A11103 034595

NIST
PUBLICATIONS

NISTIR 89-4117

NATIONAL
PDES
TESTBED

THE USE OF GMAP SOFTWARE AS A PDES ENVIRONMENT IN THE NATIONAL PDES TESTBED PROJECT

Kim L. Perlotto

**Pratt & Whitney,
United Technologies**

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Engineering Laboratory
Factory Automation Systems Division
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Raymond G. Kammer, Acting Director

NIST

QC
100
.U56
89-4117
1989

NISTIR 89-4117

NISTC
QC100
.U56
NO. 89-4117
1989
C.2

NATIONAL PDES TESTBED

THE USE OF GMAP SOFTWARE AS A PDES ENVIRONMENT IN THE NATIONAL PDES TESTBED PROJECT

Kim L. Perlotto

Pratt & Whitney,
United Technologies

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Engineering Laboratory
Center for Manufacturing Engineering
Factory Automation Systems Division
Gaithersburg, MD 20899

Sponsored by
U.S. Department of Energy
Office of Buildings and Community Systems
Washington, DC 20585

June 1989

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Raymond G. Kammer, Acting Director

THE USE OF GMAP SOFTWARE AS A PDES ENVIRONMENT IN THE NATIONAL PDES TESTBED PROJECT

PREPARED UNDER RESEARCH AGREEMENT BETWEEN
PRATT & WHITNEY AND NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
AS PART OF THE USAF GMAP CONTRACT

KIM L. PERLOTTO

JUNE 1989

ABSTRACT:
=========

This report is a basic guide to the use of the GMAP System Architecture as installed on the NIST AMRF VAX as part of the National PDES Testbed Project. An overview of the GMAP System Architecture is provided. The use of the GMAP software to create an implementation environment for the PDES First Working Draft (February 1989) is outlined. This PDES specification has been accepted by ISO as an international Draft Proposal. The software organization on the NIST AMRF VAX and the development of test and validation applications are described.

The GMAP System Architecture consists of software system components which meet the three basic requirements of an automated product data environment. The requirements are data definition, application support, and data exchange. The system components are defined and the role they play is described.

GMAP is the Geometric Modeling Applications Interface Program, a United States Air Force contract for which Pratt & Whitney, United Technologies, was the prime contractor. The installation of the GMAP software deliverables at the National Institute of Standards and Technology was possible under a technology transfer clause of the GMAP contract.

INTRODUCTION

As part of the technology transfer tasks under the United States Air Force Geometric Modeling Applications Interface Program (GMAP), the software for the GMAP System Architecture supporting a product data technology environment was installed at the National Institute of Standards and Technology (NIST) by Pratt & Whitney, United Technologies, in support of the National PDES Testbed Project. The Product Data Exchange Specification (PDES) [1] is a product data specification being coordinated by NIST, developed by industry, and desired by the Department of Defense Computer-Aided Acquisition and Logistics Support Program. The PDES specification has been accepted by the International Organization for Standardization (ISO) as an international Draft Proposal, and is currently in the review process. In the ISO arena, the specification is informally known as the Standard for the Exchange of Product Model Data (STEP). The capability to test, verify, and validate the PDES is a necessary one to make the specification a success.

The GMAP System Architecture installed on the NIST Automated Manufacturing Research Facility (AMRF) VAX computer is being made available to industry at large to test the current PDES specification. Testing PDES in the GMAP environment includes the creation of product model instances in the PDES schema, and the development and evaluation of application programs against those product models. The tools with which this can be accomplished are part of the GMAP System Architecture, and are accessable on the NIST AMRF VAX computer.

The GMAP System Architecture supports a PDES implementation which is based upon active file transfer. This means that product models are contained in memory, accessed and manipulated there, and mapped to a sequential exchange file on demand.

The GMAP System Architecture as installed at NIST, along with the provided PDES physical implementation files, SUPPORTS EVERY ENTITY IN THE ENTIRE PDES SPECIFICATION. At this point in time, it is believed to be the only facility available to general industry that can make this claim. As such, the interest in this environment should continue to be very high.

Furthermore, with minimal effort, the PDES specification upon which the system operates can be removed, and replaced with future versions of the PDES specification, utilizing the ability of the GMAP System Architecture to maintain schema independence.

PDES TESTING IN THE GMAP SYSTEM ARCHITECTURE

The testing of PDES using the GMAP System Architecture involves the use of the GMAP software to support the product data environment. The basic approach follows.

The focus of the system centers on an instance of a product model. A product model is a rendition or implementation of a specific product in terms of a given standard specification, in this case, the PDES specification.

In order to create a product model in the GMAP System Architecture, one should develop a product model specification that is a document describing each and every entity instance to be in the model, its attributes, and its relations to other entities. The entity pool that this specification is created from is the PDES specification itself, and attribute values are determined from the specific product design.

The product model specification can then be used to guide the modeler in the creation of the product model in the GMAP System Architecture. This operation is possible using components of the GMAP software system.

The validation of the product model created is then performed. The validation is from two points of view. The product model needs to be evaluated as to its faithfulness to the PDES specification in the formation of the entities, their attributes, and relations. The second point of view for the validation process determines if the product model accurately contains all the design attributes as specified by the product model specification.

Once a correct validated product model is available in the GMAP System Architecture, life cycle applications that access and manipulate the product data in the product model can be developed. These applications are supported by several GMAP system components.

The product data applications can then be executed, and their results evaluated. The success of the applications can be measured by the amount of data that they can be provided with from product models, and the activities they can perform based on that data being available in a single common format product model.

