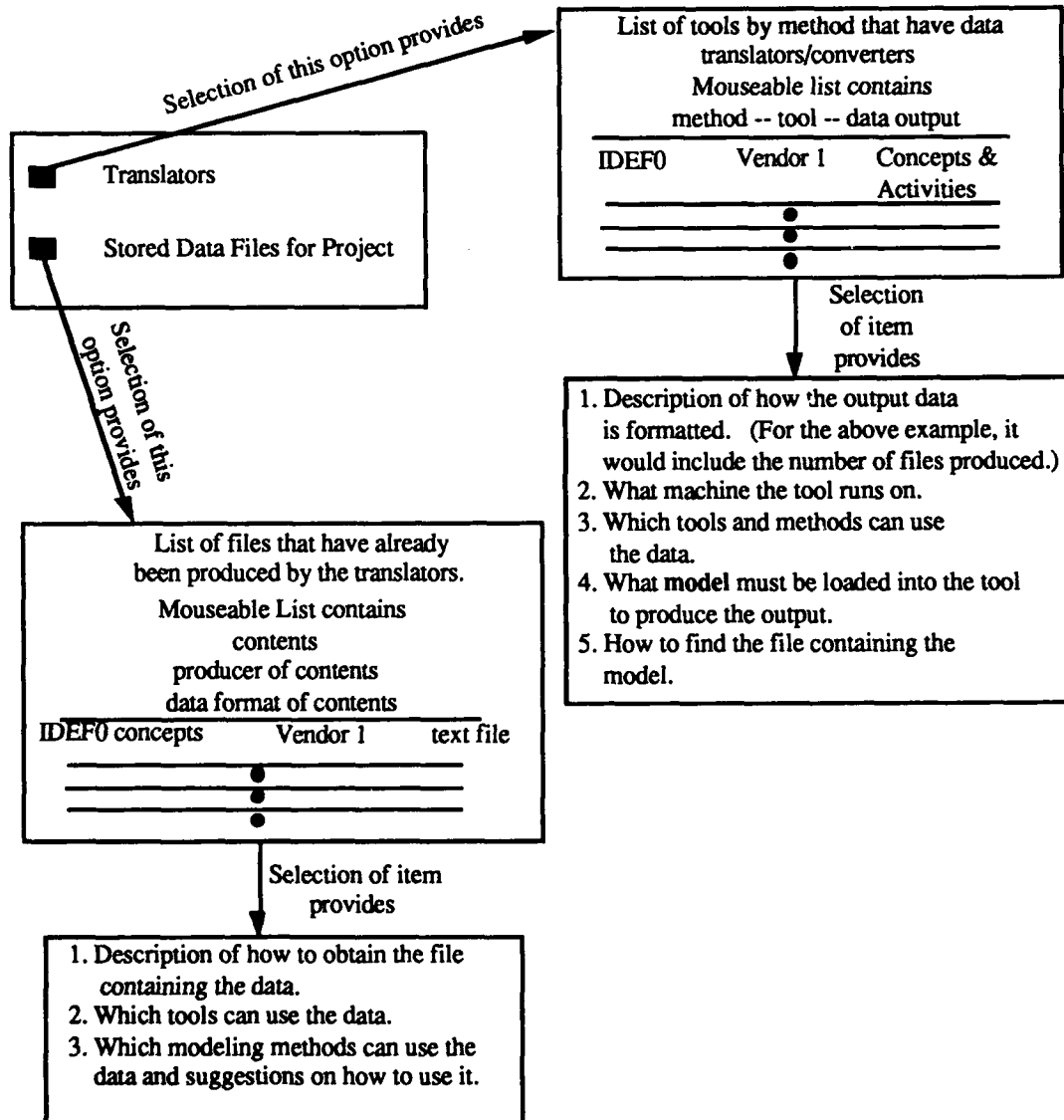
user name:                      Inserted by the system from login file.

user password:                Inserted by the system from login file.

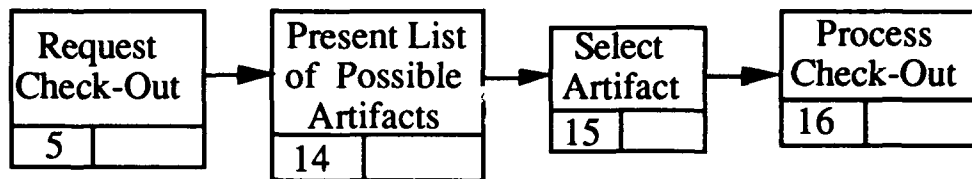
project:                        Inserted by the system from login file.

purpose:                        Provided by the user requesting the file.

