Reuse Methodology and Implementation Appnote

Abstract

In today's engineering design environment, designers are limited in their ability to maximize reuse by the fact that there is no efficient way to search for, access, and integrate reusable design objects across multiple sources; frequently, these potential sources of reusable design data are uncoupled from the design environment. This paper details an approach for managing reusable design objects in a collaborative engineering environment that enables Rapid Prototyping of Application-Specific Signal Processors (RASSP) and the architecture of the RASSP Reuse Data Manager (RRDM), specifically developed to support this approach.

Key aspects of our approach include:

- developing processes and methodologies for design-for-reuse and design-with-reuse
- developing a rich RASSP reuse classification hierarchy, or ontology, that rigorously models the various types of design and data objects in the RASSP domain
- developing a RASSP Reuse Data Manger (RRDM) that facilitates integration of legacy design data into a reuse repository using knowledge-based approaches for resolving inconsistencies and distinctions among terms, supports implementation of the RASSP reuse classification hierarchy, and provides mechanisms for searching for design objects across multiple libraries in a virtual enterprise

Purpose

The RASSP reuse processes were developed using the IDEF3 methodology for process modeling. The models specify the essential activities to be performed when designing for and with reuse. These generic processes may be adapted by an organization implementing design reuse and customized to their specific needs.

The RASSP Reuse Classification Hierarchy (RRCH) was initially developed using Rumbaugh's Object Modeling Technique (OMT). Subsequently, it was reorganized and extended as a set of ontology components using the Ontolingua toolset developed by the Stanford University Knowledge Systems Laboratory (KSL). The RRCH defines the terms relevant to the domain and the relationships among them to assist a designer in searching for and selecting design objects from the reuse repository. The focus of the RRCH development has been on the electronics design domain. Although, this approach applies to many other engineering and integrated manufacturing domains.

Sources of reusable designs may include those created within an engineering design organization, CAD tool libraries, CAD tool-independent libraries, released designs managed by product data management (PDM) systems across the enterprise, related business and engineering information, and component vendor data, among others. To query and access these heterogeneous sources of design data, the RRDM provides designers with a common view of the *virtual* reuse repository. Syntactic and semantic differences among the various source file systems, libraries and repositories are resolved through a common vocabulary representing their union and mapping algorithms that relate the source vocabularies to the common view. An ontology for a particular source repository defines the relevant terms for that source -- identifying attributes of the terms, relationships among terms, and constraints that pertain to those terms. We describe in this paper the architecture and concept of operations of the environment that we are building to support legacy data integration using knowledge-based representation and standards-based enterprise integration concepts.

Incremental versions of the reuse processes, ontology components and RRDM functions were demonstrated for the commercial space communications domain (Lockheed Martin Astrospace Company) and for the VHDL model development and reuse domain. The VHDL model application supports model repositories at Lockheed

Martin ATL, SCRA, and several universities. In this paper we will describe the specific processes applied and the lessons learned from the exercise.

.

Roadmap

Reuse Methodology and Implementation Appnote

1.0 Introduction

In today's engineering design environment, designers are limited in their ability to maximize reuse by the fact that there is no efficient way to search for, access, and integrate reusable design objects across multiple sources; frequently, these potential sources of reusable design data are uncoupled from the design environment. Also lacking are mechanisms and processes for organizing reusable design information created within a design team, and for effectively sharing that design knowledge within the organization as well as with other cooperating organizations

No two product data management (PDM) system implementations are alike, though up to 90% of the underlying database schema may be generic, for example. Every local design organization has their own classification scheme for designs developed internally, even in cases where the use of a single software product, such as the Metaphase Product Data Management System [SDRC_1997], which provides the configuration management functionality for the RASSP Enterprise Environment, has been mandated by a corporate engineering process improvement directive. Integration of multiple instances of the Metaphase system is possible in cases where the classification scheme is the same for all instances, but not easily otherwise. Costly, unique point-to-point integration is required in most cases where multiple applications and databases are involved today.

We describe in this paper an approach for integrating multiple, diverse sources of engineering and business data to provide a single entry point for searching for reusable design knowledge and to enable collaborative engineering throughout the enterprise. This approach was developed and prototyped for the DARPA Rapid Prototyping of Application-Specific Signal Processors (RASSP) program.

The Rapid Prototyping of Application-Specific Signal Processors (RASSP) is a Defense Advanced Research Projects Agency (DARPA)/Tri-Service program aimed at dramatically improving productivity in the design, manufacture, test, and procurement of digital signal processors. RASSP products include an enterprise system, which integrates the CAD tools used in the RASSP design process, workflow tools for managing the design process, and the tools for managing the design data. Reuse data management in the RASSP system involves the generation, organization, distribution, and use of reusable design knowledge. Sources of reusable designs in the RASSP environment may include those created within an engineering design organization, CAD tool libraries, CAD tool-independent libraries, released designs managed by product data management (PDM) systems across the enterprise, related business and engineering information, and component vendor data, among others.

Key aspects of our approach for reuse data management include:

- developing processes and methodologies for design-for-reuse and design-with-reuse
- developing a RASSP reuse classification hierarchy that models the various types of design and data objects in the RASSP domain
- developing a RASSP Reuse Data Manger (RRDM) that facilitates integration of legacy design data into a reuse repository using knowledge-based approaches for resolving inconsistencies and distinctions among terms, supports implementation of the RASSP reuse classification hierarchy, and provides mechanisms for searching for design objects across multiple libraries in a virtual enterprise

This approach was applied to the satellite communications domain, relevant to the Lockheed Martin

Astrospace company in Newtown, PA, and to the integration of multiple VHDL model repositories developed by several organizations associated with the RASSP program. Two prototype implementations of the RASSP Reuse Data Manager (RRDM) were created for the satellite communications application, both of which included a web-based front-end to the reuse repository. The early prototype stored all meta-data as text in HTML files, while the second uses an object-oriented database engine for persistent storage of both the common vocabulary and reusable design knowledge. Both demonstrated subsets of the target features of the RRDM. The VHDL model demonstration extended the original RASSP reuse classification hierarchy (RRCH) to support many types of VHDL models and related design data. The extended RRCH is compliant with the RASSP VHDL Modeling Terminology and Taxonomy recently adopted by the VHDL user community. Based on lessons learned from the prototype implementations we have further refined the architecture and concept of operations for the RRDM. The underlying technology of the RRDM was developed and is being commercialized by Sandpiper Software, Inc., with partial funding from the RASSP program.