The detailed description of the indiividual GMAP System Components that support the product data environment and testing development processes outlined here follow. For those not concerned with the technical aspects presented, it is possible to skip to the section called "Developing PDES Test Applications" to read how the testing development process suggested here is accomplished within the GMAP System Architecture.

GMAP SYSTEM COMPONENTS

The GMAP System Architecture (Figure 1) consists of several software system components which together meet the requirements of an automated level 2 product data environment. There are three basic requirements that need to be met. They are data definition, application support, and data exchange. The system components are defined and the roles they play are described.

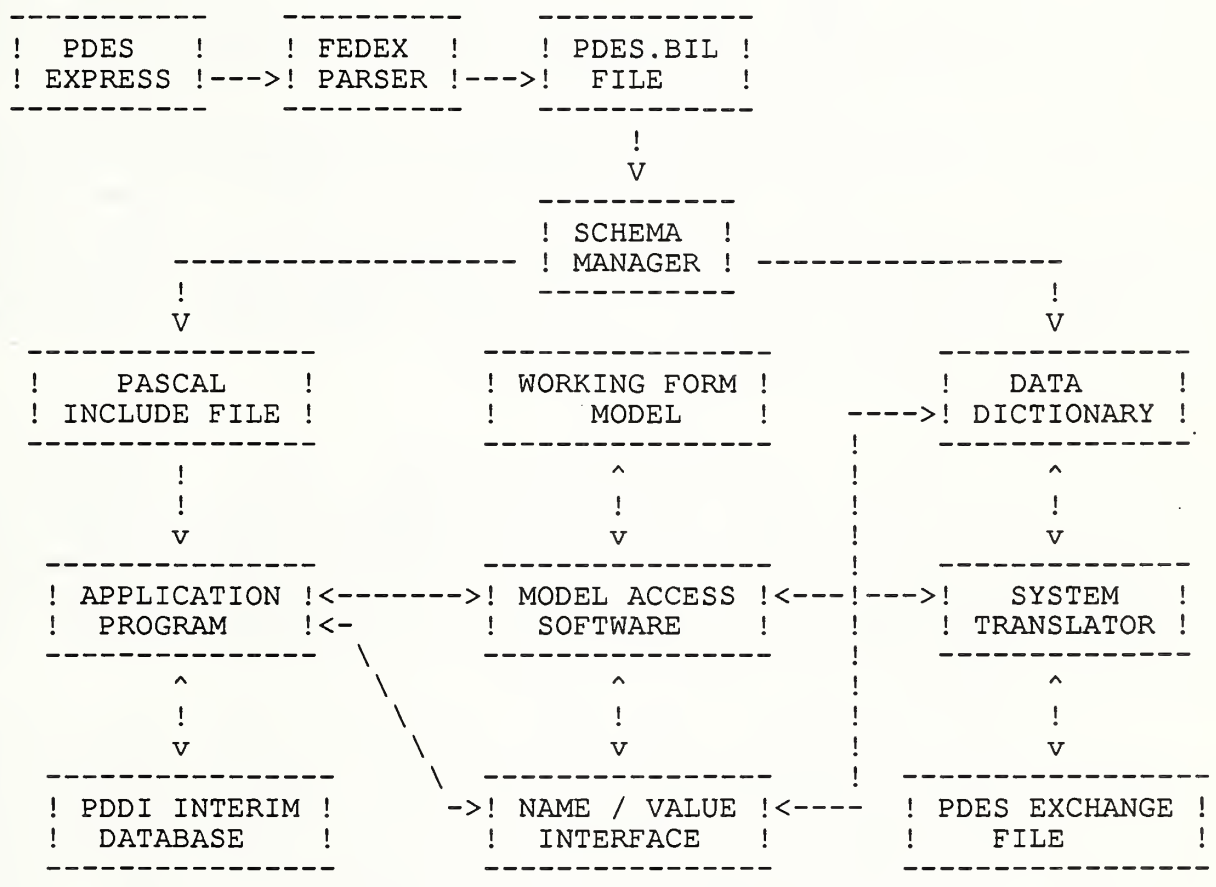

Figure 1.

A diagram of the components and interactions in the GMAP System Architecture at NIST

Working Form

The central focus of the system is an instance of the product model. This is supported by the Working Form. The Working Form is a memory resident network representation of a product model. This Working Form is machine dependent, meaning that Working Forms of a given model on different hardware platforms are not compatible. They do however, behave and appear identical to the software that interacts with them.

The Working Form of an entity consists of three basic constructs in the GMAP implementation. These are the Application Data Block (ADB), the Constituent List (CL), and the User List (UL). Each of these are separate but linked data structures, and the several Model Access Software routines available act on these entity structures as well as Application Lists that are working lists for application use.

The ADB is the base data structure for an entity. All attributes that are of primitive data type or aggregations of primitive data types are contained in the ADB. The system manages entities in the Working Form by a set of attributes included by the system in the ADB. These are present in every entity, and are named KIND, a number representing the entity type, LENGTH, the byte size of the ADB for a particular entity type, SYSUSE, a field used by the system for various flags, and IDENT, a field that contains the instance number for a particular instance of an entity type.

The Constituent List (CL) of an entity instance contains the attributes that are of the reference or pointer type. The relations between an entity and the entities that it depends on for its definition are located in the CL. The position on the list is important, and correlates the pointer with the appropriate attribute. The Data Dictionary and the Pascal Include File (see below) contain information to correlate symbolically the CL position with the attribute name. The CL must be constructed by applications that create entities in the appropriate manner.