4. Place check-out record in check-out list.
5. Place file in temporary directory.
6. Inform user of how to obtain the file.



**Figure 16. Requesting Data in a Level 0 IDSE**



**Figure 17. Check-Out Process**

This record will be available to the librarian when the user checks in the file at a later date. The librarian will use this record to determine if the user is legally entitled to request a check-in. Thus, even though the user may be able to circumvent the controls and actually modify a file without checking it out, the librarian will not allow the file to be checked in to become a part of the system description unless there is a check-out record. This procedure provides for system description security by not allowing uncontrolled modifications.

One of the most common activities that database users perform is browsing. In the LCAR, this will be just as important. Therefore, one of the commands available to the user will be the browse request. In a Level 0 implementation, the users will not browse the actual artifacts themselves but the objects that point to them. When a file is checked in, a form is filled in and later used to either update or create the pointer to the object. Although the check-in discussion is applied to a modeler checking in a model file, all check-ins will require some type of form and pointer update. This means that the pointers stored in the artifact catalog will contain information that would be useful and informative to the individual requesting the browse operation. For example:

- the description of the data contained in the file,
- what data types are in the file,
- file version and status, and
- tool and version of it used to create file.

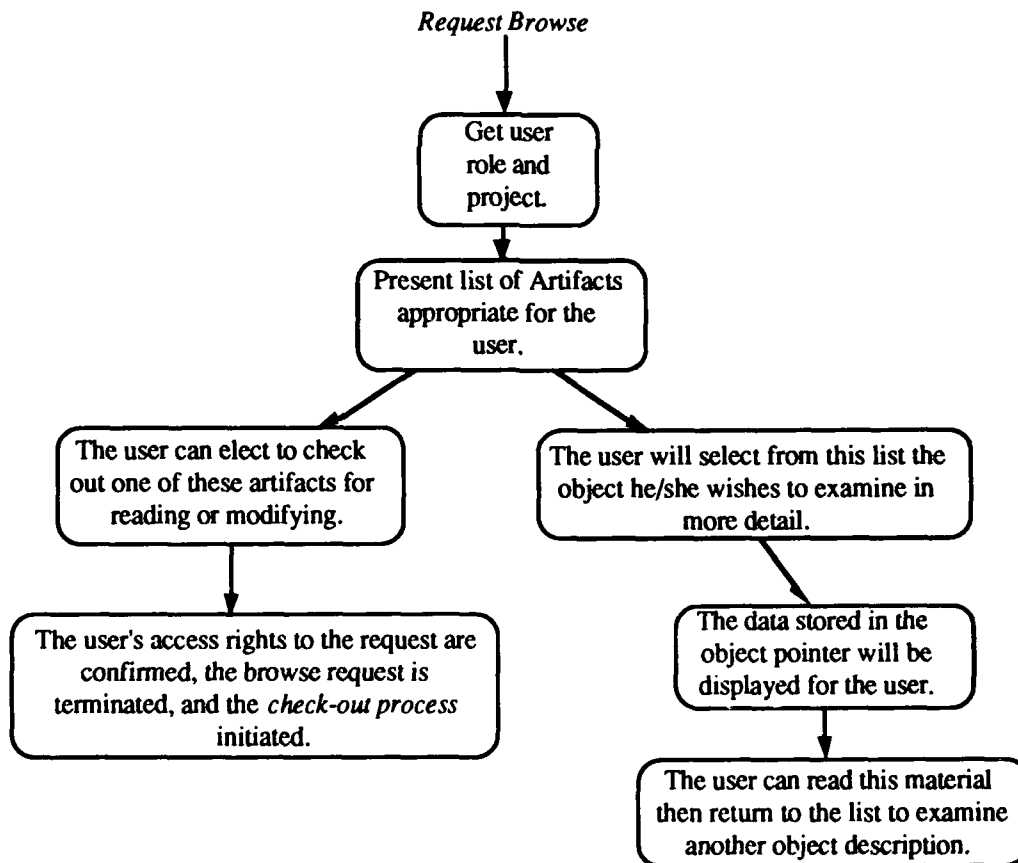
The information to be viewed by the person browsing the artifacts will be limited only by the user's access privileges and the amount of data to be stored in the pointer to the artifact.

