

# UNITED STATES AIR FORCE RESEARCH LABORATORY

---

## Design Concepts for Automating Maintenance Instructions

Norman Badler  
Charles Erignac

University of Pennsylvania  
Center for Human M&S  
200 South 33rd Street  
Philadelphia, PA 19104

Patrick Vincent

TASC Inc.  
2555 University Blvd.  
Fairborn, OH 45324

Edgar Sanchez

Boeing Company  
S0343067  
P.O. Box 516  
St. Louis, MO 63166

Edward S. Boyle  
Jeffrey L. Wampler  
John D. Ianni

Air Force Research Laboratory

February 2000

Final Report for the Period January 1999 to February 2000

*Approved for public release; distribution unlimited.*

Human Effectiveness Directorate  
Deployment and Sustainment Division  
Sustainment Logistics Branch  
2698 G Street  
Wright-Patterson AFB OH 45433-7604

DTIC QUALITY INSPECTED 4

20000925 077

## NOTICES

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

Please do not request copies of this report from the Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161

Federal Government agencies registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center  
8725 John J. Kingman Rd., Ste 0944  
Ft. Belvoir, VA 22060-6218

## DISCLAIMER

This Technical Report is published as received and has not been edited by the Air Force Research Laboratory, Human Effectiveness Directorate.

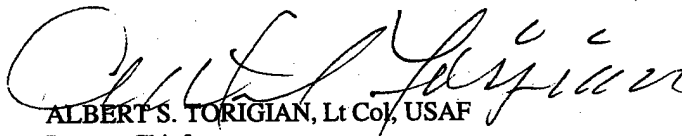
## TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2000-0088

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

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

FOR THE COMMANDER

  
ALBERT S. TORIGIAN, Lt Col, USAF  
Deputy Chief  
Deployment & Sustainment Division  
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE February 2000	3. REPORT TYPE AND DATES COVERED Final Report - January 1999 to February 2000		
4. TITLE AND SUBTITLE Design Concepts for Automating Maintenance Instructions.		5. FUNDING NUMBERS C - F33615-99-D-6001 PE -62202F PR -1710 TA - D0 WU -09		
6. AUTHOR(S) Norman Badler, Charles Erignac, Patrick Vincent, Edgar Sanchez, Edward S. Boyle, Jeffrey L. Wampler, John D. Ianni				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Pennsylvania      TASC Inc.      Boeing Company Center for Human M&S      2555 University Blvd.      S0343067 200 South 33rd Street      Fairborn, OH 45324      P.O. Box 516 Philadelphia, PA 19104      St. Louis, MO 63166		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Human Effectiveness Directorate Deployment and Sustainment Division Air Force Materiel Command Sustainment Logistics Branch Wright-Patterson AFB OH 45433-7604		10. SPONSORING/MONITORING AGENCY REPORT NUMBER  AFRL-HE-WP-TR-2000-0088		
11. SUPPLEMENTARY NOTES  AFRL Monitor: Jeffrey L. Wampler, AFRL/HESS, (937)255-7773				
12a. DISTRIBUTION AVAILABILITY STATEMENT  Approved for public release, distribution is unlimited.		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) This task was performed under the Technology for Readiness and Sustainment (TRS) contract (F33615-99-D-6001) for the Air Force Research Laboratory (AFRL), Sustainment Logistics Branch (HESS) at Wright-Patterson AFB, OH. The period of research covered the time period 29 January 99 through 29 February 00. The primary objective of this task was to design, develop, and demonstrate a conceptual framework to support the automated validation and verification USAF technical orders (TOs), specifically job guide procedures for maintenance tasks. The outcome of this effort was the definition and demonstration of a conceptual framework for an application or system that could be used by TO authors and Air Force personnel to help validate and verify the safety and accuracy of job guide procedures in maintenance TOs. In addition, this task also produced a core set of critical research and development tasks considered essential to overcoming important barriers to the development of an automated TO validation and verification system or application.				
14. SUBJECT TERMS Technical Orders      Human Modeling      Human Simulation Product Data Managers      Natural Language Generation (NLG) Computational Linguistics      Automated Validation			15. NUMBER OF PAGES 78	
			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  UL	

**THIS PAGE LEFT INTENTIONALLY BLANK**

## FOREWORD

This research task was performed under the Technology for Readiness and Sustainment (TRS) contract (F33615-99-D-6001) for the Air Force Research Laboratory (AFRL), Sustainment Logistics Branch (HESS) at Wright-Patterson AFB, OH. The period of performance spanned one year starting 29 January 1999. The objective of this task was to develop and demonstrate a framework that can support the automated validation and verification of aircraft maintenance Technical Orders (TOs).

The research team examined all stages of TO generation to determine which tasks most warranted further research. From that investigation, validation and verification of appropriate, safe, and correct procedure steps emerged as the primary research target. This process would be based on available computer-aided design (CAD) data, procedure step ordering from existing sources, and human models. This determination was based on which tasks could yield the greatest impact on the authoring process and offer the greatest potential economic benefits. The team then developed a research roadmap and outlined specific technologies to be addressed in possible subsequent Air Force research tasks.

To focus on the potential technology integration of the validation and verification component into existing or future TO generation procedures, we defined a demonstration scenario. Using the Front Uplock Hook assembly from an F/A-18 as the subject, we examined task procedure steps and failures that could be exposed by automated validation tools. These included hazards to personnel, damage to equipment, and incorrect disassembly order. Using the Parameterized Action Representation (PAR) developed on previous projects for actions and equipment behaviors, we characterized procedure steps and their positive and negative consequences. Finally, we illustrated a hypothetical user interface extension to a typical Interactive Electronic Technical Manual (IETM) authoring system to demonstrate how this process might appear to the TO author.

## TABLE OF CONTENTS

Section		Page
1	Introduction .....	1
	1.1 Background.....	1
	1.2 Scope .....	2
	1.3 Technical Orders .....	3
	1.4 TO Validation and Verification .....	5
	1.5 Potential Benefits of a TO Validation Tool.....	6
2	Part I: Survey of Current Research and Technologies.....	8
	2.1 Introduction .....	8
	2.2 Technical Domains Related to Technical Order Validation .....	8
	2.2.1 Product Data Management .....	8
	2.2.2 Virtual Design Engineering and Manufacturing.....	9
	2.2.3 Concurrent Engineering .....	10
	2.2.4 Human Models.....	11
3	Part II: AMI Conceptual Framework.....	14
	3.1 Framework Overview.....	14
	3.2 Framework Components .....	15
	3.2.1 Validation by Simulation.....	17
	3.2.2 Proof of Soundness .....	17
	3.2.3 Framework Functions.....	17
	3.2.4 Procedure Diagnosis.....	18
	3.2.5 Action Failures and Hazards.....	19
	3.2.6 User Interface.....	19
	3.2.7 Translation to PAR and Execution of Textual Orders .....	20
	3.2.8 Model Storage and Importation .....	21
	3.3 Source Databases .....	23
	3.4 Framework Processes.....	24
	3.4.1 Editing Process.....	24
	3.4.2 System Dependent Validation .....	25
	3.4.2.1. Geometric Domain .....	25
	3.4.2.2. Physics-based Domain .....	25
	3.4.2.3. Agent Domain .....	26
	3.4.3 Human Factor Dependent Validation.....	28
	3.5 Real-World Applications.....	28

Section	Page
3.5.1 Publishing Simulation Graphics .....	28
3.5.2 Model Importation, Building, and Maintenance.....	29
3.6 Authoring and Simulation Library.....	30
3.6.1 Task Scenario.....	30
3.6.2 Actionary .....	31
3.6.3 3D Models .....	33
3.6.4 Human Models.....	33
3.6.5 Physical Behavior Models.....	33
4 AMI Conceptual Demonstration.....	35
4.1 Introduction .....	35
4.2 Conceptual Demonstration Development .....	35
4.2.1 Initial State.....	35
4.2.2 Role of Technical Order .....	37
4.2.3 Script Authoring.....	39
4.2.4 Support Data.....	43
4.3 Summary.....	44
5 AMI Research Roadmap .....	45
5.1 Critical Technologies .....	45
5.1.1 Human Models.....	45
5.1.2 PAR-Based Agent and Specific Skills .....	46
5.1.3 Natural Language Technologies .....	47
5.1.4 Semi-Qualitative Simulation .....	49
5.2 Outline of Future Task Efforts - FY2000.....	49
5.2.1 Represent Procedure Steps with PAR .....	49
5.2.2 Validate TOs through Automatic Generation of Virtual Motion Simulation.....	50
5.2.3 Determine Knowledge Representation Requirements .....	50
5.2.4 Create PARs Through Human Performance Motion Capture and Semantic Analysis.....	51
5.3 Brief Outline of Future Task Efforts - FY2001 .....	51
5.3 Development Program.....	52
References .....	54
APPENDIX: IETM Authoring Requirements	

## LIST OF ILLUSTRATIONS

Figure		Page
1	IETM Document.....	4
2	AMI Framework.....	14
3	Human Model at Work .....	16
4	Translation and Execution of Natural Language Orders .....	21
5	Translation and Simulation of a TO .....	22
6	3D Model Decimation .....	29
7	Open Action Taxonomy.....	32
8	F/A-18.....	36
9	General View of the Work Area .....	37
10	Front Uplock Hook Assembly .....	38
11	First Version of Subtask Sequence.....	40
12	Second Version of Subtask Sequence .....	42
13	Third and Final Version of Subtask Sequence.....	43
14	User Interface .....	46
15	Text and PAR Script Input.....	47
16	Simulation and Status Feedback .....	48
<b>Table</b>		
1	Geometric and Non-Geometric Models.....	30

## **GLOSSARY**

AMI	Automated Maintenance Instructions
API	Application Program Interface
ASL	Authoring/Simulation Library
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
COTS	Commercial Off The Shelf
FUH	Front Uplock Hook
HFDV	Human Factors Dependent Validation
IETM	Interactive Electronic Technical Manual
KBE	Knowledge-Based Engineering
LSA	Logistic Support Analysis
NLG	Nose Landing Gear
NLP	Natural Language Processing
PAR	Parametrized Action Representation
PDM	Product Data Management
PDM	Product Data Models
SD	Source Databases
SDV	System Dependent Validation
TO	Technical Order
VR	Virtual Reality
XML	Extensible Markup Language

**THIS PAGE LEFT INTENTIONALLY BLANK**

# SECTION 1

## INTRODUCTION

### 1.1 BACKGROUND

Technical Orders (TOs) and engineering drawings are the most expensive and arguably the most important data acquisitions made in the support of a weapon system [1]. The production of maintenance TOs, job guides, illustrated parts breakdown manuals, and other publications used to support the maintenance of weapon systems and support equipment accounts for a significant percentage of the total procurement cost for weapon systems. In fact, informal estimates project that this cost may be as high as 15% of the total acquisition cost of new weapon systems [2]. Based on this information, it only seems logical that there should be significant impetus for investing in research activities that focus on identifying and maturing technologies that can make the TO authoring process more efficient and cost effective.

AFRL/HESS has sponsored several research efforts over the past few years to investigate specific technologies that may be considered elements of a unified solution set for automating the production of TOs. These efforts were conducted under the Automation of Maintenance Instructions (AMI) program through the the AFRL Logistics Technology Research Support contract (F41624-97-D-5002). These research tasks included the following:

- Technologies for Maintenance Instructions - Delivery Order 8.
- Product Modeling Technologies for Automating Maintenance Instructions - Delivery Order 14.
- Maintenance Action Representations - Delivery Order 17.

While these research tasks provided some important insights into core technologies related to the authoring of maintenance instructions, it became evident that a more focused effort was needed to define a framework that could effectively support the automated production of maintenance instructions (AMI), and formulate future research objectives.

## 1.2 SCOPE

At the outset of this task, three primary research objectives were pursued. The first objective was to identify and evaluate current research and technology trends applicable to the automated production of maintenance instructions. The second was to define an engineering computing framework that could effectively support the automated production of aircraft maintenance instructions, or TOs. The third objective was to develop a storyboard (concept) demonstration that could be used to explain the main components of the computing framework defined earlier. The main components included existing technologies and methods, as well as technology gaps. Existing technologies included CAD, computer-aided manufacturing (CAM), computer-aided engineering (CAE), and current authoring tools. Methods included concurrent engineering design and development practices. Technology gaps represented important, unsatisfied research needs that lie in the critical path to successfully develop and demonstrate a prototype system for the automated authoring of maintenance TOs.

As the task progressed and a more detailed analysis of the current TO authoring process was conducted during two workshops, the scope of the research task was refocused on the TO validation and verification process. The decision to refocus the research effort was made for primarily three reasons. First, no tools or applications currently exist to support this very important step in the TO authoring process. Second, the prospects for successfully addressing and demonstrating all aspects of the current TO authoring process in an automated system was considered unrealistic, given the timeframe and projected budget outlays for the AMI program. Finally, major breakthroughs would be required in key technology and research areas that support AMI, including breakthroughs in geometric reasoning to get at the function logic of parts/components, the assembly/disassembly sequence related to the removal and installation of components, etc.

Thus, rather than aiming for complete automation of the TO authoring process, the decision was made to focus future research efforts on high-payoff technology areas that could support the automated validation of TO maintenance procedures. The objective would be to provide the TO author the capability to visualize the performance of a candidate procedure to determine whether a procedure, as written, is reasonable and safe to perform. In

essence, are procedure steps logically ordered and clearly defined, or is there a better way to arrange the steps? This type of validation should be possible with existing human modeling technology.

### 1.3 TECHNICAL ORDERS

A TO is a document that instructs technicians how to perform an operational or maintenance task on a weapon system. It is an "order" because the actions it describes are formulated as individual orders. The technician is expected to carry them out as if they were issued by a supervisor. Many types of TO publications are produced to support the operation and sustainment of a weapon system. These publications include operations manuals, job guides, illustrated parts breakdown, support equipment manuals, software manuals, etc. The focus of the research conducted under this effort, and preceding AMI research, is on maintenance instructions found in job guides that are used by Air Force personnel to repair and maintain a weapons system. Hence, the reference to a "TO" throughout the remainder of this report will refer to maintenance instructions found in job guides.

A TO starts by defining a set of input conditions that describe the initial state of the maintained system. It also identifies the necessary spare parts, special tools, and personnel required to perform the task. The remainder of a TO is an ordered sequence of actions that the maintenance technician(s) must perform to safely and effectively complete the task. Where applicable, warnings and caution messages are also specified to point out potential safety hazards associated with the task.

Until recently, TOs were only available in a printed form featuring technical drawings to complement textual information. They are now available in several electronic forms:

- **Logistic Support Analysis (LSA):** LSA is an electronic database of maintenance tasks. This is a purely textual representation. A task is modeled as a sequence of subtask records. Each record contains a narrative description and miscellaneous data.
- **Interactive Electronic Technical Manual (IETM):** An IETM is meant to replace printed material with portable computers suitable for use in the field. An IETM TO is an interactive electronic document combining text and graphics (see Figure 1). Aside from enabling browsing of large numbers of technical manuals, IETM allows step-by-step

tracking of task execution, adapting the task sequence to the actual situation, and performing electronic checklists.

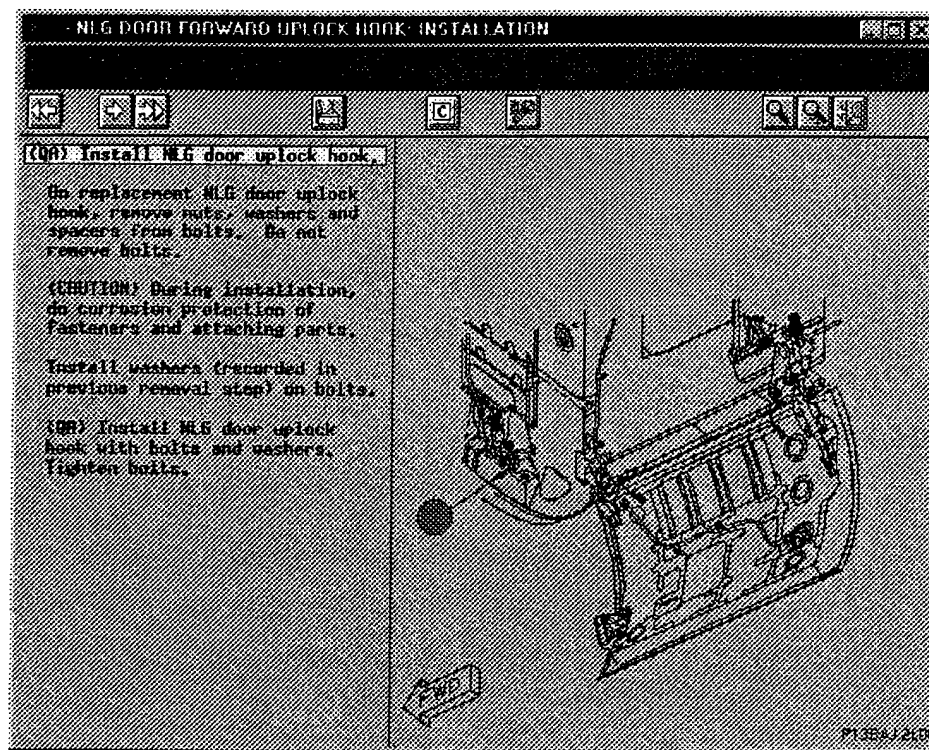


Figure 1. IETM Document.