In the next section we describe the architecture of the RASSP enterprise system, and the role of the RRDM in that environment. In section 3 we describe the processes and methodology developed on the program that are specific to integrating new or modified designs with the reuse system and accessing those designs for use in new applications. In section 4 we present a knowledge-based approach for modeling the RASSP domain as well as components of the resultant RASSP reuse classification hierarchy. In section 5 the high-level architecture of the RRDM and its concept of operations is discussed. An application of the RASSP reuse methodology at an operating company and the RRDM prototype demonstrations is the subject of section 6. Section 7 details the key benefits and implications of the technology developed, provides some comparisons with related technologies, and summarizes the significant lessons learned.

Next Up Previous Contents
Next: 2 Architecture of the RASSP Enterprise System **Up:** Appnotes Index **Previous:** Appnote Reuse Index

Reuse Methodology and Implementation Appnote

2.0 Architecture of the RASSP Enterprise System

Figure 2 - 1 shows the data flow architecture of the RASSP enterprise system. A design engineer interacts with a workflow manager to perform various predefined process steps. On execution of a particular process step, the workflow manager may invoke the appropriate CAD tool for the activity, opening the requisite product data files and making them available in the designer's workspace. The workflow manager interacts transparently with the product data manager on behalf of the user to open, close, save and configuration manage the product data files. The user may also interact directly with the desktop manager to invoke tools outside of the workflow environment, as needed

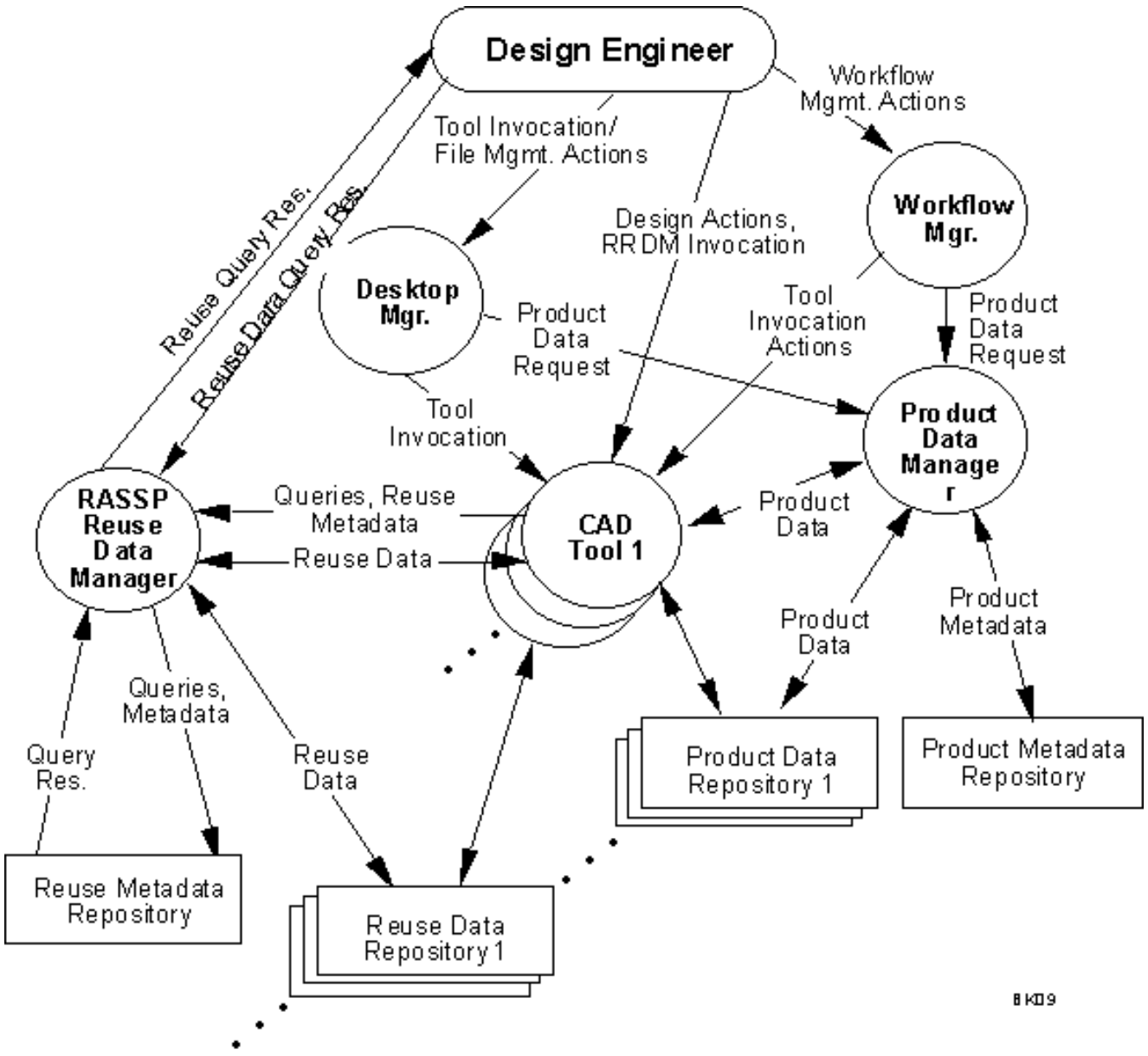


Figure 2 - 1: Enterprise System Data Flow Architecture

Once a CAD tool has been invoked, the designer interacts with the tool in its native environment to perform a design or analysis