To browse the artifacts, the user will follow the procedure outlined in Figure 18. The list provided to the user will normally be only those files of interest to his or her role and project. However, there will be some user role types that will have browsing access across project boundaries. These will be needed so that project and model development leaders can inspect information about files maintained for other projects. They may wish to copy files from other development efforts to aid those working on their project. The data about the files displayed for the requestor will also be based on the user's role type. For

### *IDSE Concept of Operations*

instance, a typical user will not be provided with information about where the LCAM has stored the files. A librarian may have need for this information; therefore, if the user is a librarian, it will be provided. As the user browses the artifact descriptions, he or she may select one of the files for either reading in its entirety or for modification. If this occurs, the system will switch to the check-out mode, verify that the user has read/modification rights, and then process the check-out as described previously.

Finally, IDSE would allow a user to change role or project declaration and ultimately "Exit" the system. Each initial menu will contain these options. The reason for them is that a user will likely have more than one role on a project and will also be involved on more than one project at a time. The user will be able to change roles within the project or just change to another one without leaving IDSE. The effect of these commands would be to reset the current role and project designators. The "Exit" command would in effect allow a graceful termination of an IDSE session. It would remove from local memory the information about the user and prepare the system to be used by another.



**Figure 18. Browsing the Artifacts**



At the beginning of this section, we stated that IDSE is an environment designed to aid and support individuals responsible for development of information systems. We have described how even a Level 0 implementation provides the means for transferring data from one development phase to another via the translator/converter utilities. Finally, it provides for control over the system development process by providing an audit trail of system usage via the check-out facility and human enforcement of the local system development process via the librarian-controlled check-in procedures and commands.

## **5.0 Summary and Conclusions**

In summary, the IDSE concept is structured around the idea of support sophistication levels. The notion of an IDSE level should not be confused with IDSE versions; levels represent *levels* of technological sophistication. Each level represents a complete and viable development support environment, not a version of some environment. Level 0 represents what we believe is a reasonable environment requiring little or no additional computer resources investment from the implementing organization. Indeed, for a small organization, this may be all it will need for the foreseeable future. Advancing from Level 0 to Level 3 will, with each step, provide increased integration support, increased automation for the system development process, increased tool sophistication, and correspondingly require increased financial and computer resources.

The concepts behind the development environment for IDSE have five general requirements placed on the design of each level. Each IDSE level must provide:

- 1) an integrated set of system development and management tools,
- 2) automated support for information transfer between tools,
- 3) management and control of the system development process,
- 4) management of information system design artifacts, and
- 5) IDSE internal development, management, and evolution capabilities.

In this section, we will summarize how these requirements would be addressed in each of the IDSE levels.

## **5.1 Integrated Set of Development and Management Tools**

An integrated set of system development and management tools will be provided in all levels of IDSE. The IDSE design will provide an environment in which any tool, regardless of the vendor who produces it, can be used to address the area of the system development process for which it is designed. Furthermore, the design allows for the development of tools to assist in as yet unautomated areas of the system development process. As tools for these areas are developed, they can become part of IDSE without interfering with the execution of existing tools.

In a Level 0 implementation, tools are loosely coupled. Their use is controlled manually in the sense that the users login to IDSE and request a tool. The user will be presented with a list of acceptable tools given his or her role and user profile. However, he or she can circumvent this process because it does not actually load the tool for him or her, it just records that user *x* used tool *y*. The manual enforcement comes when he or she attempts to check in any work done. If the correct tool was not used, the librarian will not accept his or her work. Tools that provide for a Level 1 capability will have as part of their construction a process by which they can be remotely executed.

Tools that support Level 2 integration must be able to generate an ISyCL representation of models created with that tool. It must also support the attachment of ISyCL constraints to models. In other words, a text editor must be supplied in which constraints can be written which will be appended to the ISyCL representation of a model when it is produced. The tools will not be required to do anything with these constraints other than to append them to the output file. The Level 3 tools will have the capability of internally accessing the integration services and displaying partial models provided by the ISyCL projectors that will exist in a Level 3 IDSE.

An important and noteworthy characteristic is that each IDSE level will provide all the capabilities of the preceding levels. This means that any system development tool, regardless of whether it has been developed to exist within an IDSE, can be useful in the environment even if in a somewhat manual form, as will be common in a Level 0 environment.