To date, maintenance manuals have been manually authored in printed or electronic form. Although electronic systems like IETM provide major assistance, the authoring process is still labor intensive (see Appendix). The author must gather information from multiple sources (engineering, vendors, maintenance, etc.) and in various forms (printed material, CAD files, LSA records, etc.). When this is not enough, the author must consult experts (such as design engineers). Once the research step is completed, the author compiles the information into a TO.

The author draws from personal experience and researched material to produce the initial conditions and the sequential steps of the maintenance procedure. If an IETM-like system is used, the elements are structured as database references, interactive text, and annotated graphics. The level of detail in the TO (subtask breakdown and the associated set of notices and warnings) must be explicit enough to allow Air Force technicians to safely complete the task.

TO authoring is similar to writing a computer program in that each order contains precise semantics. Its execution modifies the state of the world in a predictable way, based on standard human factors and technical personnel training and performance. An order is inserted in a TO's sequence with the assumption that the orders preceding it will be carried out successfully and establish the expected execution context necessary for the inserted order to perform correctly. In other words, when placing an order in a TO, one assumes that its input condition will be satisfied by the initial state of the scenario or the accumulated effect of the preceding orders.

#### **1.4 TO VALIDATION AND VERIFICATION**

The primary purpose of the Air Force TO validation and verification process is to ensure that procedures can be effectively and safely performed by personnel with the planned skills and training. TO validation is the process the weapon system manufacturer performs to ensure that maintenance procedures can be performed as written, by properly trained personnel, in a safe manner. TO verification is the confirmation process the Air Force uses to ensure the adequacy and accuracy of TOs. While the verification process is equally important, the main focus of this research effort was on developing a conceptual framework for developing an automated tool to support the validation of TOs by the weapon system manufacturer. It is possible that the same framework could be extended to include the Air Force verification process, but this report does not purport this view. The current methods of TO validation include desktop validation and validation through demonstration. In general, a desktop validation is accomplished using engineering drawings and other supporting technical documentation. It is typically used in cases where existing procedural data may be validated by comparison to source data (e.g. engineering data, etc.) if the procedure was originally validated by demonstration on equipment of identical configuration. The more prevalent method for TO validation is through demonstration on a physical system. In general, TO procedures that are detailed in a job guide must be validated through demonstration if the procedure falls into any one of the following categories:

- New procedures.
- All software-dependent procedures for any new production buy or for any software update.

- Modified system procedures that have encountered problems.
- Procedures in which steps are re-sequenced, added, or deleted to the extent that any hookup, operation, or indications are affected.
- Procedures validated by demonstration on equipment of a different configuration.

The framework that will be discussed in Section 3 of this report focuses on the conceptual design of an automated tool or application that would support both types of TO validation, namely desktop validation and validation through demonstration.

## **1.5 POTENTIAL BENEFITS OF A TO VALIDATION TOOL**

The current TO validation process could be streamlined in some respects through the use of a desktop validation tool or application that would remove some of the constraints a TO author must operate under in the current process. First, a demonstration validation requires that a physical system (i.e., the aircraft) must be in place and available to the author to perform the validation of a procedure. This requires that authors must schedule time to get access to an aircraft just to view the work area. It may also require assembly line/flight testing activities to be temporarily halted to perform the validation. This may not be efficient and effective from a scheduling standpoint, particularly with regard to manufacturing concerns. Second, the TO author cannot perform the actual maintenance procedure or task. A union technician must perform the task. If the author cannot fit in the confines of the work area to observe the performance of the task, the author must rely on the technician to identify problems or improvements with the procedure. This can be difficult in some cases because the aircraft must be through the assembly process sufficiently to provide the "users view" of the task.

Third, many times inspection seals must be broken to perform a procedure. This requires re-inspection of the area and new inspection seals. Finally, accidental damage to aircraft may be induced during the physical validation of the maintenance task (e.g., dropped parts, cross-threaded hardware, etc.)

Although a TO validation tool should not be purported as direct replacement for the physical validation of TO procedures, it could help alleviate some of these problems by providing the TO author the capability to validate selective tasks even earlier in the weapon

system development process. This may help reduce the time and effort associated with scheduling aircraft and personnel resources required for the validation task, and possibly serve as a legitimate, certifiable method for validating specific maintenance tasks without the need for follow-up validation on a physical system.

The remainder of this report focuses on synthesizing the state of the art in key technology areas related to the development of such a validation tool. It also describes the engineering and computing framework that would be required to support the design, development, and implementation of a TO validation tool. Finally, the development of a conceptual demonstration is discussed. The conceptual demonstration is intended to explain the core functions and requirements of such a tool and convey the important, unmet research needs that should be addressed to foster the development of a prototype TO validation tool.

## **SECTION 2**

### **PART I: SURVEY OF CURRENT RESEARCH AND TECHNOLOGIES**

#### **2.1 INTRODUCTION**

We identified four technological domains related to TO authoring, and more importantly TO validation. These domains included (a) product data management (PDM), (b) virtual design engineering and manufacturing, (c), concurrent engineering, and (d) human form modeling. Virtual design tools produce the models necessary to validation simulations. PDM systems deliver this information to the validation application. Concurrent engineering practices and supporting software, including human modeling applications, allow authoring and validation to be pursued while products are still in their design stage.

#### **2.2 TECHNOLOGY DOMAINS RELATED TO TECHNICAL ORDER VALIDATION**

##### **2.2.1 Product Data Management**

PDM [3] is an enterprise-wide framework aimed at modeling and tracking all of the data concerning produced goods and services as well as related processes. PDM was initially developed to organize and store data pertaining to engineering activities in a company producing industrial, transportation, and consumer goods. A PDM system stores the design, manufacturing, and maintenance data for each product in a uniform framework, and also manages the processes critical to a product's life cycle. PDM is increasingly used on a larger range of products such as buildings, bridges, factories, cable networks, software, and services. Since PDM is a general framework, it can be used for any production activity. Furthermore, its scope is wide since anyone who deals with products consumes or creates PDM data.

Product data generally consists of specifications, configuration data, CAD/CAM/CAE files, manufacturing data, revisions, and maintenance manuals. However, it also extends to financial and marketing documents. In any case, a PDM system can be scaled up or down to manage specific disciplines of a company. PDM covers the entire life cycle of a product - design, testing, manufacturing, support, and maintenance. In particular, it supports concurrent engineering functions.

The core of a PDM system is its *data vault*. This *metadatabase*, or database of databases, inventories every product datum in the system by maintaining an associated *metadata* record. It contains format, location, ownership, security, and revision information. The data itself is stored in an application-specific database. CAD/CAM/CAE software and other applications can directly access a PDM system to store and retrieve data.

The major functions of a PDM system are listed below:

- **Uniform Data Referencing and Access:** Objects or data records can be referenced with a unique name by different databases. They can also be accessed uniformly by various applications.
- **Process Management:** A PDM system can model and manage data workflow. It can trigger and monitor specification, design, approval, revision, or any other business process.
- **Data Administration:** The content of PDM metadata allows access privileges (security) to be managed, authorship to be recorded, and multiple versions of a single datum (revision control) to be tracked. Furthermore, all the data concerning a product can be transferred, backed up, and archived as a single block.

Many vendors offer PDM solutions. The Object Modeling Group developed a *PDM Enablers* specification [4] to promote interoperability between different PDM systems. For more information on PDM see CIM, 98.

### 2.2.2 Virtual Design Engineering and Manufacturing

For decades, weapon systems have been modeled with CAD software. More recently, emerging technologies have supported other related processes such as engineering, prototyping, and manufacturing. Off-the-shelf CAE software can model and test the mechanical, thermal, and structural properties of a product before its first prototype is even built. Similarly, CAM software is used to develop molds, stamping tools, weaving patterns, and machining paths from CAD models.

The advent of virtual reality (VR) has spawned the development of virtual prototyping applications that enable a product to be assembled, inspected, and tested in a

virtual world. Some aspects of designs can be tested for maintainability and human factors while they are still on the drawing board. More generally, interactive 3D visualization is used at every step of a design cycle to view single parts, animate assemblies, visualize scientific data, and create marketing and technical documentation.

Current CAD models are *parametric*. Along with their geometric description, they include dimensioning constraints that relate different parts. For example, the diameter of a shaft and the bore it runs through can be constrained such that the bore is updated when the diameter changes. These parametric models allow encoding some of the design intent into the CAD data.

Knowledge-based engineering (KBE) goes further in this direction by capturing design expertise. With a KBE approach, a design bureau can record the lessons learned from past projects and reuse them in the future. For example, a KBE system can help choose an energy source or manufacturing process. A KBE application is mostly an expert system.

It is now possible to go “virtual” through most of a product’s design. Software vendors sell integrated computer-based solutions that support virtual design and manufacturing [5]. Nevertheless, “hands on” virtual prototyping applications, such as technical order validation, remain experimental [6].

### 2.2.3 Concurrent Engineering

Traditional product design is a chain of sequential development stages going from conceptual design to a finished product. Each stage deals with a specific aspect of the product, such as engineering, manufacturing, prototyping, testing, and servicing. When a design fails to satisfy the constraints of a given stage, it is sent back to one of the earlier stages for redesign. This process can be very costly, because most design defects are discovered during the later stages when changes are more expensive to correct.

Concurrent engineering [7] attempts to reduce development costs by accomplishing each development stage in parallel. This method is particularly challenging because it relies on the collaboration of specialized teams. Aside from the organizational difficulties, concurrent engineering relies on collaborative design environments to share models and ideas.

These environments are built upon available CAD/CAM/CAE software integrated within a PDM system. In particular, the PDM system allows the versioning and review processes to be shared across design teams. Along with these core applications, simpler and “lighter” CAD model visualization tools are being used to communicate the design’s shape and function to a wider audience within and outside a company. These tools allow models to be annotated to simplify review processes. Some are geared toward collaborative design sessions, allowing remote users to share the same virtual space where they can manipulate, modify, and annotate a 3D assembly in turn while communicating through voice or video links [8].

TO validation fits within a concurrent engineering process as a co-design activity. Concurrent engineering can help authors identify maintainability flaws early in the design process and prevent cost overruns and delays later during physical prototyping.

#### 2.2.4 Human Models

Computer-graphic human form models (referred to as *human models*) have been available for 25 years. Significant developments occurred during the last decade, as computer power and three-dimensional graphics improvements have led to interactive models with sufficient biomechanical accuracy to allow their use as ergonomic evaluation surrogates. These models allow figures with anthropometric variations based on a sample population and represent body shape with more or less smooth polygonal surfaces and adjustable joints. The more capable human models provide mechanisms to control the actions of the model, for example, through a walking algorithm, inverse kinematics limb reach, and automatic satisfaction of balance and other postural constraints. Additional improvements include analytical reports on strength, visibility, reach zones, comfort zones, and lifting hazards.

As human models mature, they appear to be departing from stand-alone systems and assuming a more integrated role in the design engineering process. This requires that they interface with CAD models and the design process at increasingly early stages of the product life cycle. Where human models were used primarily by human factors engineers, they are now used by engineers throughout the design process. This shifting of responsibility should affect both the engineering process and the need for human modeling software. Engineers

will be able to easily perform cursory human use analysis as part of form, fit and function analyses.

Features important to manipulation and maintenance (as well as manufacturing) should find their way into the Product Data Models (PDM) to be shared across the engineering enterprise. We have decried the lack of what we called *maintenance features* that would allow a human model to understand a device in terms such as handles, connections, contents, and even function. As designers use human models to evaluate their designs, we hope they will note these features, sites, parts, and contents in the PDM databases so that such information can be used elsewhere by the design team. Besides helping the human factors analyst, such annotations will clearly help the TO author. Annotations that relate part features to CAD features are now inserted manually by TO staff and used by the TO author to create callouts in the graphic images that accompany and amplify the TO steps and text. It seems inefficient to ask the TO authoring staff to insert information that is already known to the design engineers. Although it is not within the research scope of this project, we hope that PDM systems will emerge to reinforce good labeling habits. The alternative – directly automating the determination of maintenance features from the CAD data – is a fascinating research project but appears unlikely to be economically justifiable in actual practice.

The second effect of this shift in responsibility is related to the design of human modeling software. Early human models required that each joint be posed manually, sometimes through a tediously-created and non-intuitive data file. Interactive systems ameliorated some of these problems, but not enough: only the development of robust and flexible inverse kinematic algorithms made human models usable. As the human models are integrated into enterprise-wide CAD systems, users will want better software tool integration and easier-to-use interfaces. A proven approach to the former is through a software library and application program interface (API) that allows another system (such as a host CAD package) to access and control the human model. The user interfaces for the human model, in turn, are expected to resemble those of the host software. The best example of this situation is the *Jack Toolkit*, from Engineering Animation, Inc. The toolkit interface lives in the host software, which encourages user interfaces that are as simple and straightforward as possible. We expect that in the future, other human models will have to adopt this software approach

to live across multiple systems. The alternative is to wed the human model with the CAD system, but this becomes difficult to extend outside the CAD vendor's environment.

In general, human models can potentially aid the TO validation process. Since the primary role of the TO author is to create instructions for real human maintainers, such instructions should be first tested on synthetic maintainers within the given CAD environment. Accordingly, a level of control as close as possible to the actual instruction level will greatly aid the TO creation and validation process. In particular, a user interface that supports the expression of task instructions in natural language (as found in TOs) will reduce the need for the TO author to be both animator and programmer.

## SECTION 3

### PART II: AMI CONCEPTUAL FRAMEWORK

#### 3.1 FRAMEWORK OVERVIEW

The TO authoring process, as described in the Appendix, is a labor-intensive process. The author must compile information from existing technical and LSA manuals, engineering drawings, and various other sources to produce a specific maintenance procedure. Once the procedure is initially authored, it must be validated by comparing it with design documents and safety guidelines, or by actual demonstration on a physical system. If a problem is detected, the author modifies the TO and re-validates the procedure. The cycle is iterated until the maintenance procedure is successfully validated. Once validated, TOs are published by the airframe manufacturer (vendor) and delivered to the Air Force for follow-on verification of the adequacy and accuracy of TO procedures.

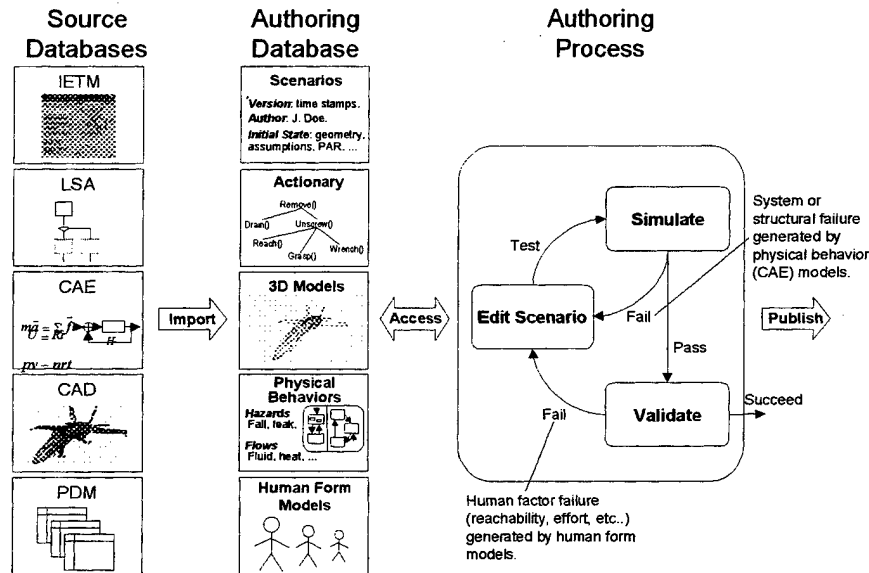


Figure 2. AMI Framework.