activity. He or she may invoke the RRDM through the CAD tool environment or directly. The RRDM may receive reusable design knowledge from the CAD tool, or meta-data only in cases where the design is managed in place. Additional meta-data is solicited from the environment as well as from the designer through the use of domain-specific templates. Both the meta-data and design knowledge (where applicable) are stored in the local design knowledge repository. The workflow manager for the RASSP Enterprise System is implemented through Intergraph Corporation's Design Methodology Manager (DMM), and the product data manager is implemented by the Intergraph Asset information Manager (AIM). Additional information on the RASSP Enterprise System is provided in Enterprise Application Note

The architecture of the RASSP Reuse Data Manager (RRDM) and its relationship to users and other system resources is detailed in Figure 2- 2. The RRDM consists of a client browser, a "perspectives" application server (including the search and vocabulary mapping engines), and multiple source data repositories. A designer looking for reusable design knowledge browses the hierarchy and identifies a potential source for designs of interest. Then, through a custom view of the common vocabulary, the user may specify attributes of the candidate design(s) he or she is interested in. Queries may include arithmetic expressions, logical expressions, attributes of related classes, and constraints on any of the attributes, as appropriate. Mapping algorithms that account for a variety of data inconsistencies, differences in precision, translation of units of measure, and other distinctions among terms across repositories are resolved to the user's customized vocabulary and display requirements. The search of available (and optionally selected) source repositories is performed in parallel, with meta-data for all results displayed through the client browser. Depending on the number and nature of the results returned, he or she may further refine the search by specifying additional attributes and/or more restrictive criteria. Objects of interest may be selected and the corresponding designs (*e.g* ., schematics, drawings, models) displayed in their native tool environments given that appropriate resources are available to the user.

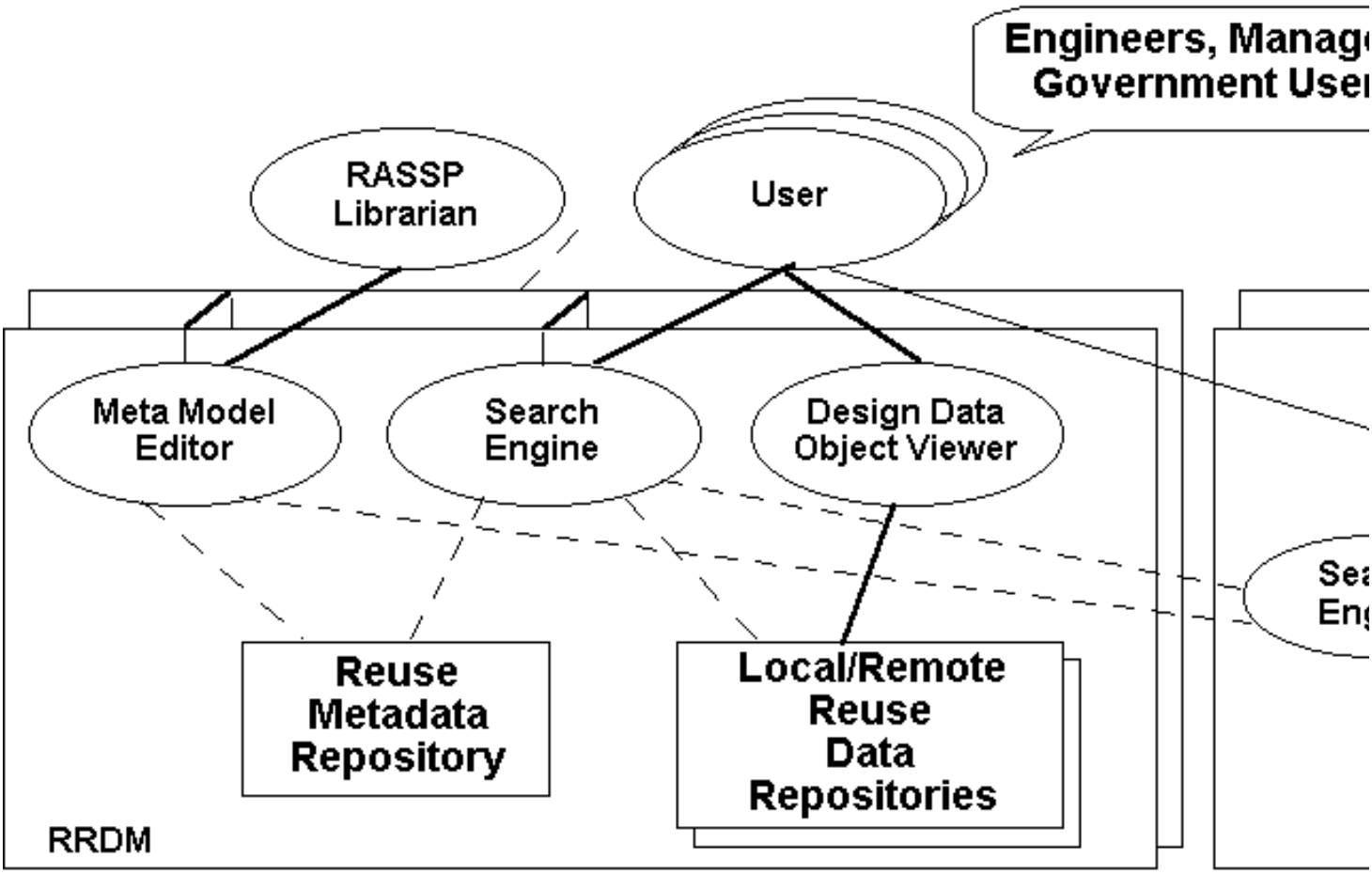


Figure 2 - 2: RASSP REUSE Data Manager Architecture

A prototype of the RASSP Reuse Data Manager (RRDM) was demonstrated in April 1998, providing access to multiple, distributed repositories of VHDL models. The key capabilities featured at this RASSP demonstration are summarized below.