## **5.2 Automated Support for Information Transfer**

A minimum level of automated support for information transfer between tools is provided in Level 0. Such transfers are provided by the Data Conversion/Translation

## *IDSE Concept of Operations*

Utilities & Libraries, which provide a means whereby tool vendors can provide more than just one data type. The first type would be their normal output since the individual methods that the tools automate address a particular area of the system development process. However, part of the output is logically input to other areas of the process. Translators or converters can be written for a tool to convert that part of its output usable as input to another phase of the development process. Other tools can be modified slightly to accept this data as input, thus providing a minimum level of integration and information transfer for the tools. As a rule, this would require very little modification of existing tools.

A site that implements a Level 1 IDSE will have enhanced integration and information transfer via the integration services concept. The services approach to integration concentrates on (1) service advertisement (advertisement), (2) service specification (protocol), and (3) integration services facilitation (contract specification). The integration services component of the IDSE will consist of the ISM, Network Transaction Manager, Plan Builder, and Plan Executor. Associated with these components of the integration services will be libraries of pregenerated plans (hard-coded plans frequently used as translation plans at the site) and a library of tool information that consists of the information about the data that tools can either produce or consume. These two libraries will be used by the ISM utilities to provide the requested integration services. The ISM will allow the IDSE to focus on providing for the integration of tools and the information created by those tools by establishing a client/server relationship between them. Tools that request data from the Session Manager will do so via the Network Transaction Manager, which will interpret the integration request and pass it to the Plan Builder. It will be the responsibility of the Plan Builder to determine if the user's request can be satisfied. Once a valid plan for the service request has been built or extracted from the library, the plan is passed to the Plan Executor, which will execute the plan and return the results.

The ultimate success of the ISM and the ISP depends on the definition of the languages necessary to support the communication between the user and the ISM/ISP and between the ISM/ISP and the legacy tools that make up the existing hardware and software platform. Four languages have been identified: (1) the Service Advertisement Language that provides a high-level abstract description of the services provided by the current software and hardware configuration, (2) the Service Contract Language that will describe the data artifacts, hardware requirements, applications, and invocation sequences necessary to fulfill an advertised service, (3) the Service Protocol Language that will define the input and output formats used by the an advertised service, and (4) a Query Language to provide not only a mechanism for formulating requests to the ISM/ISP but a means by which the

### *IDSE Concept of Operations*

ISM/ISP can formulate responses as well. With the ISM/ISP concept, the IDSE will offer the advantage of integration without the corresponding requirement that all tools conform to the same data format.

The Level 2 IDSE will have an ISyCL Loader & Reader with an object repository to enhance the integration of tools and data transfer. ISyCL is designed to meet the definitional and knowledge representation needs across all system design levels. ISyCL provides an object-centered approach to the description of model elements and provides the facilities for attaching constraints to the model elements. It is designed to support all individual types involved in a system design effort, from the area expert to the database designer. ISyCL is one language, but to address the variety of system development needs, it is organized in layers. The current design of ISyCL includes four layers: Area Expert Layer, Analyst Layer, Information Systems Design Layer, and Method Formalization Layer.

Constraints are added to the model using the area expert layer. These constraints will be formalized by an analyst using the analyst layer. He or she will submit the model for release approval. Finally, whoever is given the necessary authority will save the model to ESD. The Loader will place the design artifacts into the ESD. However, in order for a second tool to benefit from this, an ISyCL Reader will be provided that will interpret/translate the neutral description of the ESD into usable and understandable information by a tool that can interpret an ISyCL representation. This use of ISyCL and the Loader and Reader will eliminate the need for the tool format translators that were part of the Level 0 IDSE. However, it will also mean that all the attached Level 2 IDSE tools must write and read the ISyCL representation of the method they automate.

A Level 3 IDSE will provide the functionality to interpret the meaning of the model elements through the use of Domain Ontologies, Method Ontologies, and an ISyCL interpreter and projector. This functionality will be provided by the ISyCL interpreter and projector that will work with the domain and method ontologies. The ISyCL interpreter and projector that will work with the domain and method ontologies are knowledge-based systems that provide a truly integrated information system development environment in which information developed in one phase of a project can be automatically used to decrease the design and development time on other phases of the project. Ontologies are knowledge bases containing information about the concepts and relationships that exist between the concepts. The IDSE will have two ontology types: domain and method. A Method Ontology is a meta-level description of the objects, relationships, and constraints pertaining to a particular method; in other words, "what is" true about a specific method. A

Domain Ontology is a meta-level description of the universe of discourse relating to a specific domain. A Method Ontology and the Domain Ontology used together will guide the information transfer from a model into the Object Repository through the use of the ISyCL Interpreter. In the reverse direction, the ISyCL Projector, used in conjunction with a Method Ontology and the Domain Ontology, can generate a partial model for that method, given the existing information in the Object Repository. With the inclusion of these knowledge-based systems, the IDSE will provide for automatic model translation, data consistency across the entire system development process, and the maintaining of traceability links between data artifacts.