Our AMI framework (see Figure 2) will define the concepts and processes necessary to implement a desktop TO validation and verification tool that could support the TO author

in the validation process, and ultimately reduce the time and cost associated with the validation of TO procedures. Such a tool would test a maintenance procedure by simulating it in a 3D virtual environment. The assessment would cover both human factors and system domains. A variety of human models would be used in the simulation to ensure that the ergonomic requirements (visibility, accessibility, effort, and exertion) are within occupational standards. Likewise, the simulation would test whether the procedure is feasible with respect to the maintained systems. Using physics-based models, the simulation would detect unfeasible or hazardous actions.

The main benefit of our framework is to complement current TO authoring systems with a desktop validation tool that would enable rapid prototyping and development of maintenance procedures. The first iterations of the traditional “generate and test” approach would take place entirely at the author’s workstation: the author would edit and simulate a procedure until it passes a validation test. Only a few final validations would need to be done using manual methods such as engineering review or demonstration.

Although the framework is meant to complement existing TO authoring tools, it would make sense to specify the validation tool as a stand-alone application with authoring capabilities. Furthermore, entities could also use the tool in their verification process. The validation may also be stored and replayed. We eventually expect manufacturers and regulating authorities to endorse or certify maintenance procedure validation tools for adhering to industry and government standards.

It is difficult to envision the precise boundaries of the validation tool’s specifications on a conceptual level. However, it is clear that the validation tool would have to interface with other engineering or authoring applications. We anticipate that such integration would be achieved with a PDM system, which could potentially position the maintenance authoring process as a co-design activity early in a product development cycle.

### **3.2 FRAMEWORK COMPONENTS**

Although existing technology does not allow TO authoring to be automated, we propose to simplify the task by automating the validation step. Our framework relies on simulation to apply a validation process by demonstration in a desktop VR environment. We

can simulate a maintenance procedure from a system or human factors perspective to recreate the working conditions of an actual technician (see Figure 3). As in a live validation, the simulation will be successful if the procedure can be completed with the desired result and without hazardous consequences.



Figure 3. Human model at work.

The framework is meant to bring the benefits of verification to the author's desktop. Although it would not serve as a total substitute for physical testing, we expect that most hazards and unfeasible subtasks would be detected. With such a validation tool, an author would be able to repetitively edit and test procedures until they pass desktop validation.

In addition, the 3D representation of the maintained system would enable the author to gain considerable insight into the geometric complexity of the task. The author could survey the scene through the eyes of the virtual technician or from any other point of view. Furthermore, transparency effects and swept volumes could help locate hidden components and evaluate motions in confined spaces.

The simulation will also encompass physical system behaviors. By modeling the system's behavior as it is being operated or maintained, hazards or system-related failures can be detected. Furthermore, model-based reasoning techniques would allow automatic annotation of the sequence of events that led to a hazard or failure.

These geometric and system debugging functionalities will greatly simplify analysis and repair of a procedure.

### 3.2.1 Validation by Simulation

The framework possesses the ability to simulate a maintenance procedure that would be a viable replacement for live validation. As in validation by actual demonstration, the validation tool will be deemed sound but not complete. In other words, although all the behaviors it produces are realistic (not spurious), it cannot prove that any strict interpretation of the procedure by any technician under any compliant input conditions will be successful.

The soundness of a simulation relies on the fidelity of the model with respect to the system it represents. High fidelity models are complex and require large computational power, which might not be available to produce interactive simulation. It is the responsibility of the modeler to find an acceptable compromise between fidelity and speed.

### 3.2.2 Proof of Soundness

The only thing an actual or virtual validation by demonstration can prove is that a maintenance procedure is not sound through the detection of action failures or hazards. In other words, a procedure is believed sound until proved unsound. An action fails when its expected effect cannot be observed or when its input conditions cannot be achieved. A hazard is an unwanted physical process conducive to property damage or injury.

System-related failures and hazards independent of the human model will be detected during the first run of a simulation. However, a simulation must be run with different human models that represent the technician population. If all simulations are successful, the procedure can be deemed sound. This need for multiple runs is synthesized in the Simulate and Validate states shown in Figure 2.

### 3.2.3 Framework Functions

The framework's core process is a simulation generated by a simulator. The simulation corresponds to the execution of a maintenance procedure. However, the validation tool must also provide the following functions:

- **Editing:** Direct editing of maintenance procedures is necessary regardless of whether the tool is running stand-alone or if the author imports procedures from other authoring applications. In the first case, the author must be able to input a whole procedure by using

the system's model library to compose it. In the second case, imported procedures must be cleaned up and dressed up to cover the simulation domains that were ignored by the source application, such as geometry. Editing must be validated by syntactic and semantic checks that allow the author to verify the correctness of the procedure model's form and content.

- **Debugging:** In the case of an action failure or hazard, the validation tool should stop the simulation and help the author isolate the problem. This is the first step toward correcting the procedure. Although geometric reasoning is still too complex for online debugging, causal reasoning of system behaviors is not. A computer can easily reason the sequence of events that triggered a procedural failure and assist a author in isolating the cause. Nevertheless, except for trivial errors, we should not expect the validation tool to automatically repair a procedure.

Although not the focus of this research, the validation tool could also be used to produce media for electronic technical or training manuals, including animations, still images, or interactive simulations. We did not list *publishing* as a function, because it is not strictly necessary for validation. Nevertheless, we should expect publishing to be available under some form in the framework. For example, it could be used to communicate system failures to a design team.

### 3.2.4 Procedure Diagnosis

Procedure diagnosis takes place during debugging when an author encounters an error and tries to determine the cause of a faulty procedure. Unlike traditional system diagnosis (such as aircraft fault isolation), procedural faults can be attributed to either inaccurate or ambiguous TO procedures, or an error on the maintenance technician's part in not following a validated, published TO procedure. The latter problem is not going to be resolved by a TO validation tool, and is outside the scope of this research. However, the former case can be addressed to some extent by a TO validation tool. For example, assume the maintenance technician is following the step-by-step TO procedures for a task in a suystem. If a component burns out while performing the maintenance task, the relevant failure is not an actual component failure, but rather an induced failure that may have been by an omitted step in the TO procedures (e.g. not turning power off to the aircraft or system that caused the

component to short out) The flaw in the TO procedure could very well be attributed to the author's lack of insight into the different types of hazardous conditions associated with performing the task. In this case, a validation tool might help improve this situation by allowing the TO author to simulate and visualize different scenarios and conditions for accomplishing the task to determine the safest set of conditions, as well as the proper sequence of steps for performing the task

The validation tool is similar to a software development environment. It records a simulation trace, which is analyzed for debugging purposes. The models used in the semi-qualitative simulator can be automatically analyzed along with the simulation trace to help the author locate a problem [9]. The system assists the author by answering standard queries about the function of a device or the factors influencing its behavior. Because of their explicit representation, PAR actions can be included in this reasoning. These questions can also be used as online technical documentation.

Although these self-documented models might help the author understand how a device works, a minimum engineering background will be required to use the tool efficiently.

Current limitations in geometric reasoning do not allow similar debugging facilities to isolate geometric faults. Therefore, only system-related faults can be semi-qualitatively isolated.

### 3.2.5 Action Failures and Hazards

Some hazards and action failures could be turned off or ignored by the author to focus on specific aspects of the validation. However, if too many of these events are ignored, the simulation might deviate from its expected realistic behavior and compromise the fidelity of the validation.

We recommend a tight *edit-validate-debug* cycle where the author aborts validation at the first error. This argument provides an extra incentive for including substantial editing and debugging functions in the validation tool.

### 3.2.6 User Interface

The nature of the framework's user interface remains to be chosen. We expect the tool to use a graphical interface through which a maintenance procedure could be represented in three possible modes:

**Graph:** A procedure is represented as a flowchart. The author builds a procedure by dragging, dropping, and connecting icons representing actions and sequencing constructs (used in LSA and IETM).

**Script:** A procedure is a script written in a specific high-level programming language. The author must be familiar with the language to key in the procedure.

**Free Form Text:** The author types or dictates a procedure in plain English (natural language). The interface translates the text in an adequate internal representation (script or graph). Reliable natural language processing (NLP) should be available in near- or long-term. Aside from providing a user-friendly interface, NLP would allow legacy TOs to be imported in a textual or semi-structured form.

All three modes could be combined or layered to offer multiple levels of representation. The graph and natural language modes are the most user-friendly. However, current technology can only deliver a combination of script and graph modes. We propose using the PAR [10] language for scripting human actions.

The user interface should also provide the ability to navigate and modify the virtual world in which the procedure takes place. Ideally, the objects in the scene should be "smart." Their relative geometric positions should be reflected into a corresponding physics-based model. For example, if a plug is inserted in a socket, the simulator should establish an electrical contact between them.

### 3.2.7 Translation to PAR and Execution of Textual Orders

One essential requirement of the validation tool is to provide the author with a high-level scripting language to control a virtual technician. This language is defined by a set of complex procedural actions and composition operators to sequence them. Each action can be defined by a sequence of lower-level actions, or a direct call to one of the agent's basic skills.

Since our intelligent agent must behave like an average technician, any action vocabulary must be equivalent in both the semantics and level of abstraction to the actions conveyed in textual orders. In a PAR of this scripting language, the high-level action vocabulary, as well as all of the underlying actions are stored in an *Actionary*, which represents the procedural knowledge of the software agent controlling the virtual technician.

Regardless of whether translation from text to script is manual or automated, the closer the Actionary is to the usual TO vocabulary, the better the translation will be. Ianni's specifications [11] exemplify the basic action vocabulary of a virtual technician.

If the translation is manual, the author is responsible for performing a realistic semantic mapping between the text and the actions composed in the corresponding script. If the translation is automated, the validation tool could assess the clarity of the order by exposing potential ambiguities or inconsistencies.

Figure 4 depicts the whole translation and execution process. The author creates a new subtask and fills it with a textual order. The order is translated into a compact PAR script, which is expanded into a set of sub-actions during execution. The software agent performs the primitive actions of the expanded plan. If the effect of the action is hazardous, the author revises the order and starts over.

Translating a procedure consists of scripting each of its subtasks individually. Technically, a whole TO could be modeled as one script made of the sequence of subtasks' scripts. Figure 5 summarizes the different translation steps from a TO to its simulation.

# Natural Language to PAR Conversion and Execution

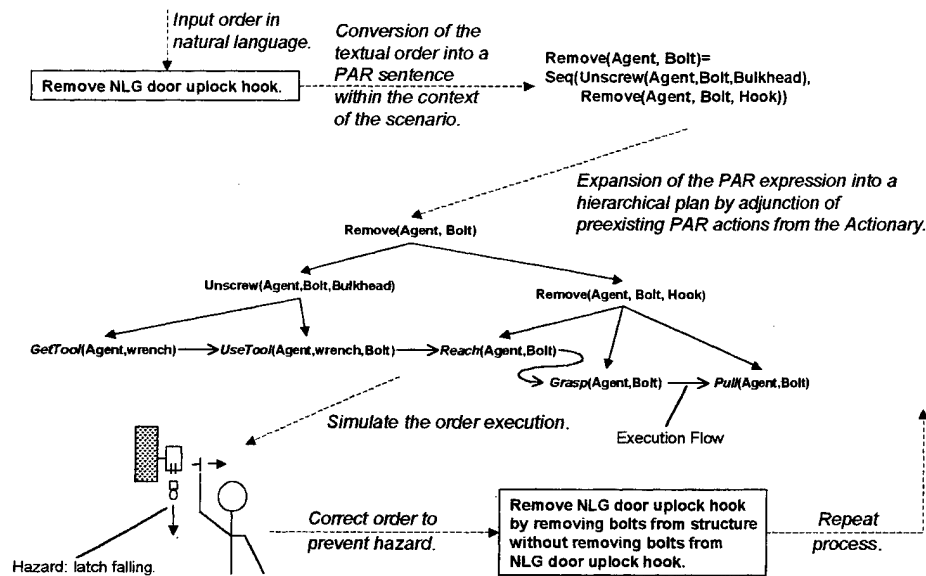


Figure 4. Translation and Execution of Natural Language Orders.

### 3.2.8 Model Storage and Importation

Each procedure is modeled as a *task scenario*. This top-level data structure is the equivalent of a TO. It defines the input conditions of the procedure by specifying initial conditions and the action sequence to perform. The initial conditions refer to the systems, tools, modeling assumptions, human forms, and metadata necessary to set up a simulation.

The simulator uses various modules to generate human and system behaviors, as well as hazards and failures. Each of the modules covers a specific domain of the simulation: geometric, physics-based, and intelligent agent.

The corresponding models are fetched from the *authoring / simulation library* (ASL) to build the corresponding simulation model. The ASL is a “backlot” of models reused across scenarios. The references between models are closed. This means that all the necessary information is in the database. In other words, the validation tool can run as a stand-alone application.

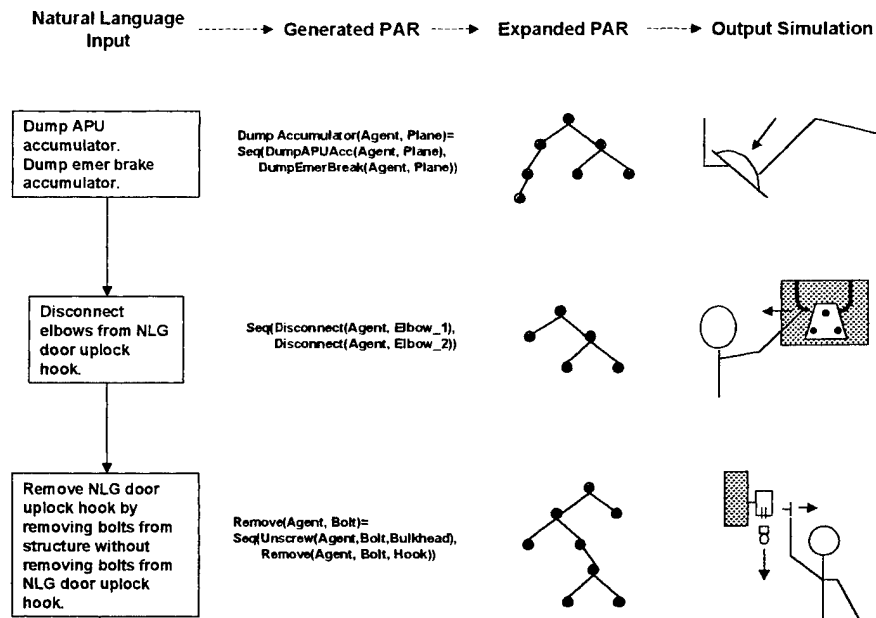


Figure 5. Translation and Simulation of a TO.

Similar to the maintenance procedure author, the framework feeds off source databases to populate the ASL. These databases are maintained by domain specific applications: TO authoring (IETM, LSA); CAE; CAD; and PDM.

The records from input data sources might require specific conversion operations before being stored in the ASL. For example, geometric models must be simplified to allow real-time rendering. The complexity of the importation process is highly dependent on the difference between the format of each particular data source and ASL internal models. Some of these import steps might have to be accomplished manually if the data representations between the source data and ASL differ beyond what can be automated.

We expect the validation tool to interface with each applicable data source (e.g. LSA tables containing task narrative descriptions, etc.) through an enterprise-wide PDM system. This would keep the source data and ASL in synch and trigger the necessary re-validations when an update to a TO procedure occurs.

### 3.3 SOURCE DATABASES

Figure 2 identifies four kinds of source data necessary to support the framework:

- **TO Data:** The TO data contains a description of the maintenance procedure in a format similar to LSA or IETM. Each description contains the input conditions (maintained system, spare parts, number of technicians, etc.) and a step-by-step narration of the procedure. It can be seen as a semi-structured document containing tabular information (various references, etc.), plain text (subtask narration), and pictures (schematics).
- **Computer Aided Design:** CAD data describe the shape and structure of the maintained systems. Once imported, it will be used to supply geometry to the VR system and model system assemblies. Modern CAD models are parametric; they contain information such as position constraints or dimensioning that can be reused, and component geometry and behavior models.
- **Computer Aided Engineering:** In general, a system CAE model is a block diagram that interconnects quantitative component models. The same topology can be reused in the framework. However, models may need to be simplified to perform interactively while remaining realistic. They also must be upgraded with a qualitative layer. Finite-element CAE models do not correspond to our lumped-parameter framework, and are outside the scope of our framework.
- **Other Product Data:** Other product data is all the data used to complete a simulation model for the validation process. It may include serial or part numbers, references to technical documentation, etc.