- Data Modeling - Through the use of the Ontolingua tool, the actual ontology components developed in support of the RASSP satellite communications and electronics domain were highlighted. These components share basic information-asset characteristics, such as the date of creation, version, author, and so forth, but are classified uniquely by fit, form and function on the design object hierarchy, and by the nature of the information (*e.g* ., VHDL model, script,

test vector, specification) in an engineering data object hierarchy. For the most part, multiple engineering data objects were collected to define a single design object. The resultant ontology components have been published and made available through the Stanford Knowledge Systems Laboratory (KSL) Ontolingua server.

- Search and Retrieval - The ontology components were implemented and demonstrated in the prototype environment, showing hierarchy and relationship browsing as well as attribute-based query construction and results retrieval. Physical design objects related to the meta-data were also displayed in their native environments. Examples of designs used for demonstration purposes include complex architectural, behavioral, and performance models developed for the RASSP benchmark program, with the relevant documentation, source code, test environments, and drawings where available. Also included were a number of VHDL models developed for the program by several universities and South Carolina Research Authority (SCRA).
- Heterogeneous Integration - Integration of heterogeneous sources (*i.e.*, with syntactically and semantically distinct database schemas) was demonstrated through the use of DBMS vendor tools in addition to the RRDM. A complex query across multiple sources was initiated from the RRDM, and the results examined. Each individual repository was then searched using vendor-provided tools to show not only that there were distinctions in the data stored in each repository but in the terms defined by the repository schemas as well.
- Distributed, Object-Oriented Architecture - The demonstration environment included multiple client browsers hosted on several machines (both PCs and UNIX workstations), multiple back-end repositories hosted on distinct server machines, and a single perspectives server host. The commercial product will support multiple, distributed application servers in order to optimize local performance in addition to distributed clients and source repositories. The prototype provided source repository selection capabilities, which, in contrast with physical examination of individual repositories, demonstrated distributed repository access capabilities.

Planned Enhancements

A number of additional capabilities are planned for the commercial product and subsequent releases by Sandpiper Software, Inc. These include:

- Function-Based Search and Retrieval
- Content-Based Search and Retrieval
- Configuration Management for all ontology components and meta-data with optional release management for design knowledge
- Fine-Grained Authorization Control
- Enhanced User Authentication
- Extensive User Customization Capabilities, including user and role-specific views of the virtual repository as well as support for custom forms, templates, and reports
- Standards-based application integration for third-party tools through CORBA-compliant APIs, including automated data synchronization where required
- Multi-vendor DBMS support (relational, object-relational, and object databases)
- Intelligent agent integration for data mining and decision support, subscription, notification, and other extensions to the baseline technology

Operational support tools and utilities (*e.g.*, ontology generation and development, data migration, system administration) are also being developed independently by Sandpiper Software to support the commercial information broker product.

Next Up Previous Contents
Next: 3 The RASSP Authorization Model **Up:** Appnotes Index **Previous:** 1 Introduction

Reuse Methodology and Implementation Appnote

3.0 Design Reuse Methodology

The methodology developed for design reuse in the RASSP environment is composed of two areas:

1. design-for-reuse
2. design-with-reuse

Generic processes were developed and modeled using the IDEF3 methodology for process modeling [AL_1992]. These models provide a foundation which organizations can customize and extend to implement specific reuse strategies. The process models have been captured in the "Domain Independent Reuse Methodology" document [Lockheed Martin_1996a]. In this section we provide a summary of the methodology and provide guidance on how the underlying processes may be adapted to meet a particular organization's needs. Additional information on the process modeling methodology is provided in the Process Modeling Application Note

3.1 Design-For-Reuse

The *Design-For-Reuse* methodology supports the identification of reusable design knowledge and related business and engineering information as well as its integration with the local reuse repository for enterprise-wide access. The major steps include:

1. Identification of a desgin object for inclusion in the reuse repository based on criteria established by the organization
2. Submittal and approval of the design object by a peer review board (*i.e* ., for quality, utility and generality assessment)
3. Evaluation of the design object by a local reuse administrator for classification and collection of meta-data
4. Integration of the design knowledge with the repository, and release for general access
5. If an appropriate class is not found, process-driven steps are taken to create a new class or modify existing classes in order to accommodate the new object