### **5.3 Management of the System Development Process**

Management and control of the system development process will be provided for in IDSE with an environment for system development which will allow intelligent and automated coordination and control throughout the system development process. IDSE will use a framework to capture and employ the system development knowledge at a particular site. This framework will include not only the life-cycle analysis, design, implementation, maintenance, and decision-making activities but also functionality that allows method identification for specific development situations. Functionality will also allow an individual site to define and specialize its own system development processes and, in the more advanced levels, provide automatic enforcement of this definition. The SDF definition in the system framework will provide individual sites with a means of specializing the system development process to meet their needs and those of individual projects at that site. The system framework will provide the following.

1. A framework graph: This will indicate the types of questions that can be answered depending upon the situation and cell of the framework.
2. Definitions associated with each cell in the graph, which will provide:
  - a. the properties, objects, and user roles that characterize the development situation associated with each cell, and
  - b. the context and intercell relations that characterize the cell interpretations.
3. The overall and individual role system development process descriptions. These will be process models of the IDEF3-type.
4. The identification of the component methods and tools associated with each cell in the system framework.

### *IDSE Concept of Operations*

A framework generator is being designed to assist in the development of site- or project-specific system development frameworks. It will also help tailor the general development methods, tools, and procedures to fit the specific enterprise, site, project, and user. It will request enterprise and user information profiles and, from these, generate the necessary system description framework for the specific site or project. It will outline the tools and methods to use, as well as what each user role or involvement should be at each step in the system development process. However, prior to the development of this utility, the sites will have to generate the system description framework more or less by hand with the assistance of various automated tools, such as an IDEF3. Each IDSE site will have a "Framework Installer" that will allow the installation of the framework into the system. These components will eventually comprise the framework definition mechanism and, together with a "Framework Inspector" (a utility for inspecting the various components of the framework), will make up site framework utilities.

In a Level 0 IDSE implementation, the definition of the framework will be primary manual, which will likely consist of IDEF3 process models and textual descriptions of the rules to be followed in the various processes. The enforcement of the system description process will be almost totally in the hands of the librarian. The installer will likely be simply a utility that stores this process description in a schema accessible by librarian queries. Level 1 will not be much more automated in the definition of the framework, but the enforcement of the rules and procedures will become more automated. The Framework Generator and the fully automated execution of the site-defined system development process will be available in both Level 2 and Level 3 implementations of the IDSE.

## **5.4 Management of Information System Design Artifacts**

Management of information system design artifacts will be provided for first in the *minimum* Life-Cycle Artifact Repository and in more advanced levels by an Object Repository. IDSE requirements could almost be called the requirements for a repository. It will provide the storage and management utilities necessary for controlling the massive amounts of information generated in a system development effort as well as the available tools and system development process models. Associated with these repositories will be the means for providing object versioning, change control, and configuration management. Not the least important will be the means for management and control of complex objects.

In the minimum LCAR of a Level 0 implementation, and to some degree in a Level 1 implementation, the repository Artifact Catalog database will store only pointers to the file

## *IDSE Concept of Operations*

storage areas, and the Life-Cycle Artifact Manager will be managing these pointers and the file storage areas. The Artifact Repository Manager will provide the utilities necessary for controlling and maintaining the environment. Version, configuration, and change control capabilities in a Level 0 implementation will be provided via special commands available to the system librarian. Due to the increased networking capability of a Level 1 implementation and the auto-check-in facility, these features will be more robust.