Under normal exploitation conditions, the validation tool would routinely exchange maintenance procedures with third-party authoring systems. This should take place under the auspices of the PDM system. CAE, CAD, product data, and any other kind of data necessary to maintain the virtual “backlot” would be imported as needed.

### 3.4 FRAMEWORK PROCESSES

Simulation can validate both the systems and human factors aspects of a maintenance procedure. The systems aspect checks the soundness of the procedure regarding the maintained system. The human factors aspect checks that the procedure is feasible and safe to perform for a representative set of human models. The human factors check requires the execution and analysis of several simulation runs using technicians of representative

anthropometry for accommodation analysis. This decoupling allows the author to first concentrate on the systems part of a procedure before dealing with specific human factors. We decompose the validation process in two steps:

*Step 1: System Dependent Validation (SDV).* SDV is a simulation to detect systems failures and hazards. It also detects human factor hazards that are independent of a specific human model. For example, it can detect if the technician is exposed to toxic substances.

*Step 2: Human Factor Dependent Validation (HFDV).* HFDV is a system validation with a specific human model. It detects the human factor hazards or failures specific to the technician model, such as failure to reach or insufficient strength.

The difference between SDV and HFDV lies in the anthropometric and biomechanic characteristics of the technician, which are relevant in HFDV and not in SDV. For a given procedure, the author will have to run at least one SDV and enough HFDVs for an accommodation analysis.

#### 3.4.1 Editing Process

LSA or IETM data does not contain all of the information necessary to produce a simulation. For instance, the system geometry necessary to render computer graphics and collision detection in a human model is not included in LSA task narrative data. The validation-specific part of the editing process must allow an imported procedure to be completed with the adequate data.

If the validation fails, the author can edit the procedure in the source system. However, in order to use the validation tool as stand-alone, or to quickly re-test a modified procedure, the application should support part of the procedure editing process. In particular, the author should be able to edit a subtask sequence as well as its caution and warning messages.

#### 3.4.2 System Dependent Validation

SDV is related to how simulations are performed. The simulation itself emerges from the interaction of dedicated simulators.

A *scene management system* uses the geometric description of the world to render interactive 3D graphics and detect collisions. A physics-based simulator generates systems behaviors. Finally, an *intelligent agent* drives each technician in the scenario by interpreting the scripted actions or orders.

Simulated procedures interact with each other across domain boundaries. For example, an agent performing an action, such as opening a valve, will generate an animated motion in the geometric domain. The result of its actions also affects the simulated systems (physical or systems domain). This in turn may change the appearance of an object (position of a gauge). Finally, the change, needle motion, can capture the attention of the agent. Having perceived the system's new state, the technician might decide to close the valve. The simulation halts if an action fails or a hazard occurs. The program should then switch to a debugging mode.

The simulation can be broken down into three domains: geometric, physics-based, and agent. These different simulation domains run in parallel and share the state variables common to their models.

**3.4.2.1. Geometric Domain.** Geometric simulation generates the 3D graphics fed in the user interface. It is produced by a *scene management system* which stores the scene's geometric description in a *scene graph*. This system also is used to detect collisions between simulated solids.

**3.4.2.2. Physics-based Domain.** A physics-based behavior simulation allows the production of a realistic response from the maintained system to the actions of the technician. This includes simulating physical processes that are reported as hazards (leaks, corrosion, combustion, electrical hydraulic or mechanical failure).

There are two types of physics-based simulations. The first deals with the behavior of physical objects due to their geometry. It prevents objects from interpenetrating with realistic collision reactions and contact forces. The second type is non-geometric and models systems as interconnected functional modules that exchange signals.

Most systems are modeled with non-geometric or lumped parameter models. They are assembled by interconnecting functional modules, as in block diagrams. The modules exchange signals representing flows of matter or energy. This paradigm applies to most

engineering domains (electric, hydraulics, control, mechanics, etc.). Traditionally, simulators use quantitative models, but we recommend semi-qualitative modeling. Semi-qualitative modeling allows physical processes to be simulated independently from the components in which they take place. This essential feature allows specific kinds of processes to be identified as hazards and the simulation to be halted when one of them occurs. For example, one can use a generic model of a fluid flow process and categorize it as a leak if it goes from the maintained system into the environment. In addition, the qualitative part of the simulation can be used as sensory input to the agents in the environment. Finally, a semi-qualitative simulation engine can “explain” the behaviors it generates. This self-explanatory feature is a core element of the system’s debugging functionality.

3.4.2.3. Agent Domain. The main product of TO authoring is a sequence of orders whose strict interpretation by technicians guarantees safe and successful maintenance procedures. The level of detail conveyed in the maintenance procedures must correspond to the skill level of the person performing the task. This can impact the level of detail the author must convey when writing the specific steps for a maintenance procedure. A desktop validation application would require an agent model with similar skills and expertise to interpret and perform a maintenance task in a realistic manner. In particular, interpreting means “understanding” an instruction and inferring the corresponding elementary action sequence. For example, when a technician is instructed to unscrew a bolt, the elementary task of grasping the right tool is not explicitly described. Furthermore, some of these tasks may be optional. In our example, this is the case if the technician already holds the right tool.

We propose to model the agent’s experience and skills with the Parameterized Action Representation (PAR) [10]. PAR would also be used as an internal representation or scripting language for action sequences that are explicitly described in a TO. In other words, in each task scenario, a PAR script would represent the orders stated in the corresponding TO.

A PAR action contains input and output conditions. The action is executed if the input conditions are satisfied. The output conditions are asserted upon completion. These conditions apply to facts about the state of the world known by the agent at the time of execution. For example, an agent operating a valve will be interested in its state (open or closed). These facts are acquired via sensory input simulated as sensory actions, which can

be limited by the situation or capability of the agent. For example, an agent cannot read a gauge if it is out of sight. Alternatively, if the author does not care for sensor modeling, the agent can be omniscient and extract facts from the whole simulation environment at any time.

Some of the input conditions can specify preparatory actions. These condition/action pairs are sub-goaling constructs where the action can be executed to achieve its associated condition if necessary. In our example, grasping a tool before using it is a preparatory action.

An action can either trigger PAR sub-actions or call out a primitive action. Primitive actions are skills such as locomotion, grasp and attention that are built into the agent. Until now, the PAR framework has been implemented with the EAI Jack Toolkit that provides the human model and the aforementioned skills.

PAR interpretation represents the cognitive process of the simulated technician. It drives the actions of the software agent controlling the geometric representation of the technician. A PAR-based agent is *reactive*. This means that its choice of preparatory actions will be based on the state of the world at the time of execution and not on planned or past actions. In other words, PAR actions are pre-set (static) hierarchical plans with optional parts. The agent is responsible for completing an action or reporting a failure, as well as its immediate cause.

While statically-defined actions might be sufficient for most tasks, dynamic action plans will be necessary for complex ones. For example, disassembly requires specific planning algorithms to compute a valid extraction sequence and path for each part of an assembly [12]. One possible solution is to use dedicated *skill modules* to generate PAR actions on the fly. In our disassembly example, a disassembly action would call upon a disassembly-planning module to generate a whole PAR hierarchic plan to perform the task. The action would be, in essence, refined or expanded dynamically. The plan returned by the module would be interpreted as a regular static PAR action.

### 3.4.3 Human Factor Dependent Validation

HFDV is a series of system dependent validations, each using a different human form model. Each simulation takes into account the specific biometrics of the human model. Most

of the hazards (collision with a moving part) or failures (failure to reach or see) detected at this stage will be caused by geometric factors. Other model-dependent factors such as endurance and strength might reveal that a procedure is too demanding for certain segments of the technician population.

The human models are supported in parametric form by dedicated software such as the EAI Jack Toolkit. The software provides physical skills and capabilities to our virtual technicians. A software agent completes the model with cognitive abilities. As with the biomechanical model, the parameters of the cognitive model could be manipulated to capture different expertise levels. The author could use them to assess whether a TO is explicit enough for an average technician.

HFDV can be automated as a batch process. If the system has a predefined database of representative models, they can be tested in sequence until a fault occurs or until the whole group is exhausted.

### **3.5 REAL-WORLD APPLICATIONS**

#### **3.5.1 Publishing Simulation Graphics**

Our framework focuses on supporting the TO validation process. It uses desktop virtual reality to depict the execution of maintenance procedures in a simulated environment. The produced 3D computer graphics appeal to the author's natural geometric reasoning to provide valuable insight into a maintenance task problem. Similarly, electronic maintenance manuals could be enhanced with these graphics to further assist technicians. Likewise, material from failed procedures could be used to communicate maintainability issues to engineering teams.

The published material could range from pictures and movies to animated or interactive 3D environments. It is still too early to say under which form data from the validation tool could be published or exported to another system; however, it will most likely be as an interactive multimedia document. Also, specific data combinations (2D, 3D, sound, and hypertext) could be generated. Such combinations will be platform independent as standards for pictures, movies, virtual worlds; human models will be integrated with semi-structured modeling languages such as the eXtensible Markup Language [13].

### 3.5.2 Model Importation, Building, and Maintenance

As stated earlier, we expect to use a PDM system to interface the authoring/simulation library (ASL) used to support a TO validation application, with the databases used to support TO authoring and engineering applications used by weapon system manufacturers.

The ASL is populated and updated by importing data (graphical and textual) from sources such as CAD, CAM, CAE, IETM, LSA, etc.). The data from each source must be converted to the ASL format using a specific data import process. Although CAD data is straightforward to convert by automated decimation (see Figure 6), CAE or TO conversion will tend to be more labor intensive. The main reason is that the formats of the

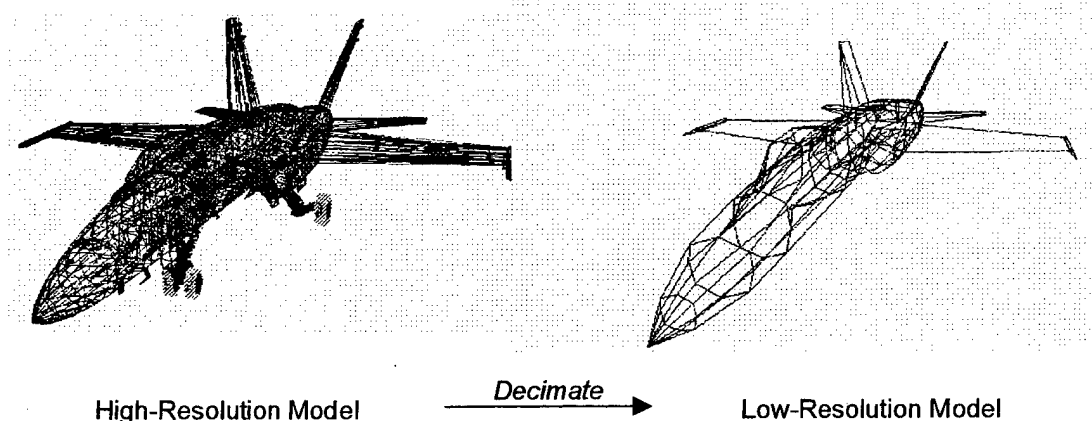


Figure 6. 3D Model Decimation.

source data and ASL are quite different. Natural language processing might assist in converting the narrative parts of LSA and IETM files into PAR.

Data importation will be a major process during the early phase of the validation tool. The virtual “backlot” will be updated with new models each time a procedure refers to a non-catalogued task scenario element. However, this up-front cost will be amortized once the ASL reaches its critical mass. Afterwards, occasional updates will keep the ASL in synch with the rest of the authoring system.

If TOs are edited within the validation tool, they will have to be exported back to the original authoring system (if any).

Manual data conversion may require more time and skills than a single author may have to offer. This implies that support modelers may be required to assist authors in setting up new simulations.

### 3.6 AUTHORIZING AND SIMULATION LIBRARY (ASL)

Authoring and simulation data support the author-test-validate cycle. The data is obtained through import from other engineering and logistics data sources , or input directly during the authoring activity. In either case, the data is stored as reusable models in the ASL.

We can classify the models by domain and geometric nature. We have three modeling domains: human, system, and task. Each can be divided into geometric and non-geometric components (see Table 1).

Model Domains	Components	
	Geometric	Non-Geometric
Human	Human model	Action representation, human model (biomechanics)
System	Individual system component shape, system assembly	Physics-based component model, hazard models
Task Scenario	Initial environment layout (technician and system position)	Initial environment state, subtask sequence

Table 1. Geometric and Non-Geometric Models.

#### 3.6.1 Task Scenario

A *task scenario* is the internal representation of a TO. It lists the initial state and composition of the environment. In particular, it indicates the number of required technicians, the configuration of the maintained system, and the required spare parts. This information describes the spatial position of each entity having a geometric appearance, as well as its internal state. It also includes the TOs themselves along with caution and warning messages. These subtask sequences are stored in textual and scripted (PAR) form.

Task scenarios are the master data of the validation tool. The rest of the ASL supports them. They are also the subjects of the validation process.

As a master document, a task scenario contains all the references to all the models it explicitly requires. Its metadata tracks their respective versions for revision control purposes. Extra meta-information such as the name of the author and a record of the validation process may be included.

### 3.6.2 Actionary

The Actionary is the library containing all the PAR actions of the validation tool. It represents the knowledge of the software agent driving the human model. This knowledge must be broad enough to enable a virtual technician and a human technician to interpret an order in a similar fashion. In other words, the Actionary provides a well-founded vocabulary of actions suitable to support direct translation of an order in textual form to a short script.

Each action is a procedure with parameters such as agent, objects, and manner. The agent designates the entity that performs the action. The objects are the entity on which the action is performed. The manner indicates how the action is performed.

An action has input and output conditions. The input conditions are subdivided into *applicability* and *preparatory* specifications. Applicability conditions define the properties that the agent of the object must have by design. For example, the `Open_Container` action will only accept containers that have a lid. Preparatory conditions specify the initial state in which the environment must be to perform the action. In the example, a container must be closed in order to be opened. A preparatory condition can be associated with an action whose execution will satisfy the corresponding condition. This action/condition pair is a way of formulating sub-goaling. In our container example, the `Open_Can` action could have the `Has_Can_Opener/Get_Can_Opener` condition/action pair. An agent would have to get hold of a can opener with the `Get_Can_Opener` action if it started executing the `Open_Can` without one. Output conditions define the effects of an action when its execution is completed.

The Actionary is an action taxonomy in which actions are grouped by categories and subcategories. For example, the `Open` action is refined depending on the type of its parameters. The most general `Open` category contains all the `Open` subcategories. We could have an `Open` subcategory for opening containers and another subcategory to open doors.

The latter category could be refined depending on the way a door opens: rotates or slides (see Figure 7).

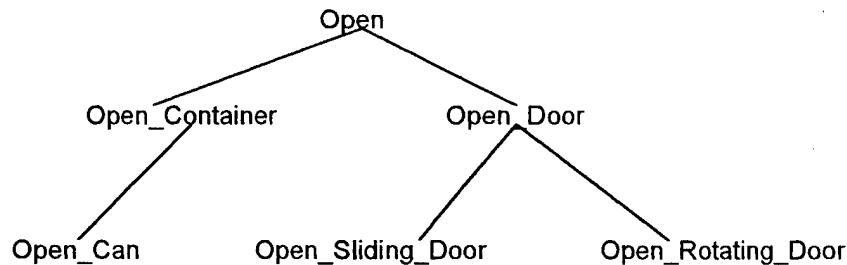


Figure 7. Open Action Taxonomy.

An action inherits the conditions from the action categories to which it belongs. If the Open action requires its subject to be Openable, then all the Open subcategories will perform that check.

When executed, a PAR action can either call a primitive action or execute a sequence of sub-actions (composite action). An action can execute many sequences in parallel. For example, the Remove\_Panel action might require the agent to hold the panel with one hand and open its latches with the other. The action's execution sequence would be of the form:

`Remove_Hatch(agent, hatch) = par_join(Hold(agent, hatch, left_hand),  
Unlatch_Hatch(agent,hatch, right_hand)).`

The `par_join` construct executes the hold and Unlatch\_Hatch subactions in parallel. It also ensures that the action completes when both subactions are completed.

Because of its procedural nature, its action composition constructs, and its taxonomic structure, PAR can be used as a scripting language for TOs, as defined by Ianni.