3.2 Design-With-Reuse

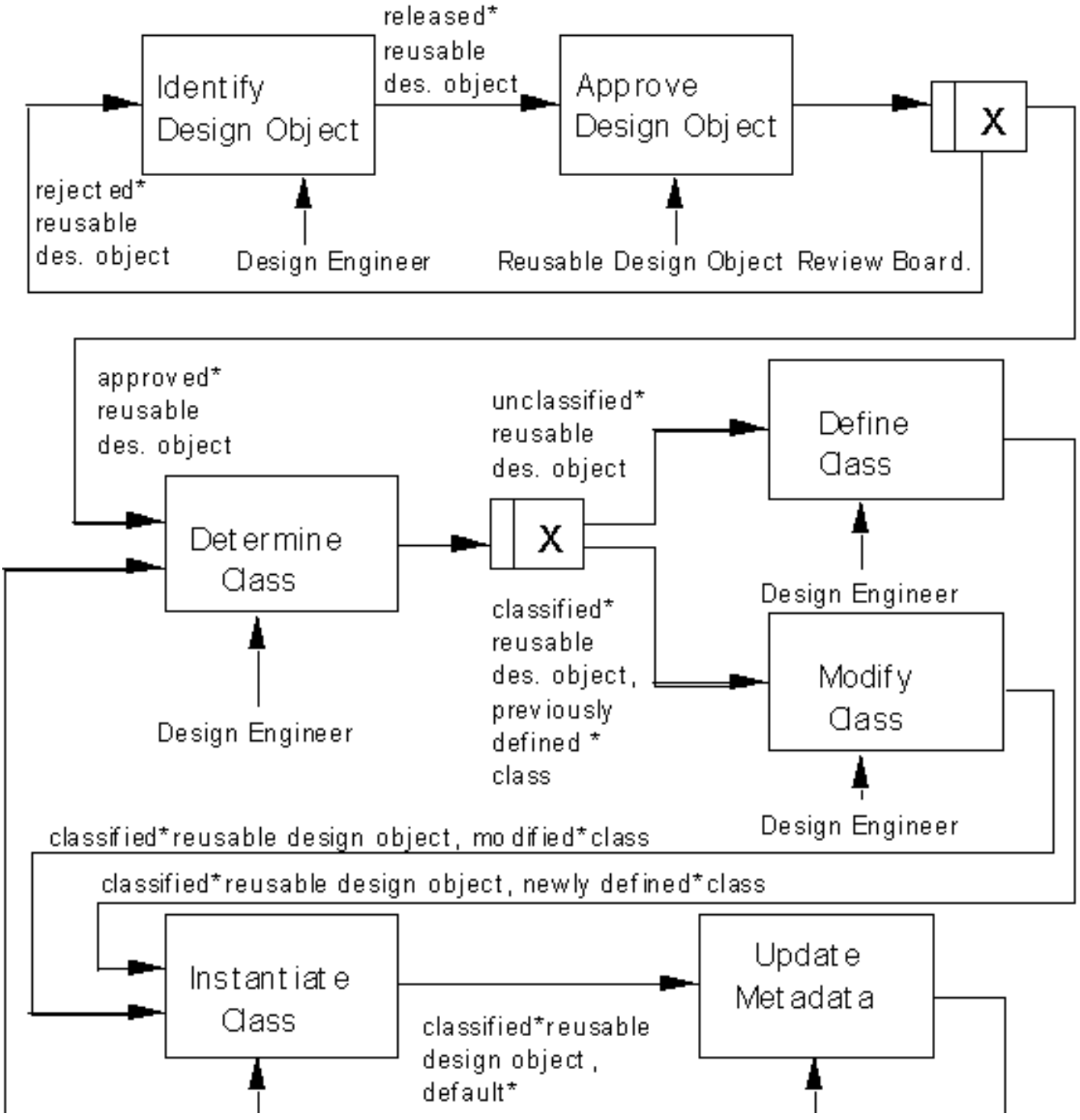
The RASSP program goal for *Design-With-Reuse* is to facilitate maximum integration and use of existing design knowledge where feasible. The steps include:

1. search for and select candidate reusable designs, based on form, fit, and function
2. copy the candidate designs into the designer's local workspace
3. integrate and test the various candidates to select the optimal design for the requirements
4. make any adjustments required to the selected design as needed
5. reintegrate and test the new design

The methodology document provides additional detail for each of the processes highlighted above. Adaptation of these processes may require integration and use of corporate standard taxonomies and engineering process improvement guidelines within the common vocabulary, the addition of corporate and domain-specific knowledge to the templates and methods to reflect cost, reliability, manufacturability, and other process information, for example. Customization of the generic Design-With-Reuse process may also include performing tradeoffs between designing from scratch versus modifying the selected design object, the addition of lessons-learned knowledge to the reuse

repository, and so forth.

The default workflow for addition of reusable design objects to the RRDM is shown in figure 3 - 1 and represented in IDEF3 (Integrated computer aided manufacturing DEFinition) notation. The boxes represent individual activities in a process, and the links between the activities represent precedence relationships between activities. The links are annotated with information about the data that flows between two activities -- the state of the data and the type of data separated by a '*'. A junction box, represented by a box with a vertical line parallel to the left edge and an 'X' within the box, is used to model alternate paths within a workflow. The arrows coming into the bottom edge of an activity box indicate the mechanisms that are involved in the activity, usually the job classification of the individual(s) performing the activity. The workflow is implemented using the workflow manager of the RASSP enterprise framework and may be further customized by a RASSP administrator for a particular organization.