The User List (UL) is a list of all entities in the Working Form that reference a particular entity for definition of one or more of its attributes. While it represents information used by the system to control the deletion of an entity, it also represents the backward pointers in the network model. In some instances, depending on the schema definition, it is necessary to investigate the UL instead of the CL to navigate from a given point in the network to points of interest. The UL is completely maintained by the system, however, and need not be constructed or otherwise manipulated by application programs.

The Model Access Software accesses entity instances in the Working Form by a key, or unique number that addresses the ADB, CL, and UL of the entity instance.

The Working Form, being the central focus of the GMAP System Architecture, participates in all three roles required of a product data environment: data definition, application support, and data exchange.

Schema Manager

The Schema Manager is the data definition component. It can accept input in the form of an implementation schema specification in a format unique to the Schema Manager called the Batch Input Language (BIL). The Schema Manager is able to read and process the BIL schema definition into the physical implementation files that define the schema to the system components and the applications. The outputs of the Schema Manager are those physical files. They are the Data Dictionary Index, the Data Dictionary Data, and the Pascal Include File. The Data Dictionary contains all the entity type definitions in the implementation schema along with a description of their attributes, attribute types, and physical locations. The Data Dictionary is dynamically refered to by the system components and application programs that require knowledge of the schema structures. The Pascal Include File contains much the same information as does the Data Dictionary, except it is used to define the schema types and structures to Pascal application programs and provides compile-time binding to the schema definition.

The Schema Manager also can create certain reports that provide valuable references for validation of the schema implementation, and schema implementation structure information needed in application development.

Data Dictionary

The Data Dictionary is the physical schema implementation file that defines the implementation of target conceptual schema to the GMAP system components and certain application programs. In this case, the conceptual schema is the PDES specification. This information is usually referenced in a dynamic, or runtime binding manner. The Data Dictionary contains the structuring information for the implementation structuring of entities in the schema. The entity structure definitions contain entity and attribute names, data types, and physical location information within the entity instances. It also contains the information that can control the required presence or optionality of the particular attributes. If attributes are of aggregate types, it defines the particular aggregation parameters. If an attribute is of an enumeration type, it defines the particular applicable enumeration values. This component plays a role in all the product data requirements: data definition, application support, and data exchange. It is created automatically by the Schema Manager from the schema definition input. The Data Dictionary satisfies the data definition requirement for the system components and applications that choose to use the information in this form.

Pascal Include File

The Pascal Include File contains much the same information as does the Data Dictionary. It is designed to be used by Pascal application programs at compile time to define the data structures of the implementation schema to the application. It allows Pascal application programs to refer to the schema structure elements in a symbolic manner. The Pascal Include File is also created automatically by the Schema Manager from the schema definition input. The Pascal Include File provides data definition to Pascal applications.

Model Access Software

The Model Access Software (MAS) is a library of routines that allow the access, navigation, and manipulation of the Working Form product model. The functionalities of the routines include creation, deletion, modification, and retrieval of entities in the Working Form. It also provides the ability to navigate along relationship paths in the product model to access entities and attributes that are linked to other entities. There is a complete set of list operations available for managing the operations of the Model Access Software. The Model Access Software plays a role in all three product data system requirements: data definition, application support, and data exchange.

Name / Value Interface

The Name/Value Interface (NVI) is a set of routines that are extensions to the Model Access Software. They provide direct store, direct query, and procedural query capabilities to the application at the attribute level of granularity. The application does not need to have compile-time binding to the schema structure as it does with MAS use with the Pascal Include File. The application provides the qualified path name of an attribute of interest, and needs to know the data type of the attribute to interpret the returned data from the Name/Value Interface. The NVI uses the Data Dictionary to gain data definition knowledge, and formulates the required MAS calls to perform the required function. The primary use of the Name/Value Interface is in the role of application support.

Application Program

The Application Program is any life-cycle functional application that uses the Model Access Software and/or the Name/Value Interface to access the data it requires from a product model in the Working Form. In this role, applications such as product modelers, Numerical Control tool path generators, and the like, interface to the product model.

System Translator

The System Translator is the system component that allows the data exchange between dissimilar hardware platforms. To minimize the confusion between PDES physical files of models for communication and the PDES schema implementation physical files for the GMAP software, the PDES physical file will be called the PDES Exchange File. The System Translator is capable of postprocessing a product model in PDES Exchange File format into a Working Form model on the system. The System Translator is also capable of preprocessing a product model in Working Form into a PDES Exchange File. The System Translator accomplishes these tasks through the Model Access Software to manipulate the Working Form, and the Data Dictionary to retrieve schema definition information to guide its operation. It is accessable through the Product Information Exchange System application and can also be linked into user programs.

PDDI Interim Database

The PDDI Interim Database (PID) is a routine that allows an application to store a machine dependent Working Form product model to disk, and to retrieve a disk model back into an active Working Form model. This capability allows preservation of the model on a host platform without having to utilize the System Translator and Exchange File to translate between transient Working Form and persistent Exchange File forms of a product model. It is used by applications and the Product Information Exchange System application.

Product Information Exchange System

The Product Information Exchange System (PIES) is an executable application program that allows manipulation and verification of a product model. It can accept as input a product model PID file and/or a PDES Exchange File. It allows the user to run the System Translator to translate bidirectionally between the Working Form and the PDES Exchange File. It also provides for the verification of product models against the Data Dictionary. It checks for properly sized and structured entities and correct entity relations. The PIES also provides several analysis tools such as creating network hierarchies of constituent or user entities. It allows data query on the entities, and can provide overall product model statistics, such as total size of a model, number of entities, and distribution of types of entities in the model.