### 3.6.3 3D Models

3D models capture the shape and structure of the objects represented by the rendering system, including assemblies, tools, and human figures. Although this information is mainly used by the scene management system, it may be accessed by other components of the simulation such as an assembly-planning module.

### 3.6.4 Human Models

The parametric human model completes a human figure geometric representation with anthropometric data to assess the human factor impact of a given scenario on technicians varying in size, force, and gender. This assessment also includes reachability, effort, and attention. Each individual of the technician population is represented with a unique set of parameters to plug into the parametric model.

There are different implementations of human models, each with its own parameterization. Industry standards are being developed to improve interoperability, for example, by the SAE G-13 subcommittee [14].

### 3.6.5 Physical Behavior Models

We propose to represent the maintained systems as assemblies of elementary devices. Each device has a model stored in the ASL. Therefore, complex systems can be modeled as a network of interconnected devices. This modeling method is based on block diagrams, and is used in most engineering fields.

We do not need the same accuracy and level of detail as in engineering simulations. We only need physical models that are realistic enough to simulate hazards and action failures that are simple enough to run at interactive rates.

We diverge from traditional engineering practices by advising the use of semi-qualitative models instead of purely qualitative ones. Semi-qualitative modeling allows the numerical behavior necessary for an interactive simulation to be generated and an abstract qualitative representation suitable for the sensory or cognitive tasks of software agents (such as PAR execution, planning, or diagnosis) to be maintained. Aside from its dual representation, semi-qualitative modeling has the following features:

- **Physical Processes Can be Modeled:** Physical processes such as matter or energy flows can be modeled as independent entities. These models are automatically instantiated when the conditions supporting a flow are met somewhere in the simulated system. For example, a fluid flow can be instantiated in any pipe whose pressure gradient is non zero.

- **General Physics-based Models Can be Encoded:** Semi-qualitative models can directly encode the most fundamental behavior of most physical domains. This allows very general model libraries from first principles to be created.
- **Domain Models can be Integrated:** For example, a model library for fluids and a model library for thermodynamics can be combined within the same scenario. Dependencies between domain models can be encoded to provide automated model building.

The use of physical behavior models has two benefits. First, we can model hazards as processes. When such a process is instantiated, it signals itself as a hazard and the simulation stops. For example, any flow of toxic vapors escaping from a pipe can be flagged as hazardous. Second, representing processes separately from the components in which they take place provides a process-centered view of a simulation. This is particularly useful to understand what is happening. For example, it is easy to understand why the fluid level of a tank varies if one knows the active adjacent fluid flows. This type of analysis can be partially automated for debugging purposes.

## SECTION 4

### AMI CONCEPTUAL DEMONSTRATION

#### 4.1 INTRODUCTION

To illustrate the AMI framework, we present a conceptual demonstration of desktop TO validation. Our goal is to show how the framework discussed in Section 3 could be applied in the process of authoring a TO procedure, and identify the various technologies on which our proposed validation tool and framework rely. We will strive to show how the framework supports the incremental authoring and validation of a given TO procedure via a “generate and test” loop.

The sample maintenance procedure used for this conceptual demonstration involves the task of removing the nose landing gear (NLG) front uplock hook on an F/A-18 aircraft. To illustrate the use of the validation tool, we present an analyst’s iteration through three variant methods for performing the task. Each of these variants is referred to as a scenario. Each scenario will be demonstrated in sequence, and each scenario ends or terminates when a hazard occurs, or when all the prescribed actions have been performed. The first two scenarios illustrate the occurrence of hazards: a hydraulic fluid leak, and the dislocation of the front uplock hooks latch (the hooking element of the front uplock hook). The third scenario represents successful accomplishment of the task. We will call the technician performing the scenarios *Jack*, after the EAI Jack human model use to develop this demonstration. It should be noted that the same framework could be implemented with other types of human models

#### 4.2 CONCEPTUAL DEMONSTRATION DEVELOPMENT

##### 4.2.1 Initial State

The environment is comprised of Jack, an F/A-18 aircraft (see Figure 8), and two tools – a manual wrench and a pneumatic or electric wrench.

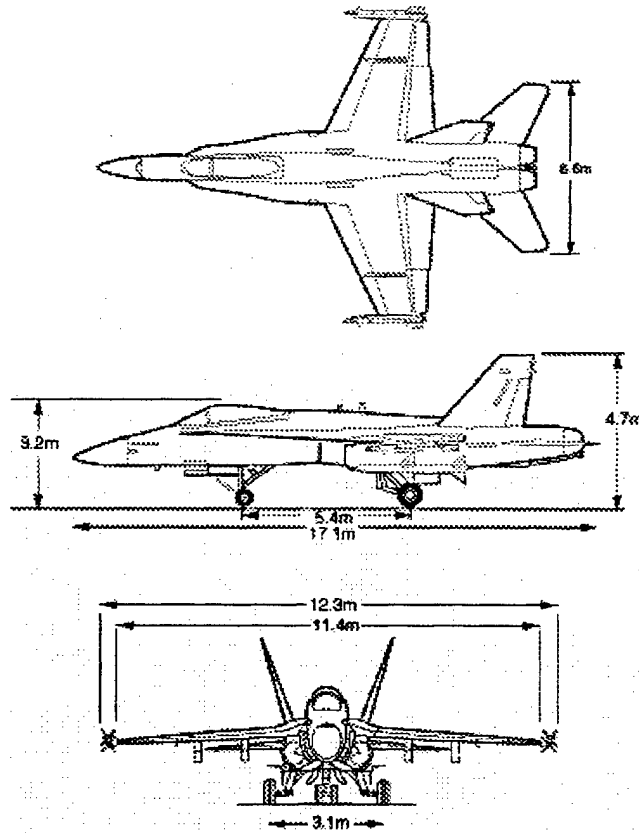


Figure 8. F/A-18 (©1999 The Boeing Company).

The airplane is jacked. The front-uplock hook is located in the front NLG's wheel well (see Figure 9), or more precisely, at the bottom of the bulkhead close to the front of the well.

There are three jack pads: one located behind the NLG and two under the wing, 2.87 m (113 in) off the centerline (see Figure 9). The front pad is 1.27 m (49.97 in) above the ground when the plane is resting on its landing gear (LG). The jack lifts it to 2.11 m (82.94 in). The nose landing gear front uplock hook is approximately at that height (a few centimeters higher).

Each scenario starts with Jack standing in the wheel well facing the FUH (forward). Jack stands on a step stool or a ladder to reach the front uplock hook. We assume that the hydraulic lines are pressurized.

CAUTION

TO PREVENT DAMAGE TO  
GROUND CABLE, DO NOT  
USE THIS LOCATION FOR  
GROUNDING WHEN LANDING  
GEAR IS CYCLED.

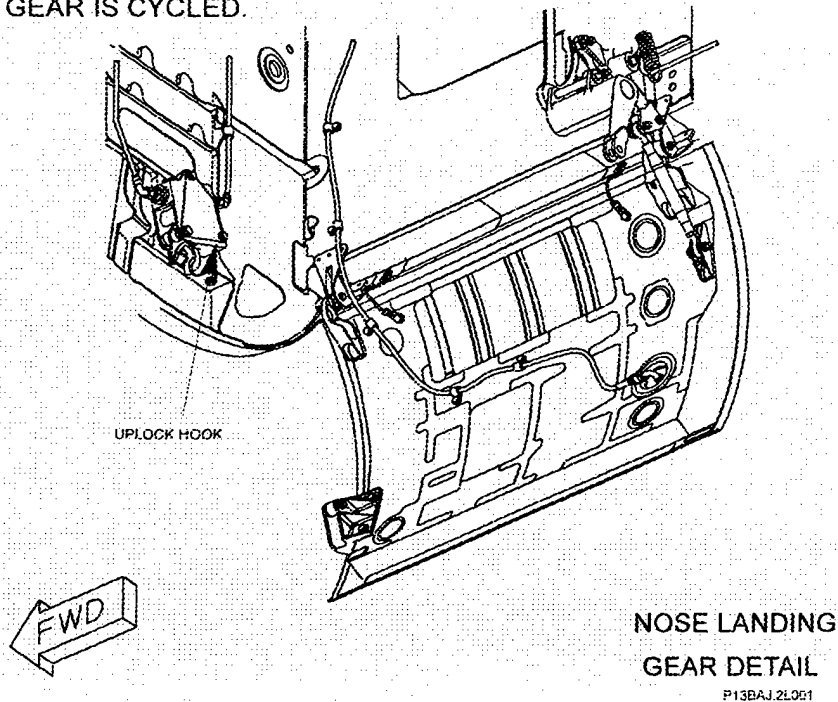


Figure 9. General View of the Work Area.

#### 4.2.2 Role of Technical Order

The goal of the TO procedure is to safely remove the front uplock hook assembly without inducing or experiencing any hazards. This removal task requires special care because the bolts that hold the front uplock hook against the bulkhead also hold spring-loaded mechanisms inside the front upload hook (see Figure 10).

As previously mentioned, the TO is a sequence of warnings and procedural steps that must be followed by the maintenance technician to prevent hazards during task execution. The direct orders must be carried out within the context of the warnings (standing orders). However, we must assume that the standing orders are consistent with the direct orders. The former only affects the latter in the way they are performed and do not require re-planning at that level.

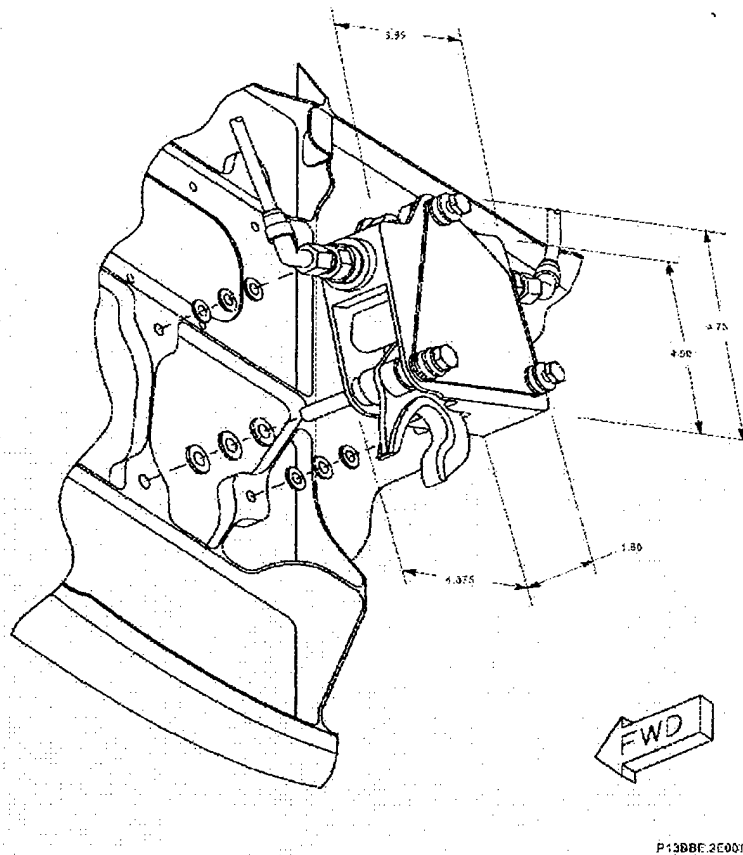


Figure 10. Front Uplock Hook Assembly.

We restricted the TO to the following sequence of direct orders:

1. Dump APU and emergency brake accumulators.
2. Disconnect the two fluid lines (elbows) from the front uplock hook.
3. Unbolt the front uplock hook from the bulkhead without removing the (three) bolts from the front uplock hook

Each of these direct orders translates to a sequence of PAR scripts. They are refined in lower level PAR actions during execution.

Two applicable warnings are associated with the direct orders:

1. *Titanium alloy lines will break if flexed or twisted too much during component removal or installation.*

2. *To prevent damage to NLG uplock hook spring loaded internal mechanism pivot points, do not remove bolts from NLG uplock hook.*

The first warning instructs the technician to disconnect the elbows with care. It specifies how the second step of the TO must be performed (with care). Also, the elbows do not provide structural support for the front uplock hook. Therefore, if direct orders 2 and 3 were reversed, there would be a risk of breaking the fluid lines (a structural hazard). The second warning requires that the bolts be unscrewed without removing them from the front uplock hook. This standing order is directly translated the third direct order.

The first standing order pertains to the neutralization of the fluid lines to be disconnected in the third order. It is an abstract version of the two separate steps. Each corresponds to complex procedures described in separate TOs. In the original TO they are performed at the beginning of the task rather than just before their effect is relevant.

#### 4.2.3 Script Authoring

An author could write and validate the simplified TO in three iterations. We assume that editing is performed within the validation tool. It could also be done in a separate application. In this case, the new version of the TO should be re-imported into the validation tool.

Setting Up the Environment. The author starts by creating a *task scenario* corresponding to the input conditions of the TO. This includes setting up the virtual environment. For the simplified TO, this includes a jacked F/A-18 model, a Jack figure representing the technician, a front uplock hook assembly, and two tools (a manual wrench and a pneumatic or electric wrench). The author must also adjust one or more cameras to observe the simulation. Some parts of the aircraft model may need to be removed or made transparent to facilitate observation.

The author must tell the system which system models to use during the simulation and what kind of behaviors are of interest. Part of this step may be automated. For example, when the author adds the geometry of the NLG front uplock hook in the scene, its corresponding physics-based model is automatically added to the simulated system.

Dependent models may be added as well. However, the author controls the scope of the simulation.

Editing the Subtask Sequence. The manner in which the author inputs the sequence of sub-tasks (or direct orders) in the task scenario depends on the tool's interface. The author uses a scripting language or free form text. In the first case, the text of the order could be input along with the script to simplify exporting the TO back to its original authoring environment. The text also serves as documentation for the script. In the second case, the program generates the script by natural language processing.

The overall subtask sequence is represented in a flowchart. To add a subtask, the author must insert it in the flowchart, and input the order's text and script. When the subtask's input is completed, the program performs a syntactic and semantic check of the script.

When writing the script, the author translates the order into a sequence of calls to procedural actions that are already defined in the system's database (or the Actionary if the script is PAR-based). The sequence of calls define a basic action vocabulary representative of the virtual technician skills.

Demonstrated Authoring Process. The author edits and tests the TO three times. The first two versions fail validation because of improper formulation. The third one passes the test.

The first version of the subtask sequence consists of two orders. The first instructs that the hydraulic lines attached to the be disconnected, and the second simply instructs the technician to remove the hook (see Figure 11).

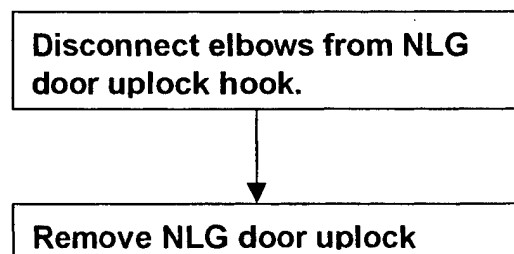


Figure 11. First Version of Subtask Sequence.

The first subtask is translated into PAR as:

```

Disconnect_Elbows(agent, hook) =
  seq(Disconnect(agent, hook.Left_elbow), Disconnect(agent,
    hook.right_elbow))

```

The action uses an agent and a hook as parameters to translate into the sequential disconnection of the front uplock hook elbows. The `Disconnect` action is a complex action assumed to be part of the Actionary. To execute it, the agent must know where to stand, what tool to use, and where to apply it. Since there are only two elbows, commanding their disconnection in the script is a simple task. Alternatively, the script could have been of the form:

```

Disconnect_Elbows(agent, hook) = execute(
  Generate_Dissassembly(agent, hook, { hook.Left_elbow, hook.Left_elbow},
    DISCONNECT))

```

`Generate_Dissassembly` is a direct call to a disassembly-planning module. The planner returns a plan in a PAR script form equivalent to the original disconnection sequence. The `execute` construct actually performs the plan once it is returned by the module.

The second order can be naïvely scripted as:

```

Remove_Hook(agent, hook) = seq(par_join(Get_Hold(agent, hook),
  seq(Remove(agent, hook.bolt1), Remove(agent, hook.bolt2),
    Remove(agent, hook.bolt3))),
  Put(agent, hook, Table))

```

The action has three nested constructs. The first sequence removes the hook and puts it away. The second removes the bolts and secures the hook. The third sequences the removal of each bolt. One could argue that this script should be purely sequential, starting with the `Get_Hold` and ending with the `Put`. Also, the author assumes that the `Get_Hold` will complete before the last `Remove`. Otherwise, the hook would fall. The agent must fetch the appropriate tool to disconnect an elbow. This information is in the script of the `Disconnect` action. Fetching a tool is optional. It will only be performed once for the first elbow. The `Remove` action also

requires a specific tool, an electric wrench. It has two steps: the technician first uses the wrench to unscrew the bolt, and then pulls it out from the assembly with the free hand.