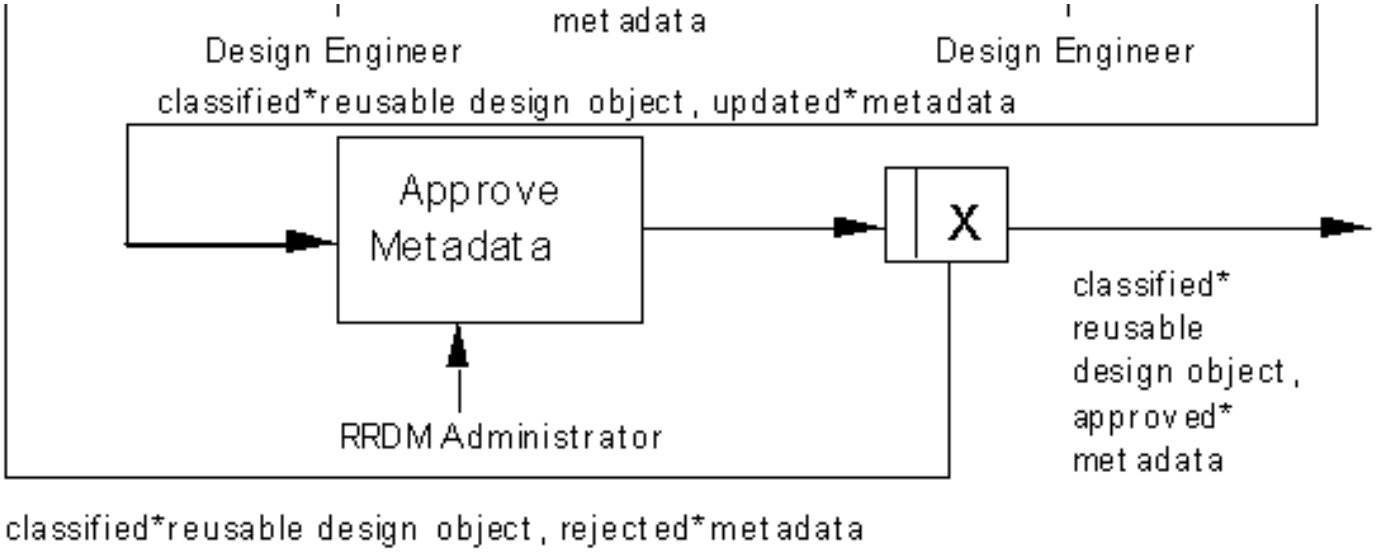


Figure 3 - 1: The workflow for integration of a new reusable design

---

 Dennis Basara

Reuse Methodology and Implementation Appnote

4.0 The RASSP Reuse Classification Hierarchy

The common vocabulary for design reuse used by the RASSP Reuse Data Manager (RRDM) is a knowledge-based representation of the domain, defining the organization of and relationships among the reuse elements. The ontology consists of two primary components -- a design object classification hierarchy, which models the functional, behavioral, structural, and performance characteristics of its constituent elements, and a data object classification hierarchy, which models the characteristics of individual drawings, source code files, documents, images, and other information assets.

A design object may be a system, subsystem, or component of a system that performs a real world function. Examples of design objects from the RASSP environment include analog-to-digital converter, baseband channelizer, network controller, Fast Fourier Transform (FFT) algorithm, and synthetic aperture radar (SAR) processor. A data object is essentially a document, either paper or electronic, that describes at least one facet of one or more design(s), related projects, or research. Examples of data objects include simulation model, schematic, test vector, and specification. Typically, a design object will be described by a number of data objects, and may refer to additional information assets that identify sources for algorithms, specific technology descriptions, and so forth.

The RASSP Reuse Classification Hierarchy (RRCH) was initially developed using Rumbaugh's Object Modeling Technique (OMT). This preliminary version of the RASSP Reuse Classification Hierarchy (RRCH) is documented in [Lockheed Martin_1996b]. On review of the integration requirements for the program, however, it became clear that the computer aided software engineering (CASE) tools available in the marketplace, including OMT, were inadequate to model the distinctions among the diverse, complex sources of engineering data in the RASSP environment.

Two key issues led us to move from a traditional CASE-based modeling approach to a knowledge representation approach:

1. a requirement to capture the semantic and syntactic differences among terms -- to model the constraints on attributes in a formal way (e.g., numeric precision, translation of units of measure, handling issues such as global pricing that require input from multiple external systems on demand)
2. the need to resolve communications issues among organizations with distinct cultural heritage (i.e., organizations that have been brought together as a result of mergers and acquisitions that have differing engineering practices and use distinct terminology to refer to the same or similar practices and data).

Ontolingua [KSL_ 1996a] was selected as the appropriate representation environment from several knowledge representation approaches due to the level of formality that it provided, accessibility of the tool environment on the Stanford web site, and the availability of support from key Stanford Knowledge Systems Laboratory (KSL) researchers.

An important development goal for the RASSP Reuse Classification Hierarchy was that it should be general enough to be adopted, ultimately, as an industry standard, either in part, or in its entirety. Thus, it should lend itself to extension without requiring destructive changes. Destructive changes include deletion of classes in the hierarchy, deletion of attributes, relationships or constraints, or moving classes within the hierarchy. Adding

new classes to the hierarchy and adding attributes, relationships and constraints to existing classes should be allowed. These restrictions enable upward compatibility with previous releases (i.e., previous integration of CAD tool libraries with the RRCH will continue to be valid). The International Electrotechnical Commission Draft International Standard 1360-1 for classifying electric components [IEC_1994] was adopted as a baseline for quantitative and qualitative attribute definitions as an initial step towards achieving this goal. Other standards for domain-specific information have been integrated throughout the modeling process.

The salient features of the ontology development methodology for an individual class of design objects are as follows:

- select the type of reuse data to model
- source standard taxonomies from the various standards organizations
- source data from vendors, government and commercially available libraries, the RASSP Enterprise team
- determine the set of potential attributes based on RASSP requirements, user input, analysis of products output by the CAD tools, the standards sourced
- specify the semantics and legal values for each attribute
- review the draft data dictionary and related ontology with local experts in the domain as well as with Stanford KSL modeling experts
- publish the data dictionary for comment within the program
- demonstrate the Ontolingua implementation to solicit feedback
- iterate through the process until consensus is reached

For all classes defined in the RRCH, a certain number of common attributes are defined. These attributes include information required to identify an individual information asset that may be distributed in an enterprise, regardless of where it resides or what its functional characteristics are. The baseline used for this purpose is the Global Information Locator Service standard, developed by the National Archives and US Geological Survey based on the ISO 10163 (ANSI Z39.50) standard, which specifies how electronic searches should be expressed and how results are returned to enable international access to diverse, distributed data [OIW/SIG-LA_1997].