Getting PDES Into The GMAP Implementation

In creating the environment for PDES to be used as the schema under the GMAP System Architecture, it was necessary to create an implementation schema specification from the PDES specification that the Schema Manager would accept. This functionality was provided by the National Institute of Standards and Technology. They have developed a parser for the EXPRESS specification language. This parser is called FEDEX [2], and runs in the Unix environment on Sun Workstations. The PDES EXPRESS specification was analyzed and corrected until it would successfully parse under FEDEX. FEDEX has a number of output modules, one of which was developed to create a .BIL file as Schema Manager input. The FEDEX output module for GMAP also creates an .AKA or alias file that contains conceptual schema identifiers and implementation schema identifiers. These identifiers had to be shortened due to compiler restrictions on unique length of identifiers. The resultant PDES.BIL file from FEDEX was then successfully processed by the Schema Manager to create the physical implementation files that allow the system to utilize the PDES schema.

The GMAP Schema Implementation of PDES

This section describes the physical implementation of the PDES schema specification under the GMAP System Architecture. It outlines the implementation constructs, and the application view of those constructs. The EXPRESS language specification of a schema utilizes several information modeling techniques and constructs to convey information content. In this case, the EXPRESS specification of interest is the PDES specification [1]. The PDES specification contains the conceptual structuring, organization, and content of a schema that supports several product data disciplines. The implementation of these conceptual constructs within the GMAP System Architecture is described.

The use of the information modeling construct of categorization is supported in the GMAP system. Categorization involves the abstraction of several categories of entities into a generalized construct. It also involves inheritance of the attributes of the generic entity by all the category entities. This construct in the EXPRESS language is the SUPERTYPE/SUBTYPE construct. All supertype entities in the schema specification are contained in the schema model created by the Schema Manager. In the physical implementation files, no supertype entities exist. All attributes are distributed to the category, or subtype entities in succession, resulting in the implementation of only the leaf entities of the defined inheritance chains. However, there is a GMAP implementation construct that supports the generic reference to the supertype entities, called the Class construct. For every supertype in the conceptual schema, a GMAP Class contruct is created. The Class is an implementation type that supports attribute references to a supertype. If a given attribute is a class reference, the class further defines, in a recursive manner, all the implementation entities, or other classes containing implementation entities, that are allowable for reference in that context. These constructs exist in the Data Dictionary for reference by the GMAP system components and application programs.

The aggregation attribute type constructs in the conceptual EXPRESS specification of entities are supported in the GMAP implementation. Any attributes that are aggregations of primitive data types other than references to other entities are supported by direct imbedding of the aggregation within the entity ADB structure. Any attributes that are aggregations of entity references are supported by the reference of that attribute to a GMAP internal entity called the ARRAY_ENTITY. The entity's aggregation reference attribute on the entity CL points directly to an ARRAY_ENTITY. The ARRAY_ENTITY collects the references to each element of the aggregation on its CL.

The optional attribute type constructs in the conceptual EXPRESS schema specification are supported in the GMAP implemention. The absence of an optional primitive data type needs to be compensated by the addition of an attribute that signals the presence or absence of an optional attribute. In the present condition, these flags are not in the GMAP implementation of the PDES specification unless they were explicitly defined in the PDES schema. However, attributes that are optional pointers or references to other entities are handled. If an optional entity reference is not present, the attribute is a reference to a GMAP internal entity, the NIL_ENTITY. The use of the NIL_ENTITY signals that the particular attribute has been opted to be absent. If the optional entity reference attribute is indeed present, it will point to an instance of the appropriate entity type.

The situation can be extended to the support of optional aggregation attributes. In the case of optional aggregations of primitive data types, the inclusion of additional attributes that signal the number of elements that are present for the particular attribute instance are suggested. Again, these flag attributes are not included in the GMAP PDES implementation, except where they are defined explicitly in the PDES specification. However, the support of optional aggregation attributes that reference other entities is available under GMAP. The particular attribute will reference a GMAP internal entity, the ARRAY_ENTITY as above, except that the ARRAY_ENTITY will contain no CL references to aggregation elements, effectively representing a null list. This implementation strategy melds nicely with the Model Access Software capability to detect the number of references an entity contains.

The setting of upper limits for implementation in the GMAP System Architecture was one addition that was required to map the PDES conceptual specification into an implementation schema. This type of information was not present in the PDES specification, but should be required of the final PDES specification if consistent and verifiable implementations are to be expected.

The specification of all attribute and user defined types that were of type STRING in the PDES specification were implemented as STRING(80). All string types in the GMAP implementation are character arrays, and not compiler or system dependent implementations of strings due to the need for inter-system portability. If a STRING type had a precision qualifier, it was preserved as that maximum length, otherwise it was specified as having a length of 80 characters. In this respect, the EXPRESS language qualifier VARYING on STRING types is ignored.

The attributes in the PDES conceptual schema that were defined as aggregations specifying infinity (#) as the upper bound were implemented with an arbitrary upper bound of 255 elements. This controls the fixed allocation and total length of entity ADBs that contain such attribute definitions, creating a finite limit to the number of elements that these attributes may contain. However, the use of the upper limit of 255 does not limit the actual number of attributes of the aggregation reference or pointer type, because these are contained on ARRAY_ENTITY Constituent Lists, and as such are dynamically allocated, and only limited by the amount of memory available to the Working Form.

DEVELOPING PDES TEST APPLICATIONS

Information on how to access the NIST AMRF VAX is included as Appendix A. All GMAP software organization and location information is included as Appendix B.