The aircraft hydraulic systems are pressurized. Therefore, a leak is spawned by the simulator as soon as the first elbow is removed. This hazard stops the simulation.

The author corrects any mistakes by inserting a new order at the beginning of the subtask sequence (see Figure 12). This order dumps the appropriate aircraft accumulators. It is a complex standard procedure described by its own TO. As such, we assume it is already part of the Actionary. The author simply reuses it as the script associated to the first order.

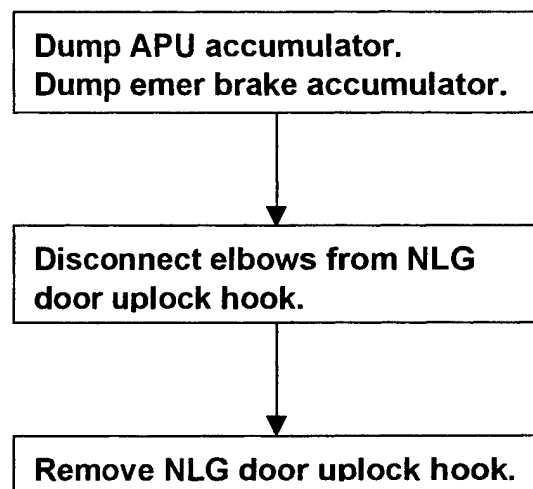


Figure 12. Second Version of Subtask Sequence.

The second simulation runs past the second order, which, this time, ends without hazards.

Because of the 's structure, removing its bolts completely from its assembly releases spring-loaded components. As the first removal action completes (*hook.bolt1*), the simulator detects the free spring-loaded parts and creates a hazard. This stops the simulation.

The author must modify the last order (see Figure 13) to prevent the hook from falling apart. The Remove action is replaced with an Unscrew, which will only detach the hook from the bulkhead and leave the bolts in the hook.

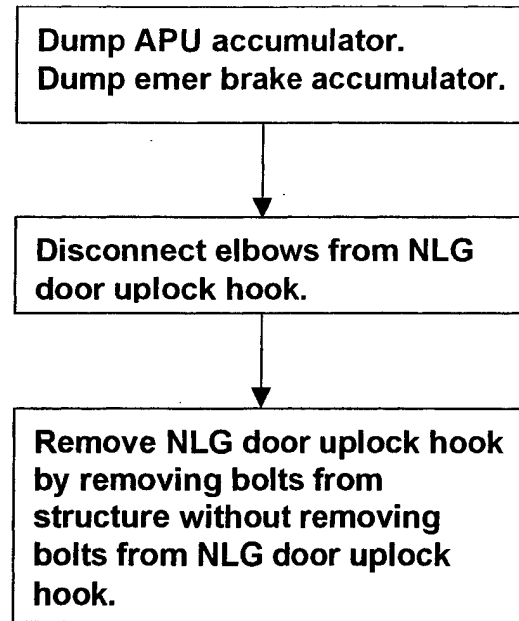


Figure 13. Third and Final Version of Subtask Sequence.

The script is of the form:

```

Remove_Hook(agent, hook)=seq(par_join(Get_Hold(agent, hook),
seq(Unscrew(agent, hook.bolt1), Unscrew(agent, hook.bolt2),
Unscrew(agent, hook.bolt3))), Put(agent, hook, Table))
  
```

This time the procedure executes successfully.

#### 4.2.4 Support Data

The edition-validation session requires a variety of models to set up the initial environment, edit the PAR scripts, and run the simulation. Geometric models are required for the airplane, the hook, the human model, the tools, and other maintenance structures such as a ladder or table. These models must carry the necessary sites, or landmarks, indicating locations to which the elements of the subtask sequence are relatively positioned. They also must identify grasp points on the tool or hook. If the agent uses a disassembly-planner, the geometric models must be appropriately annotated. The physics-based model of the relevant mechanical and hydraulic systems must be assembled from the semi-qualitative model library. Their initial state must also be specified. Finally, the Actionary must contain the

relevant high-level action vocabulary. In particular, the action corresponding to the accumulator dump must be available.

#### 4.3 CONCEPTUAL DEMONSTRATION – SAMPLE INTERFACES (VALIDATION MODE)

The conceptual validation tool used to model each of the three scenarios previously discussed consists of an interface mockup similar to what we envision an author would use to build and monitor the TO *validation process*. It is expected that the tool would provide additional user interfaces to support other core functions of a TO validation tool, such as setting up initial conditions for the simulation environment, data import, reporting, etc. The user interfaces for these type functions were not developed as part of this research effort.

The mock interface presented in Figure 14 contains the various text boxes used to input text and PAR scripts, as well as to display simulation status messages. It also features a window through which the user can see a 3D animation of the simulation environment, and change the virtual camera's position to inspect the execution of a task or the environment from different camera perspectives.

A task diagram box allows the user to insert new tasks as boxes and sequence them with each other by interconnecting. We anticipate that other connectors, such as decision nodes, could be used to create sequences with conditional steps. Each box contains a narrative text for the corresponding task.

The PAR box displays the PAR script corresponding to the active (in red) task box. The script can be edited manually or generated by natural language processing. It is envisioned that auxiliary windows could be brought up to browse an "Actionary" list for standard maintenance actions or tasks (e.g. jack aircraft).

The final box on the interface is a status box that is intended to inform the user of the progress of the simulation. This box could also display error conditions encountered during the simulation to the TO author.

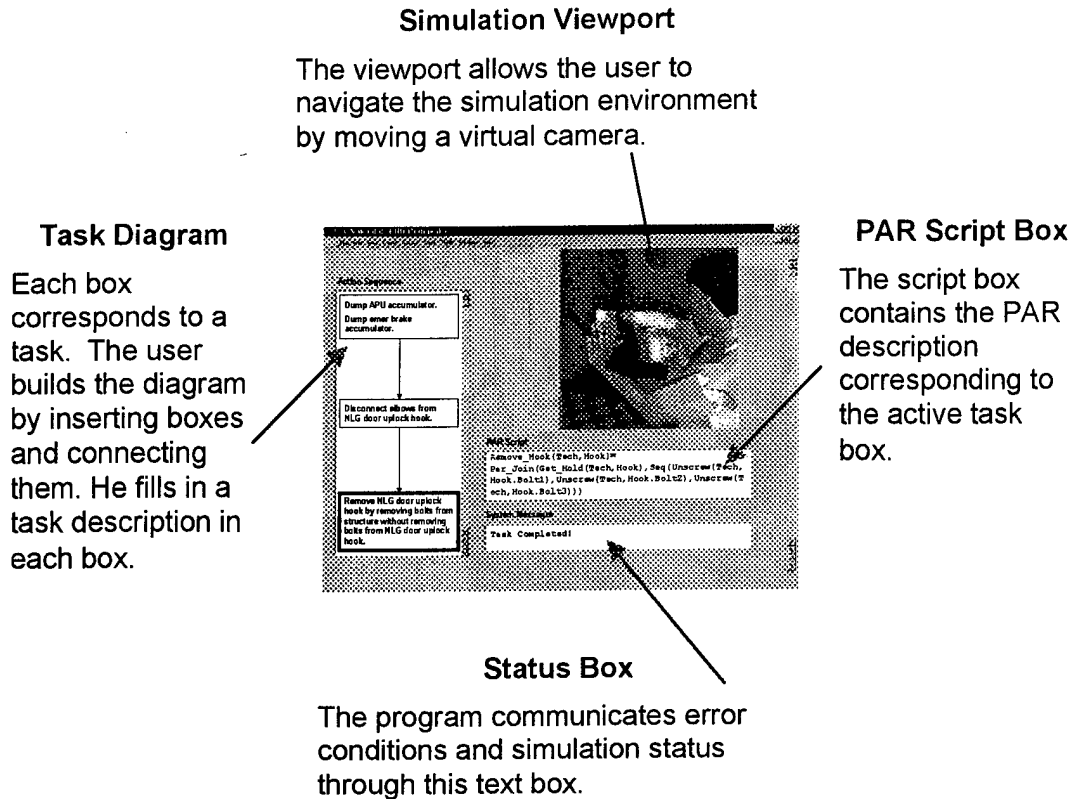
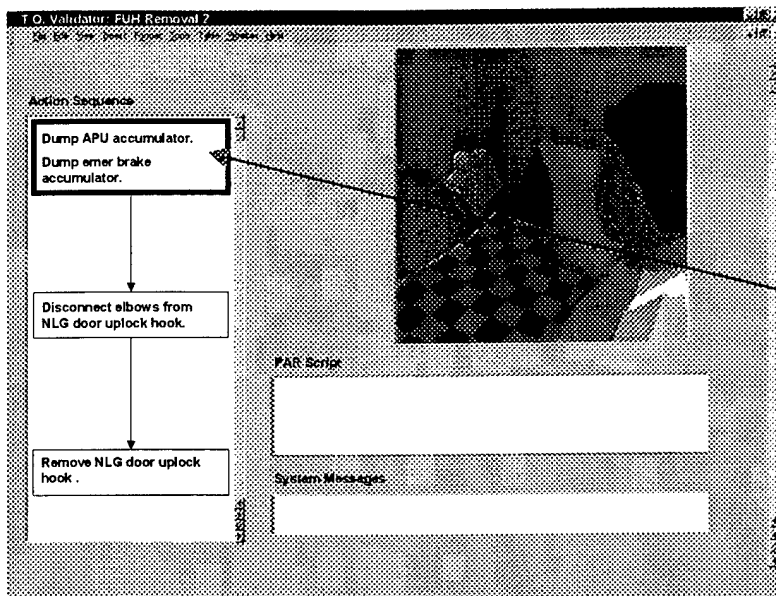


Figure 14. User Interface.

#### 4.3.1 EDIT / VALIDATE CYCLE

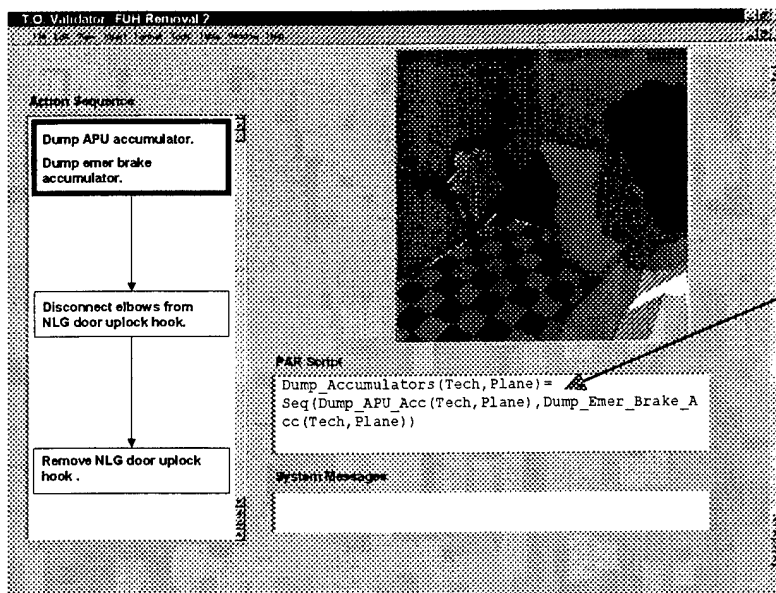
The edit/validate cycle is depicted in four steps in Figure 15 and Figure 16. The first two steps consist of data entry. The user inputs a task sequence and the corresponding PAR script. If the scripts are automatically generated, the user can always edit them.

The third step is the actual task simulation. The user can run it continuously, step-by-step, or suspend it to focus on a specific detail. The results of the validation are displayed in the status box (Step 4). If the simulation fails, the message will contain a description of the error. At that point, we could expect the interface to switch into a debug mode showing a trace of the simulation and other relevant information such as contentions or variables out of nominal range.



Step 1

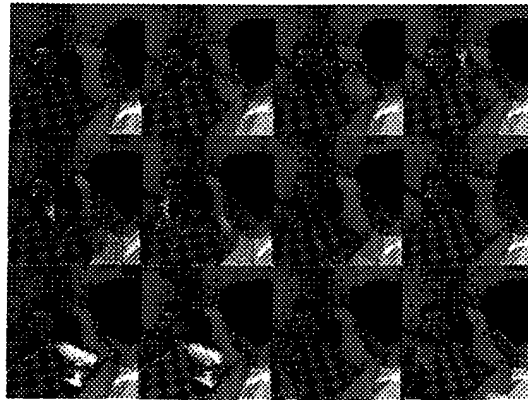
The author inputs the text of the first task of the TO.



Step 2

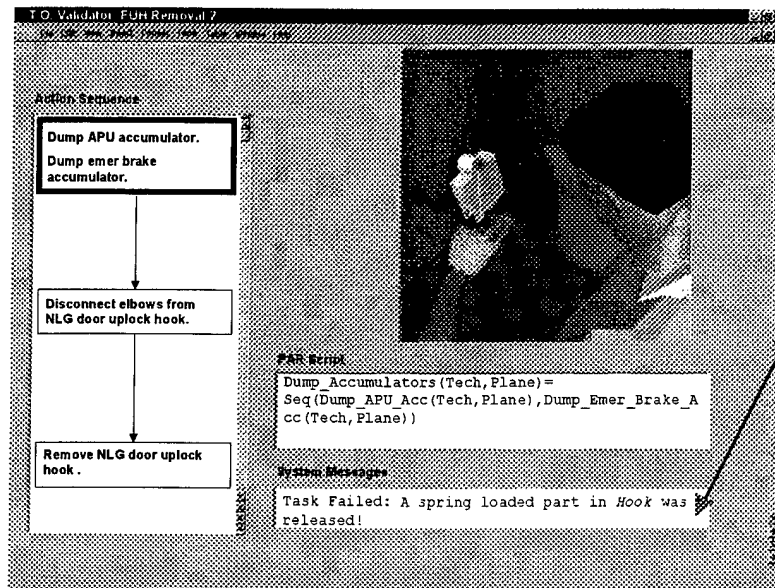
The first task description is translated into a PAR script. The translation is manual or via natural language processing.

Figure 15. Text and PAR script input.



### Step 3

The author runs the simulation.



### Step 4

The simulation halts and the system informs the author that a hazard has occurred.

Figure 16. Simulation and status feedback.

## 4.4 SUMMARY

Through a conceptual demonstration, we have shown how the AMI framework could be used to create and validate the steps in a typical removal procedure found in TOs. The author must first define the initial state of the procedure upon creating a task scenario. Once the simulation environment (layout, 3D, and simulation models, etc.) is configured, the author translates each subtask description sourced from LSA, existing IETM, or author's verbal rendering into a PAR script. Finally, through the Jack model, this PAR script sequence is executed by the virtual technician to support the validation process. If task procedure

actions fail, or hazards occur, the author can make the necessary modifications to the PAR scripts and recycle through the edit-validation process.

Our demonstration focuses on translating texts to scripts. The process could be manual or automatic (when the technology becomes available). In both cases, the Actionary must mirror in abstraction and semantics the actions commonly depicted in TOs.

The demonstration highlights the need for a dedicated skill module to handle assembly or disassembly actions. The rationale for this “outsourcing” relies on the radical difference between PAR, which is an *executive* framework, and an assembly-planner, which is *deliberative*.

The implementation of the framework hinges on the development of large-scale action and simulation model libraries. We identified the need for dedicated skill modules. In particular, an assembly planner should support the planning and execution of goal and constraint-directed orders prescribing assembly tasks. However, current assembly-planning technology handles only a small class of assembly tasks. The technological implications for developing an AMI framework is discussed in the following section.

## SECTION 5

### AMI RESEARCH ROADMAP

#### 5.1 CRITICAL TECHNOLOGIES

Two broad technology areas are critical to the design and development of a TO validation tool like the one described in this report. These areas encompass both software components and modeling framework. In general, the software components correspond to the simulators, software agent, rendering system, user interface, translators, and data manager. The modeling framework defines the standards that must be followed to engineer correct and reusable models for each domain.