It is not until Levels 2 and 3 that the system will have a true object repository and be capable of providing for automated version, configuration, and change control. The Object Repository of these IDSE levels will be defined using an object-oriented data model since they allow for versioning at the object and global level. This is ideal for tracking data throughout the modeling and design processes. Site-specific change control policies will be implemented via control points that will provide the automatic enforcement of design changes and modifications of interested individuals. In this environment, when a requirement is modified, the system will be able to immediately notify the user of the other design objects affected by the change. Furthermore, the object-oriented data model allows easy inclusion of the "part-of" relationship between complex objects that is so important in modeling the complex relationships that exist among the data elements in a system design effort. This means that in Level 2 or 3 implementations, provision can be made for the management and control of complex objects, in a system development environment, that will consist of composite, aggregate, and complex object collections. All these object types involve the "part-of" or "has-parts" relationships that exist between many objects in the real world.

### **5.5 IDSE Internal Evolution Capabilities**

IDSE internal development, management, and evolution capabilities (the IDSE administration) will be provided by the IDSE system maintenance tool sets. Throughout this report, we have discussed various utilities and features associated with IDSE. Furthermore, we have maintained that any IDSE level can evolve as new tools and methods are developed. In order to enable this evolution to occur, IDSE will provide these system maintenance tool sets. In a Level 0 implementation, the administration tool sets will include utilities that allow new tools and permit the inclusion of new translators and translation libraries. A Level 1 implementation will provide ISM maintenance tools that allow IDSE sites to extend the integration services they provide. These utilities will be used to insert new pregenerated plans into the library of plans and to include new tool information as vendors modify their tools to either provide or accept outside data. In Level 2, the

### *IDSE Concept of Operations*

maintenance tools will include those that allow the addition of new capabilities to the Loaders and Reader. New tools and methods will be developed, and knowledge of these must be included in the Loader and Reader. At all levels, the maintenance tools will provide the facilities for defining new objects in the database(s). The maintenance tools of the Level 3 environment will be particularly concerned with providing the capability of including new ontologies (either domain or method). These capabilities provide an IDSE with the ability to evolve so that the IDSE initially installed, regardless of the level, can evolve gracefully.

Finally, by providing four levels of IDSE integration technology we provide an information system development environment that can be installed and used not only by large, wealthy organizations, but by small and medium-sized organizations as well. It will provide an environment in which an enterprise can control the information system development process, and create and maintain more consistent information systems. Furthermore, regardless of the level, it provides a means whereby an enterprise can decrease the cost (both in time and money) of developing and maintaining evolving integrated information systems.



## Bibliography

- [AD/Cycle 90] Proceedings, 3 Rs of Software Automation: Re-Engineering, Reusability, Repositories. An Extended Intelligence, Inc. Conference and Tools Exhibition, 25 East Washington Street, Suite 600, Chicago, IL 60602.
- [Coleman 89] Coleman, D. S. "A Framework for Characterizing the Methods and Tools of an Integrated System Engineering Methodology (ISEM)," Pacific Information Management, Draft 2 Rev. 0, May 1989, p. 2.
- [CIM-OSA 89] *Open System Architecture for CIM*, Research Reports ESPRIT, Project 688, Amice, Volume 1. Springer-Verlag, New York.
- [Decker 92] Decker, L. P., and Mayer, R. J. *Information System Constraint Language (ISyCL) Technical Report*. AL-TP-1992-0019, Armstrong Laboratory, Wright-Patterson Air Force Base, September, 1992.
- [DICE 89] *DARPA Initiative in Concurrent Engineering (DICE): Red Book of Functional Specifications for the DICE Architecture*, February 28, 1989. Contract MDA972-88-C-0047, Concurrent Engineering Research Center, West Virginia University.
- [EIS 86] *The Department of Defense Requirements for Engineering Information Systems: Volume 1 - Operational Concepts; Volume 2 - Requirements*. J. L. Linn, and R. I. Winner (eds.), EIS Requirements Team, The Institute for Defense Analyses, Alexandria, VA.
- [EIS 89] *Engineering Information Systems: Volume 1 - Organization and Concepts; Volume 2 - Specifications and Guidelines*. Honeywell Systems and Research Center, Minneapolis, MN, October, 1989.
- [Goldfine 87] Goldfine, A., and Konig, P. *A Technology Overview of the Information Resource Dictionary System (Revision 1)*, Center for Programming Science and Technology, Institute for Computer Science and Technology, National Bureau of Standards, April, 1987.