The basic approach to using the GMAP System Architecture as a PDES implementation involves the creation of a PDES product model, verification of the PDES product model, development of applications using the PDES product model, and exchange of the PDES product model between systems.

The creation of PDES product models begins with the design and specification of the model based on the PDES specification itself, and the specific product design requirements. It is suggested that each and every entity instance in the product model be included in a specification along with each attribute value in a dependent order. Specify entities upon which other entities depend for definition before the dependent entities are specified.

From such a specification, a product model can be created using the very primitive modeling program written as a utility application program on the system. This program is called MODEL. MODEL uses Data Dictionary information to prompt the user through the creation of entities and attributes from the prepared model specification. Please note that there is a more sophisticated interactive graphics product editor available as a GMAP deliverable on the IBM computer platforms. This editor has not been ported to the VAX environment under this research agreement.

Once the PDES product model has been created, it can be verified using the PIES utilities of the GMAP software. This will allow checking of the model for proper sizing, structure, and references of each entity in the product model. Any errors in the model can be corrected or remodeled using the modeling program.

After a valid PDES product model is prepared, life-cycle functional applications can be developed. These programs are written to access, manipulate, and navigate through the product model for their data requirements. The application programs interface to the product model by using the Model Access Software and the Name/Value Interface. Refer to the GMAP Model Access Software Users Manual for the details on the individual routines. With the manual and the Physical Schema Report, generated by the Schema Manager as a guide, it is possible to develop application programs of any complexity. There are a few examples of applications available on the system for a tutorial. Some are included in this document for review as Appendix C.

Once the application is developed, it can be executed, and whatever it was designed to accomplish can be objectively evaluated. This type of testing is the ideal way to discover weaknesses, problems, or other rough areas that exist within the PDES specification. It also demonstrates the benefit and value of having such intelligent product model data available to applications. Applications that have previously been impossible due to data starvation may be enabled by the existence of complete PDES product models.

The exchange of the PDES product models between dissimilar hardware platforms can then be accomplished by utilizing the GMAP System Translator accessed through the PIES program. The Working Form PDES model as stored on disk by the PID can be translated to a PDES Exchange File, transmitted to another hardware platform. There another installation of the GMAP System Architecture, or another system altogether can use the PDES Exchange File to accomplish its tasks.

GMAP DOCUMENTATION

GMAP System Components Operators Manual	OM560240001U
GMAP Model Access Software Users Manual	UM560240031U
GMAP System Translator Users Manual	UM560240021U
GMAP Schema Manager Users Manual	UM560240011U
GMAP Product Information Exchange Sysytem Users Manual	TTD560130002

These documents are thought to be the primary references for anyone who desires to create application programs to test the PDES specification under the GMAP System Architecture as it exists on the NIST AMRF VAX. These documents, the entire set of deliverable documents, and the deliverable software source code are available through the United States Air Force. Details on how to order any of the above can be obtained from the USAF by writing to:

Charles Gilman
AFWAL/MLTC
Wright-Patterson AFB, OH 45433-6533
513-255-7371

or:

AFWAL/MLTC
ICAM Program Library
Wright-Patterson AFB, OH 45433

Request the GMAP / PDDI Deliverables Roadmap Document to get an overview of all the GMAP documents, software, and other contract deliverables available to industry.

SUMMARY

While this document is intended to provide enough initial information to be able to utilize the GMAP software environment on the NIST AMRF VAX to test and validate the current PDES specification, it is likely that this document will create more questions than it answers. In order to obtain the technical advice that might be needed to capitalize on the capabilities that the GMAP System Architecture provides, you are encouraged to contact the author:

Kim L. Perlotto
Senior CAD/CAM Engineer
CAD/CAM Technology Group
Development Operations
Pratt & Whitney MS118-38
400 Main Street
East Hartford, Connecticut 06108
203-565-4254

For more information on the National PDES Testbed Project at the National Institute of Standards and Technology, please feel free to contact:

Cita Furlani
Group Leader - National PDES Testbed Project
Factory Automation Systems Division
Center for Manufacturing Engineering
National Engineering Labratory
NIST
Building 220 Room A-127
Gaithersburg, MD 20899
301-975-3543

BIBLIOGRAPHY

[1] Bradford Smith, "Product Data Exchange Specification: First Working Draft", NISTIR 88-4004, NTIS order number PB 89-144794.

[2] Steven Clark, "FedEx: The NIST Express Parser", May 19, 1989 Draft, Factory Automation Systems Division, NIST.

APPENDIX -A-

ACCESSING THE GMAP SOFTWARE ON THE NIST AMRF VAX

Personal computer access to the NIST AMRF VAX can be obtained by using most any communications software package that supports VT100 emulation by dialing 301-258-8996 with modem parameters of 1200-N-8-1. User IDs and other access abilities can be arranged by contacting the NIST AMRF VAX System Manager, Micky Potts, at 301-975-3537.