The user interface and PDM components are mainstream technologies. Rendering and scene management systems are also reaching maturity. Although some are stand alone, others work as an integrated environment where various data and behavior sources interact uniformly. Human models are also available off the shelf. So far, we have been using the Jack Toolkit as an implementation candidate; however, other models exist. In any case, it would be prudent that the design of a TO validation tool be compatible with emerging industry standards for human models and 3D geometry [14].

##### 5.1.1 Human Models

Most human models today possess the basic capabilities needed to execute a task; they can reach and look, and walk and pose. Most models can change shape and size to reflect variations in human anthropometry; some even can adopt a specific person's body shape taken from laser or video scanning. The better models can be animated via procedural codes, motion capture, or interactive manipulation. The best models can be controlled through program interfaces and enjoy high-level behaviors such as attention, coordinated full body reach, balance, and collision detection. A desirable feature, not yet found in commercial human modeling systems, would be a direct linkage between strength, fatigue, comfort, collision avoidance, and task achievement. Inverse kinematic procedures can manage collision avoidance and task achievement, while dynamics simulation possesses all five features. True dynamics simulations are both expensive to simulate and difficult to control, and are not likely to be readily available outside the research or other specialized

communities (such as sport performance analysis or clinical biomechanic studies). However, most tasks in the aircraft maintenance domain are not characterized by fast, forceful movements – more likely by awkward postures, torque strength, repetitive actions, or hazardous substances. None of these situations requires true force-based dynamic simulations, so inverse kinematic procedures will usually suffice.

In order to function within the TO validation domain, a human model should be able to understand and execute tasks or procedure steps, preferably stated in a form convenient to the user. The software structure of the human model should facilitate access through a well-defined functional API and should permit the return of model state information useful for evaluation and validation. Ideally, such information will be used to guide or modify the simulation, thereby providing some task responsiveness in lieu of actual force-based (dynamic) simulation. For example, a reach task failure may trigger alternative access paths, collision detection may be replaced by collision avoidance, and an occluded line of sight may cause automatic re-posing of the human model. These are precisely the situations appropriate for task validation: feasibility is more important than optimality. No existing human model meets these requirements, but one with a good API and reporting facilities will be clearly superior. An instruction-level control and simulation system will fit comfortably on top of such an API. We next turn to examine a representation that will allow instructions to the human model (with a suitable API). With an instruction-level interface, the TO author should be able to launch human action validation studies from the TO text, see the results of the validation in computer graphics, and examine any resulting failure conditions.

### 5.1.2 PAR-Based Agent and Specific Skills

The AMI framework relies on the use of PAR scripts to model maintenance procedure subtasks and an Actionary to model the knowledge of the technician. We still have to prove that PAR is suitable for large-scale Actionaries. Furthermore, skill-specific technologies need to be integrated into the PAR system. In particular, (dis)assembly planning is a “must have” in the domain of maintenance simulation.

Because of its generality, PAR is not expressive enough to capture the complexity of a (dis)assembly operation in a flexible way. For example, one could script a whole disassembly sequence as in the editing process. However, writing such a script would be

labor intensive for large assemblies. In particular, the author must select an assembly sequence that guarantees that the assembly is stable at all times. This is known as the fixturing problem or finding areas to support or grasp an assembly to counteract its weight and insertion forces. In the uplock hook scenario described earlier, the author must instruct the technician to hold the hook with the left hand.

General assembly planning problems are currently too complex to be solved by computers. State-of-the-art assembly planning algorithms [15] will only handle simple problems. Simplifying assumptions such as one-step motion, one-step translation, and monotonic sequences means that the insertion path for each part is defined by a single rotation/translation or a single translation. Furthermore, the sequence cannot undo or temporarily reconfigure a subassembly to enable other insertions. Assembly planning research also addresses related problems such as fixturing [16] and use of assembly tools [17]. Few robust assembly planners exist. The most widely recognized as such is Archimedes 2 [12].

Using an assembly-planning module extends PAR functionality, but more importantly, it allows the author to let the virtual technician solve assembly problems as a human technician would and only detail critical tasks. The assembly skill allows textual and scripted orders to remain at the same level of abstraction. An assembly-planning module would allow an author to script assembly related orders the way they are naturally issued; i.e., with goals and constraints rather than with detailed assembly steps.

The uplock hook example shows that in spite of the disassembly skill of the technician special constraints have to be made to explicitly prevent hazards. In particular, the third TO simulation instructs Jack to unscrew the bolts without removing them from the hook. Assembly planners are also meant to handle such constraints [18].

### 5.1.3 Natural Language Technologies

In previous Air Force projects addressing TO generation, our research group investigated issues involving natural language understanding and generation. The theory behind this was based on the fact that TOs are written in natural language, not an artificial or algorithmic one. Therefore, the TO authoring process had something to do with the creation of such natural language text. As we studied the problem further and consulted TO authors,

the role of natural language shifted from generation more toward understanding. The main reason for the shift was that existing instruction sources -- either as TOs or as LSA records -- could be a resource in building the procedural step representation, or PARs. Once the maintenance task was described in PAR form, it could be edited, animated, and used for task validation. Moreover, the PAR form by design lends itself to natural language sentence generation should that be necessary or required.

Natural language technology can be used for TO validation under the following conditions:

- A natural language parser must understand the syntax of the sentences it is presented.
- A natural language parser must have a lexicon so that it can understand the words used in the instructions. The technology we use for this involves a particular kind of parser that uses tagged fields for each word (so-called lexical semantics) to properly interpret the input sentence.
- The parser must output its sentence analysis in a form that is digestible by other processes; in particular, we demand that the output be in an action representation form (PAR) suitable for subsequent control and animation of a human model.
- The natural language processing from sentence to PAR should occur fast enough to be transparent to the user of this technology.
- Natural language processing should eventually be satisfied by commercial off the shelf (COTS) components.

In the PAR implementation that we have developed, natural language technology is used to build the proposed framework to validate TOs. Our software module takes natural language instructions and generates one or more instantiated PARs. The basic linguistic representation of an action is a predicate-argument structure such as 'slide(John, box),' which indicates a particular action (the predicate 'slide') and its participants (the arguments 'John' and 'box'). We use the XTAG Synchronous Tree Adjoining Grammar System, which consists of a parser for extracting the predicate-argument structure of an input sentence, and a translator for generating an instruction script from this predicate-argument structure. The parser extracts these structures by first associating each word in an input sentence with one or

more elementary tree fragments, which are combined into a single derivation tree for the entire input sentence using the constrained operations of the XTAG Synchronous Tree Adjoining Grammar System formalism. These elementary tree fragments have argument positions for the subjects and objects of verbs, adjectives, and other predicates, which constrain the way the fragments can be combined, and which determine the predicate-argument structure of the input sentence. The translator then converts this predicate-argument structure into an instruction script, which in turn generates one or more instantiated PARs. With this architecture, a wide variety of inflections and grammatical transformations can be reduced to a much smaller set of predicates in the parser, and a variety of synonymous predicates can be further reduced to a still smaller set of PARs and scripting-language keywords in the translator. Although some parts of the translator may be domain-specific (some actions may depend on particular objects in a domain), the parser can easily be ported between domains, since its predicates are based on linguistic observations instead of on a particular programming language or virtual environment.

#### **5.1.4 Semi-Qualitative Simulation**

Semi-qualitative simulation has mostly been applied to build virtual laboratories. It has not yet been used for large-scale applications. We are currently developing a new semi-qualitative modeling language with standard object-oriented features that should help create and maintain large model libraries [19].

We are also addressing performance issues to reduce the lag between quantitative and semi-qualitative simulators. This difference is mainly due to the ability to change the structure of the simulated system during a simulation. This feature is required for specific applications such as maintenance simulation.

## **5.2 OUTLINE OF FUTURE TASK EFFORTS - FY2000**

### **5.2.1 Represent Procedure Steps with PAR**

Continue research to represent procedure steps with the Parameterized Action Representation (PAR) to describe how language inputs can effectively create PARs for downstream simulation and validation. The PAR allows a media-neutral form in which task instructions and their execution requirements may be stored for later retrieval, re-use, and simulation. By establishing a correspondence between the PAR parameters and the objects

and situations being examined, the PAR actions can animate a human form maintainer model such as Jack. The effort should focus on the feasibility of creating PAR instances from language and instruction analysis sources such as LSAR records, existing TOs, and the author's conception of the task.

Four tasks comprise this two-year effort: (a) create PARs for selected maintenance tasks, (b) investigate the requirements to correctly parse LSA records and produce or select PARs for them, (c) determine how to convert spatialized descriptions (in LSAR) to draw references in TOs, and (d) collect TO author monologues during changes and updates.

#### **5.2.2 Validate TOs through Automatic Generation of Virtual Motion Simulation**

Demonstrate TO validation by automatically generating virtual human motion simulations. This will consist of simulated assembly and disassembly tasks based on TO procedure steps, and should consider validation-critical issues such as confined reach task planning, spatial reasoning for part and assembly removal and replacement, and qualitative modeling of object function and behavior during maintenance tasks.

#### **5.2.3 Determine Knowledge Representation Requirements**

Determine the necessary knowledge representation requirements to actually deploy automated maintenance instructions. Beyond demonstration systems, there are real and significant issues related to obtaining and managing the large amounts of data, part information, CAD files, and the engineering schematics necessary for TO generation and validation tool. The requirements for a usable and scalable system need to be outlined.

This two-year program would include five tasks: (a) demonstrate that PARs for selected maintainer tasks can be simulated on a human model; (b) detect and report PAR simulation failures; (c) design software interfaces so that motion optimizations can be used if needed, but are not called if feasibility is more easily shown; (d) survey and establish priorities for human task functions that may need to be simulated; and (e) determine if the task analysis components of human models can be actively used during simulation to check task feasibility.

#### 5.2.4 Create PARs Through Human Performance Motion Capture and Semantic Analysis

Use human performance motion capture and the semantic analysis of those motions to construct PAR patterns (called UPARs: uninstantiated PARs) for typical maintenance activities. Human motion collected in a VR environment may be used to represent either coarse or fine motion strategies for part removal and replacement. Investigate how VR inputs and outputs impact the generation and use of PARs for maintenance actions. Since PAR is a media-neutral form used for action representations, the outputs that may be obtained from PARs should also be media-neutral. Investigate such media-neutral representations, for example, XML for multimedia markup and interpretation. Develop demonstrations that show how PARs can use a media-neutral output representation and how they may be variously interpreted in textual or graphic fashion.

### 5.3 BRIEF OUTLINE OF FUTURE TASK EFFORTS - FY2001

Extend the task validation via human form and system simulation. Candidate extension capabilities could be enhancements to the geometry/function reasoning system, improved performance in complex geometric situations, visualization of human interaction (contacts, pressure) with objects, and automatic annotation of maintenance-significant part features. The system should also provide reports on the cause of any validation failures. Such information would be used to inform the TO author of possible flaws in the task procedures. In the future, such information may be provided to automatic procedure planners who may attempt to reformulate the procedure steps or recommend other geometric alternatives to the concurrent design team.

Investigate the use of natural language as a direct means of modifying existing task PARs. This will be done initially using a fixed-initiative mode of interaction with the computer initiating the dialogue. The investigation should be expanded in the future to allow for more natural interaction, with more user initiation, and greater range of input modalities such as gesture.

Create context filters for multimedia presentations of TOs. Context is used to establish what information is presented and in what form. Different situations may require the same information be filtered and output differently. Examine the feasibility of presenting

TOs in forms useful to authors, maintainers, instructors, and trainees, including thumbnail stills, animations, and speech. Determine the role and usefulness of XML or other alternatives for these functions.

Demonstrate the prototype system on a typical TO generation, validation, and presentation task. Report on the process and recommend areas for further study as well as those ready for more systematic development.

#### **5.4 DEVELOPMENT PROGRAM**

In addition to integrating the results of any previous programs into electronic TO authoring systems, the following issues must be addressed, resolved, and implemented in any future development program:

- The PDM requirements must be defined, preferably with standardized terms and data requirements for maintenance features, object function, contents, etc. It may be best to select a target CAD system and its associated PDM and define the needed framework.
- Access to engineering and simplified CAD data on assembly shape, structure, and part function is needed to assess maintainer hazards, (dis)assembly orders, and equipment limitations. This data may be available through the PDM, but it may be scattered across enterprise databases and non-integrated software systems.
- A human modeling system interface should be based on a human modeling standard, or at least on a standardized API.
- A few robust extensions to human form animation systems need to be developed, especially collision avoidance reach planning and action failure reporting via the API.
- PAR and natural language parser software must be migrated into the TO authoring environment.
- Visual and textual interfaces must be implemented to launch validations and interpret their results within the authoring workstation.
- TO prototypes must be evaluated and iterated with real authors performing actual authoring and update tasks.

Since the issues outlined above are part of a large-scale software effort, a competent software integrator should bear prime responsibility. Software components from the proposed FY2000 efforts need to be incorporated and possibly extended. While COTS components such as human models may be available for certain aspects of the development effort, a CAD visualization tool, a language parser, and the baseline electronic TO authoring system, ongoing dialogues between contractors will clearly lead to increased likelihood of successful integration and product performance.

## REFERENCES

1. Przemieniecki, J. S., *Acquisition of Defense Systems*, Washington, DC: American Institute of Aeronautics and Astronautics, Inc. (1993)
2. Sanchez, E. and Boyle, E. Automated Support for Maintenance Technical Manuals, Technical Report AL/HR-TP-1997-0051, Human Resources Directorate, Logistics Research Division, Wright-Patterson AFB, OH (1997).
3. CIMdata, Product Data Management: The definition. An Introduction to Concepts, Benefits, and Technology. <http://www.cimdata.com/USTECH.pdf> (December 1998).
4. Object Modeling Group, PDM Enablers Joint Proposal to the OMB in Response to OMB Manufacturing Domain Task Force RFP1, <http://www.omg.org> (February 1998).
5. Chang, K-H, Silva, J., and Bryant, I. Concurrent Design and Manufacturing for Mechanical Systems, in Proceedings of ASME Design Engineering Technical Conferences, Las Vegas NV (September 1999).
6. Dai, F., Hopgood, F.R., and Hosaka, M. *Virtual Reality for Industrial Applications*. Berlin: Springer-Verlag (1998).
7. Dhillon, B.S. *Advanced Design Concepts for Engineers*. Lancaster PA: Technomics Publishing Co. (April 1998).
8. Phillips Mahoney, D. All Eyes on CAD, Computer Graphics World, PennWell Publishing, [http://pennwell.shore.net/cgw/coverstory/1999/05\\_story.html](http://pennwell.shore.net/cgw/coverstory/1999/05_story.html) (May 1999).
9. Iwasaki, Y., Farquhar, A., Fikes, R., and Rice, J. A Web-Based Compositional Modeling System for Sharing of Physical Knowledge. In Proceedings of the 15<sup>th</sup> International Joint Conference on Artificial Intelligence (IJCAI-97), pps. 23-29, San Francisco, CA, Morgan Kaufman Publishers (August 1997).
10. Badler, N., Palmer, M., and Bindiganavale, R. Agents Animation Control for Real-Time Virtual Humans. *Communications of the ACM* 42(1), 65-74 (1999).

11. Ianni, J. A Specification for Human Action Representation. In Proceedings of Digital Human Modeling for Design and Engineering. The Hague, The Netherlands (May 1999).
12. Kaufman, S., Wilson, R., Jones, R., Carlton, T., and Ames, A. The Archimedes 2 Mechanical Assembly Planning System. IEEE International Conference on Robotics and Automation, pp. 3361-3368 (1996).
13. EXtensible Markup Language.  
[http://www.xml.org/xmlorg\\_resources/whitepapers.shtml](http://www.xml.org/xmlorg_resources/whitepapers.shtml).
14. Society of Automotive Engineers G-13 Human Modeling Technology Subcommittee. Web address <http://www.sae.org/technicalcommittees/g13.htm>.
15. Halperin, D., Latombe, J.C., and Wilson, R.A. A General Framework for Assembly Planning: The Motion Space Approach. To appear in *Algorithmica*, Special Issue on Robot Algorithms.
16. Romney, B. Atlas: An Automatic Assembly Sequencing and Fixturing System. To appear in *Geometric Modeling: Theory and Practice* (Springer-Verlag).
17. Wilson, R. Geometric Reasoning About Assembly Tools. Technical Report SAND95-2423, Sandia National Laboratories (1996).
18. Rondall, J. and Willson, R. A Survey of Constraints in Automated Assembly Planning. In Proceedings of the 1996 IEEE Conference on Robotics and Automation, pp. 1525-1532.
19. Erignac, C. Semi-Qualitative Simulation in Virtual Environments. In Proceedings of the Thirteenth International Workshop on Qualitative Reasoning, Loch Awe, Scotland (June 1999).