- [FPP 90] *Framework Programmable Platform for Advanced Software Development Workstation: Concept of Operations Document*, Knowledge Based Systems, Inc., Prepared for NASA-Johnson Space Center, RICIS Program: Subcontract Number 077: Cooperative Agreement Number NCC 9-16, September, 1990.
- [IDS 89] *Integrated Design Support System (IDS)*, AFHRL-TR-89-6: *Volume I - Executive Overview; Volume II - IDS Introduction and Summary; Volume III - IDS Requirements; Volume IV - IDS Task Results; Volume V - IDS Software Documentation*. Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, OH, December, 1989.
- [IISS 85] Judson, D. L. *Integrated Information Support Systems*, 1985; *Integrated Information Support System (IISS): An Evolutionary Approach to Integration, Manufacturing Technology Division*, Materials Laboratory, Air Force Wright Aeronautical Laboratories, 1985.
- [ISO-CSA 82] *Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO/TC97/SC5/WG3, 1982.
- [Mayer 90a] "IDEFo Function Modeling: A Reconstruction of the Original Air Force Report," Mayer, R. J. (ed.), Knowledge-Based Systems, Inc., College Station, TX 1990.
- [Mayer 90b] "IDEF1 Information Modeling: A Reconstruction of the Original Air Force Report," Mayer, R. J.(ed.), Knowledge-Based Systems, Inc., College Station, TX 1990.
- [Mayer 90c] "IDEF1x Data Modeling: A Reconstruction of the Original Air Force Report," Mayer, R. J.(ed.), Knowledge-Based Systems, Inc., College Station, TX 1990.
- [Mayer 90d] *A Design Knowledge Management System (DKMS)*, SBIR Phase I Final Report, AFHRL-TP-90-81, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, OH, December, 1990.
- [Mayer 91a] Mayer, R., *Framework Research Report*, Final Technical Report, Integrated Information Systems Evolution Environment, United States Air Force Armstrong Laboratory, Wright-Patterson Air Force Base, OH, June, 1991.

- [Mayer 91b] Mayer, R. J., Painter, M. "IDEF Family of Methods," Technical Report, Knowledge Based Systems, Inc., College Station, TX, January, 1991.
- [Mayer 92a] Mayer, R. J., Menzel, C. P, deWitte, P. S., and Painter, M. K. IDEF3 Technical Report, AL-TP-1992-XXXX, Armstrong Laboratory, Wright-Patterson Air Force Base, OH, December, 1992.
- [Mayer 92b] Mayer, R. J. , Edwards, D. A., and Painter, M. K. IDEF4 Technical Report, AL-TP-1992-XXXX, Armstrong Laboratory, Wright-Patterson Air Force Base, OH, December, 1992.
- [Mayer 92c] Mayer, R. J., Griffith, P., Menzel, C. P., Cullinane, T. P., and Painter, M. K. IDEF6: A Design Rationale Capture Method Concept Paper, AL-TP-1992-0050, Armstrong Laboratory, Wright-Patterson Air Force Base, OH, November, 1992.
- [Menzel 90] Menzel, C. P., Mayer, R. J., and Edwards, D., IDEF3 Formalization Report, AL-TP-1991-0043, Wright-Patterson Air Force Base, OH, October, 1991.
- [Menzel 91] Menzel, C. P., Mayer, R. J., and Painter, M. K. IDEF5 Ontology Description Capture Method: Concepts and Formal Foundations, AL-TP-1992-0051, Armstrong Laboratory, Wright-Patterson Air Force Base, OH, November, 1992.
- [Painter 91] Painter, M. Information Integration for Concurrent Engineering (IICE): Program Foundations and Philosophy, Conference Proceedings for the IDEF Users Group, May, 1991.
- [PCTE 89] Brown, A. W. *Database Support for Software Engineering*, Wiley & Sons, New York, NY, 1989.
- [Ross 77] Ross, D. T. Structured Analysis (SA): A Language for Communicating Ideas, IEEE Transactions on Software Engineering, January, 1977.
- [SEM 83] *System Development Methodology User's Manual*, Hughes Aircraft Company, UM170131000, October, 1983, Vol 7.

[SSE 90] Barnes, F. N. Software Support Environment System Project Overview, Lockheed Missiles & Space Company, April, 1990.

[IUG 90] "Working Group 1 (Frameworks) Technical & Test Committee," IDEF Users Group, The IDEF Enterprise Framework, Document IDEF-UG-0001, Version 1.0, January 1990.

[Wilson 87] Wilson, M. L. *Information Automat: Concept Definition Facility*. IA Systems, Inc., San Jose, CA, 1987.

[Zachman 86] Zachman J. A. A Framework for Information Systems Architecture, IBM Los Angeles Scientific Center, G320-2785, March, 1986.