APPENDIX -B-

GMAP SOFTWARE ORGANIZATION ON NIST AMRF VAX
==========

```
PDM$DISK:[GMAP]                - Root directory for all GMAP files
PDM$DISK:[GMAP.PW]             - Contains all P&W supplied files, including
                                   PDES physical implementation files
PDM$DISK:[GMAP.V40]            - Root directory for GMAP V4.0 software

PDM$DISK:[GMAP.V40.COMFIL]     - Contains all .COM files for building
                                   system and running software
PDM$DISK:[GMAP.V40.DDFILS]     - Contains system files for GMAP and PDDI

PDM$DISK:[GMAP.V40.MASINC]     - Include files for Model Access Software
PDM$DISK:[GMAP.V40.MASSRC]     - Source files for Model Access Software
PDM$DISK:[GMAP.V40.MASOLB]     - Object library for Model Access Software

PDM$DISK:[GMAP.V40.NVIINC]     - Include files for Name/Value Interface
PDM$DISK:[GMAP.V40.NVISRC]     - Source files for Name/Value Interface
PDM$DISK:[GMAP.V40.NVIOLB]     - Object library for Name/Value Interface

PDM$DISK:[GMAP.V40.TRNINC]     - Include files for System Translator
PDM$DISK:[GMAP.V40.TRNSRC]     - Source files for System Translator
PDM$DISK:[GMAP.V40.TRNOLB]     - Object library for System Translator

PDM$DISK:[GMAP.V40.PIESINC]    - Include files for PIES
PDM$DISK:[GMAP.V40.PIESSRC]    - Source files for PIES
PDM$DISK:[GMAP.V40.PIESOLB]    - Object library for PIES
PDM$DISK:[GMAP.V40.PIESCOM]    - Executable for PIES

PDM$DISK:[GMAP.V40.PIDINC]     - Include files for PDDI Interim Database
PDM$DISK:[GMAP.V40.PIDSRC]     - Source files for PDDI Interim Database
PDM$DISK:[GMAP.V40.PIDOLB]     - Object library for PDDI Interim Database

PDM$DISK:[GMAP.V40.SMBINC]     - Include files for Schema Manager - Batch
PDM$DISK:[GMAP.V40.SMBSRC]     - Source files for Schema Manager - Batch
PDM$DISK:[GMAP.V40.SMBOLB]     - Object library for Schema Manager - Batch

PDM$DISK:[GMAP.V40.RTSINC]     - Include files - Schema Manager Runtime Subsch
PDM$DISK:[GMAP.V40.RTSSRC]     - Source files - Schema Manager Runtime Subsche
PDM$DISK:[GMAP.V40.RTSOLB]     - Object library - Schema Manager Runtime Subsc
```

INTERSTING FILES IN GMAP DIRECTORIES
====================================

PDM$DISK:[GMAP.PW]
```
    PDES.EXP      - PDES EXPRESS - full specification, syntax corrected
    PDES.BIL      - PDES in GMAP Schema Manager Batch Input Language
    PDES.AKA      - PDES alias file, conceptual and implementation names
    PDES.DDI      - PDES Data Dictionary Index file
    PDES.DDD      - PDES Data Dictionary Data file
    PDES.PIF      - PDES Pascal Include File
    PDES.PSR      - PDES Physical Schema Report - Guide to implementation
    PDES.CSR      - PDES Conceptual Schema Report - schema exchange file
    PDESPIES.COM - .COM file to run PIES under PDES schema
    RUN_SMT.COM   - .COM file to run the Schema Manager
    NVIINT.COM    - An interactive program that uses NVI to query a model
    MODEL.COM     - A primitive alpha-test program to create a PDES model
    PDESPTS.PAS   - A simple example of a program to create 100 points
    PDESLINE.PAS - A simple program that creates two points and a line
    PDESNVI.PAS   - A simple demo uses NVI to query the line model
    PDESDISK.SPC - A specification of a feature-based PDES product model
    PDESDISK.WF   - The Working Form of the model
    PDESDISK.EF   - The Exchange File of the model
```

PDM$DISK:[GMAP.V40.DDFILS]
```
    GMAP_DDI.DAT - GMAP Data Dictionary Index file
    GMAP_DDD.DAT - GMAP Data Dictionary Data file
    GMAP.PIF      - GMAP Pascal Include File
    PDDI_DDI.DAT - PDDI Data Dictionary Index file
    PDDI_DDD.DAT - PDDI Data Dictionary Data file
```

PDM$DISK:[GMAP.V40.PIESCOM]
```
    RUN_PIES.COM - .COM file to run PIES under GMAP schema
```