## APPENDIX

### IETM AUTHORIZING REQUIREMENTS

#### 1. IETM AUTHORIZING REQUIREMENTS (see Figure A-1)

Current weapon systems being fielded for operation are supported by Interactive Electronic Technical Manuals (IETMs). The information presently contained in paper documents is displayed electronically to technicians on Portable Maintenance Aids (PMAs). The combination of the data to support the weapon system and the presentation system running on the PMA enables the technician to interact with the computer. The system presents only the data required to complete a task and displays only that data applicable to a given weapon system. In order to deliver this type of information, data must be authored in a different manner than that used to produce paper documents.

IETM authoring is driven by totally different considerations than the authoring of paper manuals. No longer is page appearance primary. What drives the software and hardware is the content of the data. In IETM authoring, each piece of data is inserted into a "slot" in a database. A maintenance procedure is not stored as a flat file; rather, a procedure's elements are arranged by database schema, and at display time the pieces needed are pulled from the database and assembled in the proper order. The IETM Authoring System database is compatible with MIL-D-87269 (Data Base, Revisable: Interactive Electronic Technical Manuals, For the Support of). The content of the database is accessible by selecting the desired system, subsystem, and sub-subsystem, and provides the following types of data:

- Descriptive
- Procedural (Tasks)
- Fault
- Part

Additionally, when data is used in more than one place, it is created and stored once (the second, third, and fourth occurrences simply point to the first one), thus permitting common data to be reused. For instance, warnings, cautions, and notes are used throughout procedural data. Many of these are repeated many times.

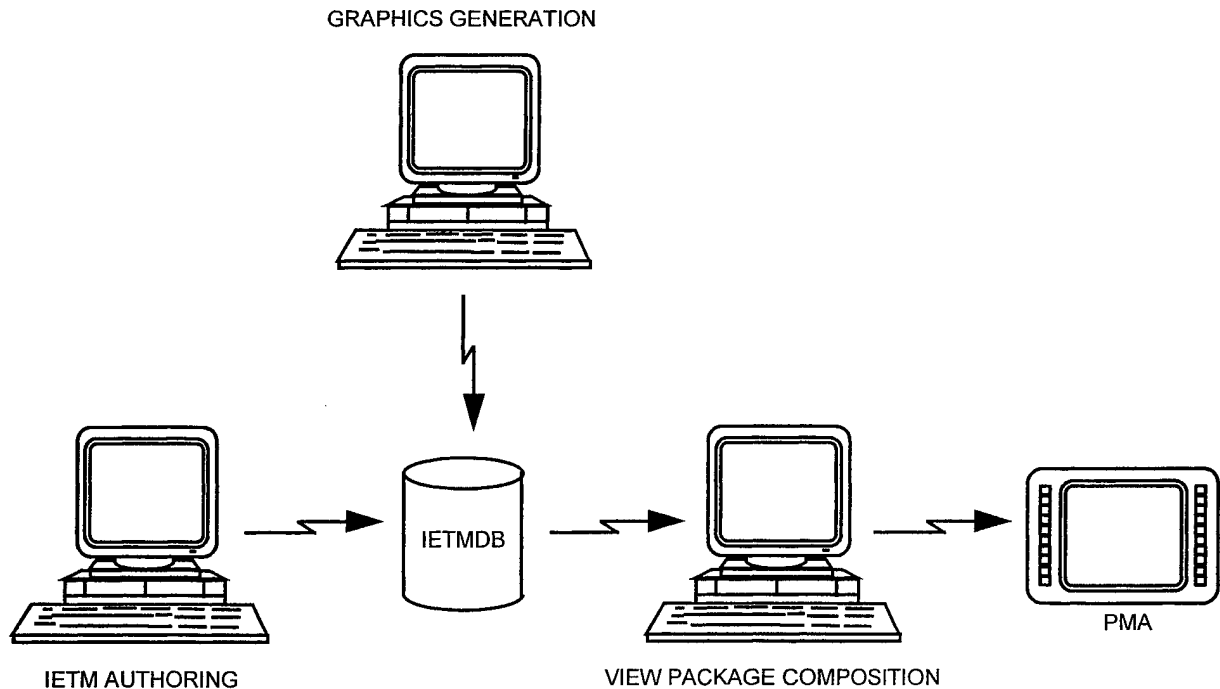


Figure A-1. Interactive Electronic Technical Manual Authoring.

Another consideration is the electronic linking of data. If one procedure references another task, it is not necessary to name the referenced procedure by manual name and number. The system automatically links and permits the user (at display time) to select an option to initiate the link.

The system also allows cross-links to be built between text and graphics. The authoring system displays graphics and permits the writer to insert a pointer or a callout to a specific spot (coordinate) on a graphic. The graphics developed by this effort comply with MIL-D-28003, the CALS Computer Graphics Metafile (CGM) standard.

## 2. RESEARCH

An author spends much time doing research. Research cannot be fit into a practical time slot. It is a continuous process during the contract period for a technical manual. Through research, the author collects and evaluates information to gain thorough knowledge of the product, including its operating principles, use, materials, and maintenance.

The amount of data available depends on the development stage of the equipment. During the early stages of development, the author may be limited to information sources such as the following:

- Detail specifications
- Design data books
- Engineering design sketches
- Models
- Mockups
- Personal working relation with design engineer.

As development progresses through production, delivery, and use of the equipment, research for the manual expands into areas such as the following:

- Engineering drawings
- Engineering orders
- Engineering change proposals
- Time compliance technical orders
- Publication change requests
- Field service reports

### **3. DATA SOURCES (See Figure A-2)**

The data sources listed in Table A-1 are used in the development of IETM data. The data are broken down by the major data types provided in an IETM (descriptive, procedural, fault, part). Procedural (tasks) data contains all the information required to do maintenance on the aircraft. Each task provides complete, step-by-step, start-to-finish maintenance instructions. A list of typical tasks is provided below:

- |                        |                   |
|------------------------|-------------------|
| • Removal              | • Adjustment      |
| • Installation         | • Calibration     |
| • Inspection           | • Ground Handling |
| • Cleaning             | • Servicing       |
| • Operational Checkout |                   |

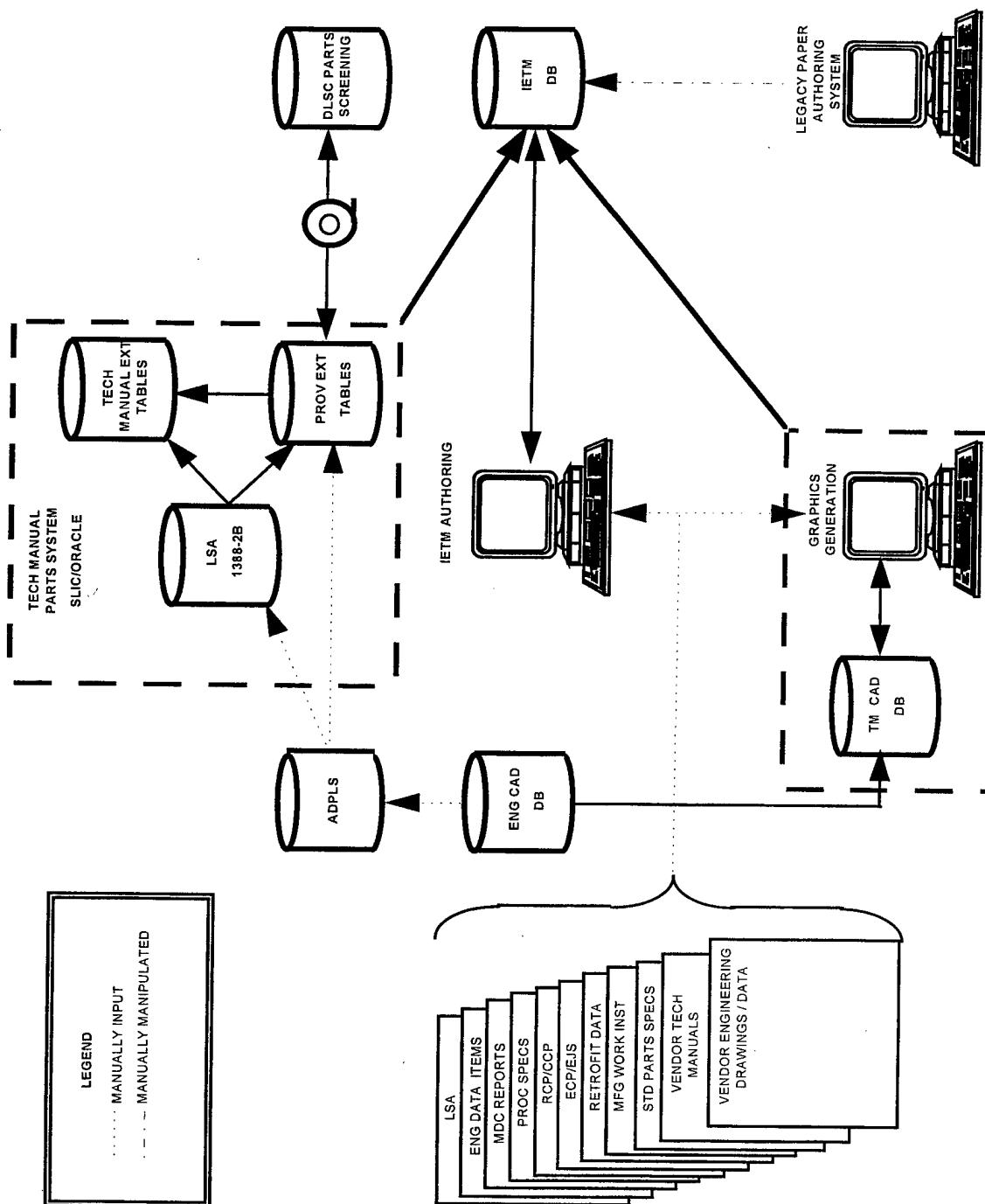


Figure A-2. As-Is Data Sources.

Table A-1. Data Sources Used in IETM Authoring

<b>LSA</b> LSAR-024 Report - Maintenance Plan Part III: Identifies support equipment requirements by task LSAR 019 Report – Task Analysis Summary: Provides sequential task narrative
<b>Engineering Data</b> (random reports/presentations) Data Item E-12.13E - Human Engineering Design Approach Document - Maintainer Data Item E-35.07E - Booklet of Maintenance and Operating Instructions
<b>Engineering/Vendor Drawings</b> (including drawing notes)
<b>Retrofit Data</b> - modifies aircraft configuration in the field
<b>Requirements Change Proposal (RCP) / Configuration Change Proposal (CCP)</b> - used by vendor to submit recommended component changes to contractor
<b>Engineering Change Proposal (ECP) / Engineering Job Sheet (EJS)</b> - used by engineering to submit recommended changes to aircraft to the customer
<b>Factory Visits/Actual Hands On</b>
<b>Provisioning Data</b> - Part ordering data / SM&R codes
<b>Process Specifications</b> - Provides process instructions for tubing inst., elec. bonding, and grounding, etc.
<b>Standard Parts Specifications</b>
<b>Engineering Coordination and Review of Data</b>
<b>Validation/Verification</b> - actual performance of the procedures (validation is performed by contractor / verification is performed by customer)
<b>Manufacturing Work Instructions (installation)/Visual Aids</b> - Provides instruction for installing parts in factory
<b>Engineering Reports</b> 94B0128A - Maintainability Equipment Access Matrix: Provides location of components (door/access information).

## 6. IETM AUTHORING PROCESS WITH RESPECT TO DATA SOURCES

STEPS	DATA SOURCES USED
<b>Identify System Components</b>	LSA /System Functional Schematics
<b>Determine Level of Maintenance Requirements</b>	<ul style="list-style-type: none"> <li>• LSA</li> <li>• Provisioning data</li> </ul>
<b>Select LRU/WRA</b>	
<b>Research Task Requirements</b>	
1. Should hydraulic and electrical power be off during maintenance?	<ul style="list-style-type: none"> <li>• LSA/system functional schematics</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>
2. With external power off, is line still pressurized?	<ul style="list-style-type: none"> <li>• LSA/system functional schematics</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>
3. <i>How is line pressure relieved?</i>	<ul style="list-style-type: none"> <li>• LSA/system functional schematics</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>
4. When a fluid line is to be disconnected, will fluid continue to drain?	<ul style="list-style-type: none"> <li>• LSA/system functional schematics</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>
5. Are safety devices required to be installed during maintenance?	<ul style="list-style-type: none"> <li>• LSA/system functional schematics</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>

STEPS	DATA SOURCES USED
<p>6. If maintenance is to be performed on an electrical or electromechanical component which is hard wired –</p> <p>a. Should wires be removed from an existing splice or cut as close to component being replaced as possible?</p> <p>b. Is hookup schematic required when splicing or reconnecting wires?</p> <p>c. Is wire bundle positioning and clamping critical?</p>	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Process specifications</li> <li>• LSA /system functional schematics</li> </ul>
<p>7. Should aircraft be on jacks during component maintenance?</p>	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• LSA /system functional schematics</li> </ul>
<p>8. If aircraft is on jacks with power applied, should circuit breakers be pulled or ground power switches off to de-energize other systems?</p>	<p>LSA/system functional schematics</p>
<p>9. Will other components have to be removed for access?</p>	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• LSA /system functional schematics</li> </ul>
<p>10. Are fasteners securing component all the same type, size and length?</p>	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• LSA /system functional schematics</li> <li>• Standard parts specifications</li> </ul>
<p>11. Are the component fasteners one-time-usage only?</p>	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Standard parts specifications</li> </ul>
<p>12. Are special torque instructions required?</p>	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> <li>• LSA /system functional schematics</li> </ul>

STEPS	DATA SOURCES USED
13. Are the fasteners safe-tied?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> <li>• LSA/system functional schematics</li> </ul>
14. Should an old sealant be removed before component removal?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Process specifications</li> <li>• LSA /system functional schematics</li> </ul>
15. Prior to removal, are special alignment marks required to eliminate unnecessary rigging?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> <li>• LSA/system functional schematics</li> </ul>
16. Is component removal procedure the same for access as for replacement?	Engineering/vendor drawings (including drawing notes)
17. Are special electrical bonding and sealing instructions required?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> <li>• LSA/system functional schematics</li> </ul>
18. Will sealant cure time affect assembly sequence?	<ul style="list-style-type: none"> <li>• Engineering/Vendor Drawings (including drawing notes)</li> <li>• Process Specifications</li> </ul>

STEPS	DATA SOURCES USED
19. Are warnings or cautions required?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• LSA/system functional schematics</li> </ul>
20. Are critical installation dimensions required?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• LSA/system functional schematics</li> </ul>
21. Are special parts assembly sequence required?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> <li>• LSA/system functional schematics</li> </ul>
22. What materials will be required to do procedure:  Tape                      Shims Hydraulic Fluid        Lockwire Cotter Pins              Grease Washers                  Fasteners	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> <li>• LSA/system functional schematics</li> </ul>
23. Which way should lubrication fittings and bolt heads be facing when installed?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>

STEPS	DATA SOURCES USED
24. Is the assembly being removed "procurable at o-level" or is it coded "assemble at o-level" which means that the parts which make up the assembly are procurable separately and assembly instructions will be required?	<ul style="list-style-type: none"> <li>• Provisioning data</li> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• LSA/system functional schematics</li> </ul>
25. Does the part have to be trimmed and drilled on installation?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• LSA/system functional schematics</li> </ul>
26. Does new replacement component come complete and ready to install, or is it necessary to remove parts from old component for installation on new component?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• LSA/system functional schematics</li> </ul>
27. Should parts be inspected (QA)?	
28. Is lubrication, servicing, air bleeding, or rigging required?	<ul style="list-style-type: none"> <li>• Engineering/vendor drawings (including drawing notes)</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> <li>• Process specifications</li> </ul>
29. What checkout is required after installation?	<ul style="list-style-type: none"> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>
30. Are test hookup and use instructions required?	<ul style="list-style-type: none"> <li>• LSA</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>
31. Is required GSE authorized and is it available?	<ul style="list-style-type: none"> <li>• LSA</li> <li>• Human Engineering Design Approach Document – Maintainer</li> <li>• Booklet of Maintenance and Operating Instructions</li> </ul>