This standard has been extended for use in the RASSP environment to include information required to launch CAD tools or access distributed databases, for example. Additional requirements were derived from the latest release of the Meta-data Interchange Standard developed by the Meta-data Coalition [MDIS, 1997], a consortium of data warehousing vendors. Other standards for the representation of specific types of information were incorporated as appropriate. The current version of the ontology that models the Global Information Locator Service is called Network-Based-Information-Asset, and is available on-line through the Stanford Knowledge Systems Laboratory (KSL) Ontolingua server at http://www.ksl.stanford.edu.

Figures 4 - 1 and 4 - 2, below, show subsets of the current Space-Systems design object and Engineering-Data data object hierarchies, respectively. The RRCH models the common vocabulary for the domain, including descriptive meta-data for the design and data objects that may be created or used within the RASSP environment. The interior nodes of the hierarchy are abstract classes, and leaf nodes are concrete classes that may be populated with real objects.

Figure 4 - 1: Communications-Function subset of the RASSP Space-Systems Ontology

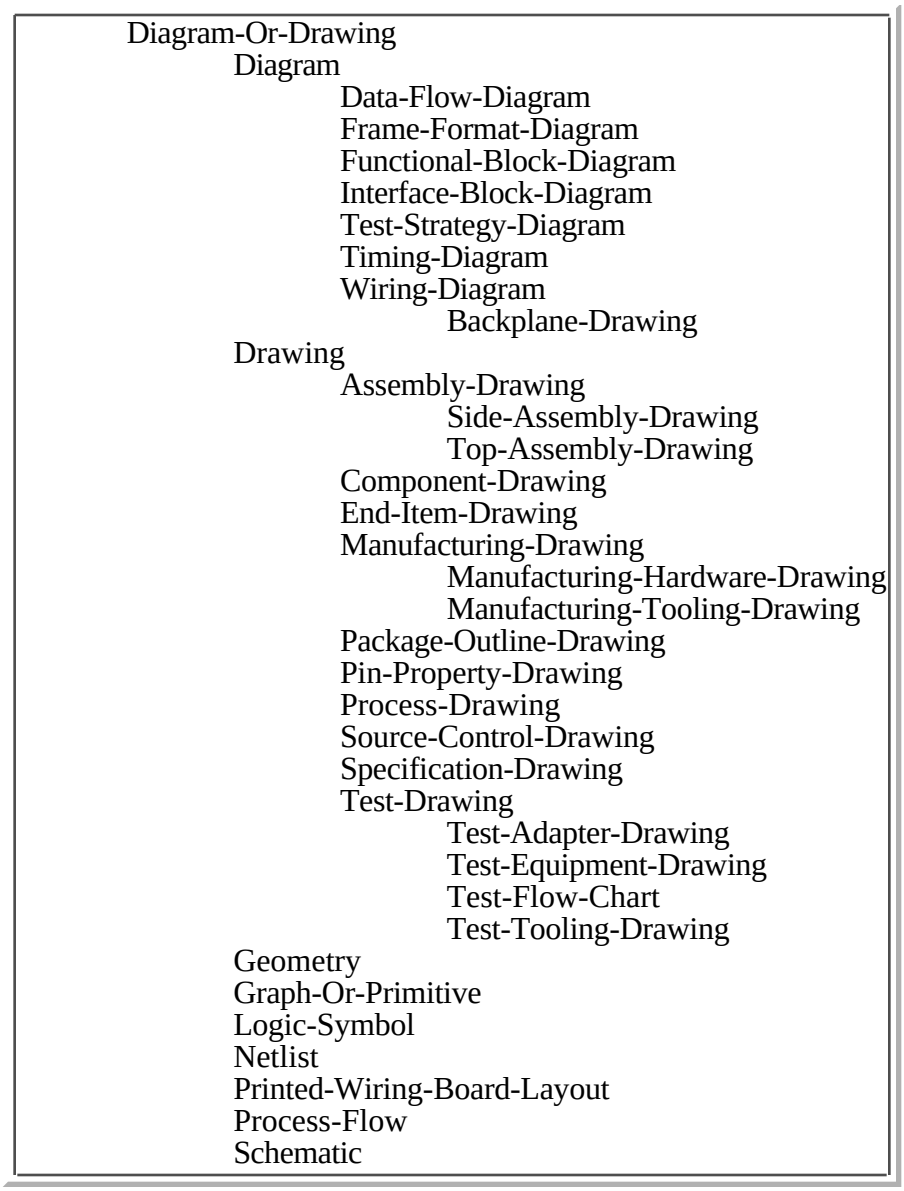


Figure 4 - 2: Diagram-Or-Drawing subset of the RASSP Engineering-Data Ontology

Additionally, individual classes in the Engineering-Data and Space-Systems ontologies include attributes that are specific to the domain. Examples of these kinds of attributes include size, weight, and power characteristics, performance characteristics, and so forth, depending on the asset type. An example class definition, extending the baseline information asset attributes to include those relevant to a Simulation Model is given in Tables 4 - 1 through 4 - 8. Many of the attributes described in the figure are inherited from the Network-Based-Information-Asset ontology. Those that are specific to the Engineering-Model class or its children, including the Simulation-Model class, are marked with an asterisk.

Note that a Data Type may be a primitive type, such as an integer, real-number, pointer (reference) or string, or may be composed of other definitions. Data types may be defined recursively, and may include semantic as well as syntactic information. The data types shown in the example are relatively simple, however, as our intent here is to show both the use of the Global Information Locator Service definitions and to highlight the Simulation Model class.