APPENDIX -C-

```
(*********************************************************************)
(* PDESPTS.PAS -- a program to create PDES CARTESIAN_POINTs using MAS *)
(*********************************************************************)
program pdespts(input,output);
const
    NUMPTS = 100;
(* type definitions supporting application use of MAS *)
%include '[gmap.v40.masinc]apltyp.inc/nolist'
(* PDES Pascal Include File - entity type definitions *)
%include '[gmap.pw]pdes.pif/nolist'
var
    adb : entblock;       (* entity data area for application use *)
    rc  : ext_ret_code;   (* MAS procedure return code variable *)
    key,nilkey : entkey; (* MAS entity key variable *)
    i   : integer;
(*********************************************************************)
(* include files to define external MAS and PID procedures used      *)
(*********************************************************************)
%include '[gmap.v40.pidinc]filrtv.inc/nolist' external; (* PID *)
%include '[gmap.v40.masinc]mainit.inc/nolist' external; (* MAS init *)
%include '[gmap.v40.masinc]maecr.inc/nolist'  external; (* MAS create *)
%include '[gmap.v40.masinc]malatc.inc/nolist' external; (* MAS ent rf *)
%include '[gmap.v40.masinc]makill.inc/nolist' external; (* MAS term *)
(*********************************************************************)

(**** main program ****)

begin
    writeln('Program to create ',NUMPTS:1,
      ' PDES CARTESIAN_POINT entities.');
    mainit(rc);                          (* initialize MAS environment *)
    adb.kind := K_NIL_ENTITY;
    adb.length := sizeof(entblock,K_NIL_ENTITY);
    adb.sysuse := 0;
    adb.ident := 1;
    maecr(adb,0,nilkey,rc);               (* create the entity *)
    for i := 1 to NUMPTS do begin
        adb.kind := K_CARTESIAN_POINT;
        adb.length := sizeof(entblock,K_CARTESIAN_POINT);
        adb.sysuse := 0;
        adb.ident := i;
        adb.cartesian_point.x_coordinate := i;
        adb.cartesian_point.y_coordinate := i;
        adb.cartesian_point.z_coordinate := i;
        adb.cartesian_point.space := 3;
        maecr(adb,0,key,rc);               (* create the entity *)
        malatc(k,nilkey,rc);     (* this is optional geo_lclcrdntsys *)
        writeln(' PDES CARTESIAN_POINT #',adb.ident:1,' created.');
    end;
    filtrv(0,rc);                        (* file working form to disk *)
    writeln('PDES PDD model of ',NUMPTS:1,
      ' CARTESIAN_POINT entities filed.');
    makill(rc);                          (* terminate MAS environment *)
end.
```

```
(*************************************************************************)
(* PDESLINE.PAS - program to create two CARTESIAN_POINTs                 *)
(*                and a LINE_SEGMENT.                                    *)
(*************************************************************************)
program pdesline(input,output);
(* type definitions supporting application use of MAS *)
%include '[gmap.v40.masinc]apltyp.inc/nolist'
(* PDES Pascal Include File - entity type definitions *)
%include '[gmap.pw]pdes.pif/nolist'
var
    adb : entblock;                        (* application data block *)
    rc  : ext_ret_code;                    (* return code for MAS calls *)
    p0,p1,l1.nilkey,arraykey : entkey;     (* entity keys for access *)
    aryid : integer;
(*-----------------------------------------------------------------------*)
(* includes for external MAS and PID procedures                          *)
(*-----------------------------------------------------------------------*)
%include '[gmap.v40.pidinc]filrtv.inc/nolist' external; (* PID *)
%include '[gmap.v40.masinc]mainit.inc/nolist' external; (* MAS init *)
%include '[gmap.v40.masinc]maecr.inc/nolist'  external; (* MAS create *)
%include '[gmap.v40.masinc]malatc.inc/nolist' external; (* MAS ent ref *)
%include '[gmap.v40.masinc]makill.inc/nolist' external; (* MAS term *)
(*-----------------------------------------------------------------------*)
(* internal procedures                                                   *)
(*-----------------------------------------------------------------------*)
procedure createnil;
begin
    adb.kind := K_NIL_ENTITY;
    adb.length := sizeof(entblock,K_NIL_ENTITY);
    adb.sysuse := 0;
    adb.ident := 1;
    maecr(adb,0,nilkey,rc);                (* create the entity *)
end;
(*-----------------------------------------------------------------------*)
procedure creatept(i:integer;x,y,z:double;var k:entkey);
begin
    adb.kind := K_CARTESIAN_POINT;
    adb.length := sizeof(entblock,K_CARTESIAN_POINT);
    adb.sysuse := 0;
    adb.ident := i;
    adb.cartesian_point.x_coordinate := i;
    adb.cartesian_point.y_coordinate := i;
    adb.cartesian_point.z_coordinate := i;
    adb.cartesian_point.space := 3;
    maecr(adb,0,k,rc);                  (* create the entity *)
    malatc(k,nilkey,rc);          (* this is optional geo_lclcrdntsys *)
end;
(*-----------------------------------------------------------------------*)
```

(* pdesline continued on next page *)

```
(* pdesline continued *)

procedure createline(i:integer;p0,p1:entkey;var k:entkey);
begin
    adb.kind := K_LINE_SEGMENT;
    adb.length := sizeof(entblock,K_LINE_SEGMENT);
    adb.sysuse := 0;
    adb.ident := i;
    maecr(adb,0,k,rc);
    adb.kind := K_ARRAY_ENTITY;
    adb.length := sizeof(entblock,K_ARRAY_ENTITY);
    adb.sysuse := 0;
    aryid := aryid + 1;
    adb.ident := aryid;
    maecr(adb,0,arraykey,rc);    (* create ARRAY_ENTITY ADB *)
    malatc(arraykey,p0,rc);      (* attach 1st point *)
    malatc(arraykey,p1,rc);      (* attach 2nd point *)
    malatc(k,nilkey,rc);         (* this is optional geo_lclcrdntsys *)
    malatc(k,arraykey,rc);       (* this is POINTS attribute *)
end;
(*-------------------------------------------------------------------------*)

(**** main program ****)

begin
    aryid := 0;
    writeln('creating two PDES CARTESIAN_POINTs and a LINE_SEGMENT...');
    writeln;
    mainit(rc);                              (* MAS initialzation *)
    createnil;                               (* create NIL_ENTITY *)
    creatept(1, 1.100, 1.200, 1.300, p0);    (* create CARTESIAN_POINT *)
    creatept(2, 2.100, 2.200, 2.300, p1);    (* create CARTESIAN_POINT *)
    createline(1, p0, p1, l1);               (* create LINE_SEGMENT *)
    filrtv(0,rc);                            (* file model to disk *)
    makill(rc);                              (* MAS termination *)
end;
```

```
(*********************************************************************)
(* PDESNVI.PAS -- program using NVI on a PDES model of two           *)
(*                CARTESIAN_POINTs and a LINE_SEGMENT                *)
(*********************************************************************)
program pdesnvi(input,output);
(* type definitions supporting application use of MAS *)
%include '[gmap.v40.masinc]apltyp.inc/nolist'
(* PDES Pascal Include File - entity type definitions *)
%include '[gmap.pw]pdes.pif/nolist'
(* type definitions supporting application use of NVI *)
%include '[gmap.v40.masinc]nviaptyp.inc/nolist'
type
    strng = varying[80] of char;
(******************** INTERNAL VARIABLE DECLARATIONS ******************)
var
    adb : entblock;
    nam : t_attribute_name;
    val : t_attribute_value;
    dim : t_dimen_value;
    ll,key : entkey;
    sl,cl : listkey;
    i,n : integer;
    name : strng;
    rc : ext_ret_code;
    ddinx,ddfile : [common]text;
(******************** EXTERNAL MAS & NVI PROCEDURES *******************)
%include '[gmap.v40.masinc]malk.inc/nolist' external;  (* make kind list *)
%include '[gmap.v40.masinc]malno.inc/nolist' external; (* count entries *)
%include '[gmap.v40.masinc]malgtk.inc/nolist' external;(* get key<list *)
%include '[gmap.v40.masinc]maegtk.inc/nolist' external;(* get adb *)
%include '[gmap.v40.masinc]maldi.inc/nolist' external; (* delete lists *)
%include '[gmap.v40.masinc]makill.inc/nolist' external;(* MAS term *)
%include '[gmap.v40.nviinc]nvdqan.inc/nolist' external;(* direct query *)
%include '[gmap.v40.nviinc]nvdsav.inc/nolist' external;(* direct store *)
%include '[gmap.v40.nviinc]nvpqav.inc/nolist' external;(* proc query *)
%include '[gmap.v40.pidinc]filrtv.inc/nolist' external;(* file/rtrve *)
(*********************************************************************)
function str2ary(n:strng):t_attribute_name;
var
    i:integer;
begin
    name := n + END_OF_STRING;
    for i := 1 to length(name) do
        nam(.i.) := name(.i.);
    str2ary := nam;
end;
(*********************************************************************)
```

(* pdesnvi continued on next page *)

```
(* pdesnvi continued *)

(**** main program ****)

begin
    open(ddinx,'ddinx',readonly,error:=message);
    open(ddfile,'ddfile',readonly,,direct,error:=message);
    filrtv(1,rc);    (* retrieve model from disk, this initializes MAS *)
    malk(K_LINE_SEGMENT,cl,rc); (* make list of all LINE_SEGMENTs *)
    malgtk(cl,1,l1,rc);          (* get 1st LINE_SEGMENT's key *)
    writeln('---- NVI DIRECT QUERY ----');
    dim(.1.) := 1;          (* index of attribute of imterest *)
    nvdqan(l1,str2ary('POINTS(1).IDENT'),dim,val,rc);
    writeln('LINE_SEGMENT.1 has a POINTS(1).IDENT of ',val.as_integer:1);
    writeln;
    writeln('---- NVI DIRECT STORE ----');
    writeln('storing 9.99999 to LINE_SEGMENT.1''s POINTS(1).Y_COORDINATE.')
    val.as_real_8 := 9.99999;
    dim(.1.) := 1;          (* index of attribute of imterest *)
    nvdsav(l1,str2ary('POINTS(1).Y_COORDINATE'),dim,val,rc);
    writeln;
    writeln('---- NVI PROCEDURAL QUERY ----');
    write('looking for CARTESIAN_POINTs with a');
    writeln(' Z_COORDINATE <= 1.300...');
    malk(K_CARTESIAN_POINT,cl,rc); (* make a list of all CARTESIAN_POINTs *
    malno(cl,n,rc);                (* count entries on candidate list *)
    writeln('number of LINE_SEGMENTs on candidate list is ',n:1);
    val.as_real_8 := 1.300;
    dim(.1.) := 0;          (* index of attribute of imterest *)
    nvpqav(cl,str2ary('Z_COORDINATE'),val,dim,LE,sl,rc);
    malno(sl,n,rc);                (* count entries on selected list *)
    writeln('number of LINE_SEGMENTs on selected list is ',n:1);
    for i := 1 to n do begin
        malgtk(sl,i,k,rc);         (* get a key off selected list *)
        maegtk(k,adb,rc);          (* get adb of that entity *)
        writeln('    CARTESIAN_POINT-',adb.ident:1,
           ' has a Z_COORDINATE <= ',val.as_real_8:7:5);
    end;
    maldi(cl,rc);                  (* delete all application lists *)
    makill(rc);                    (* terminate MAS environment *)
end.
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NISTIR 89-4117	2. Performing Organ. Report No.	3. Publication Date JUNE 1989

4. TITLE AND SUBTITLE

The Use of GMAP Software as a PDES Environment in the National PDES Testbed Project

5. AUTHOR(S)

Kim L. Perlotto

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This report is a basic guide to the use of the GMAP System Architecture as installed on the NIST AMRF VAX as part of the National PDES Testbed Project. An overview of the GMAP System Architecture is provided. The use of the GMAP software to create an implementation environment for the PDES Draft Proposal Specification (February 1989) is outlined. The software organization on the NIST AMRF VAX and the development of test and validation applications are described.

The GMAP System Architecture consists of software system components which meet the three basic requirements of an automated product data environment. The requirements are data definition, application support, and data exchange. The system components are defined and the role they play is described.

GMAP is the Geometric Modeling Applications Interface Program, a United States Air Force contract which Pratt & Whitney, United Technologies, was the prime contractor. The installation of the GMAP software deliverables at the National Institute of Standards and Technology was possible under a technology transfer clause of the GMAP contract.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Product Data Exchange, PDES, GMAP, Computer-Aided Manufacturing, Computer-Aided Design, CAD, CAM, Factory Automation

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES 25

15. Price $9.95

b