Field Name	Data Type	Description
Abstract	structure	Specifies information relating to the general nature and scope of the asset
Asset-Creation-Date	Time-Point	Date and time of asset creation
Date-of-Latest-Revision	Time-Point	Date and time of most recent asset revision
Version	string	Revision or Version number for the asset
Brief-Description-of-Asset	string	Brief narrative description providing sufficient detail about the asset to identify its general nature in summary presentations to users
Long-Description-of-Asset	string	Narrative description of asset relating to its general nature and scope; description may include a discussion of the content (e.g., data coverage, persons, events, and topics), time span, and geographic coverage if relevant (500 words or less)
Access-Constraints	structure	References any known accessconstraints for this asset
Legal-Access-Restrictions	string	Specifies any legal restrictions that may limit the user's right to examine material, such as proprietary or classified information
Physical-Access-Limitations	string	Specifies any physical access limitations, such as off-line archival

Table 4 - 1: Attribute definitions for the Simulation Model class

---

Field Name	Data Type	Description
Additional-Documentation	list of refs.	Specifies any additional documentation that contributes to the identification, selection, and manipulation of the information, in particular for documentation related to the use of an automated information system
Agency-Program	reference	References the project or program(s) for which the asset was originally developed
Availability-Of	list of refs.	References information regarding the availability of the asset from a particular distributor
Control-Identifier	string	Specifies a unique identifying number for each information asset, consisting of an acronym followed by the control number
Controlled-Vocabulary	structure	References specific controlled vocabulary elements (such as keywords) that can assist in automated searching when a large number of assets are available
Index-Terms-Controlled	list	Specifies a grouping of descriptive terms drawn from a controlled vocabulary source to aid users in locating entries of potential interest
Controlled-Term	string	Identifies multiple topics within a given controlled vocabulary
Thesaurus	reference	Provides a reference to a formally registered thesaurus or similar authoritative source of the controlled index terms (e.g., the RASSP VHDL Model Taxonomy)
Cross-Reference	list of refs.	Provides for the description of related information resources, links to additional Thesauri, etc.

Table 4 - 2: Attribute definitions for the Simulation Model class

Field Name	Data Type	Description
Date-Of-Last-Modification	Calendar-Date	Specifies the date that the meta-data entry for a particular information asset in the knowledge base was last modified
Local-Subject-Index	list of strings	Augments information provided in thesauri, or can be used in the absence of an acceptable thesaurus
Methodology	list of structs	Identifies any specialized tools, techniques, or methodology used to produce an information asset
Methodology-Description	string	Identifies a particular methodology used, through a textual description
Methodology-Documentation	list of refs.	Reference to one or more documents that pertain to the methodology used
Model-Abstract*	structure	Provides a means to categorize models according to a set of attributes that are useful in distinguishing models intended for distinctly different purposes
External-Resolution	structure	Describes how a model describes the interface of the modeled device to other devices
Temporal-Resolution	string	Represents the resolution of events that are modeled in terms of their time scale
Data-Resolution	string	Defines the resolution with which the format of values are specified
Functional-Resolution	string	Refers to the level of detail with which a model describes the functionality of a component or system
Structural-Resolution	string	Refers to the level of detail a model provides about how the modeled component is constructed out of constituent parts

Table 4 - 3: Attribute definitions for the Simulation Model class

Field Name	Data Type	Description
Internal-Resolution	structure	References how a model describes the timing of events, functions, values, and structures that are contained within the boundaries of the modeled device
Temporal-Resolution	string	Represents the resolution of events that are modeled in terms of their time scale
Data-Resolution	string	Defines the resolution with which the format of values are specified
Functional-Resolution	string	Refers to the level of detail with which a model describes the functionality of a component or system
Structural-Resolution	string	Refers to the level of detail a model provides about how the modeled component is constructed out of constituent parts
Model-Class*	string	Describes the class of model as an enumerated string from the following list: Behavioral Model, Functional Model, Structural Model, Performance Model, Interface Model, Mixed-Level Model, Virtual Prototype
Model-Maturity*	structure	States the maturity level of the model in terms of the engineering design information available to work with, test-bench availability, and so forth
Source-Maturity	string	Specifies the level of maturity of the source code for the model as an enumerated string: "Exists - completely models all capabilities", "Exists - partially models some capabilities", "Source-code" available / may be licensed / not-available, "Proposed / needed"

Table 4 - 4: Attribute definitions for the Simulation Model class

Field Name	Data Type	Description
Documentation-Level	string	Specifies the level of maturity of the documentation: "Complete user documentation available", "Design intent available", "Diagrams Only", "No documentation available"
Test-Data-Maturity	string	Specifies the test-bench maturity level, as follows: "Full test-bench exists", "Stimulus (test vectors) exists", "Expected outcome exists (test results)", "None"
Validation-Level	string	Specifies the level to which a model has been validated: "Compiles", "Runs", "Tested", "Verified", "Validated", "Mature / In Use"
Reusability-Level	string	Specifies the level of generality of the model from a reusability perspective: "Application-specific, single purpose", "Somewhat reusable - some functions generalized", "General-purpose"
Support-Level	string	Specifies the level of support by the originators: "Unsupported", "Partially supported", "Fully supported"
Model-Year-Architecture-Support*	structure	Describes the extent to which the model supports the RASSP Model- Year Architecture (MYA) methodology
Type-Of-Encapsulation	string	Specifies the type of MYA encapsulation (i.e., SVI, simple RNI, complex RNI)
Maturity-Of-Encapsulation	string	Provides an indication of the level of maturity of the encapsulation (i.e., Model Text Exists, Interfaced, Synthesized, Targeted)

Table 4 - 5: Attribute definitions for the Simulation Model class

Field Name	Data Type	Description
Original-Control-Identifier	string	Provides a means through which users can determine that while the descriptions of two assets may differ, one is a derivative of the other
Originator	reference	References the organization or agency that created and maintains the information asset
Point-of-Contact-For-Further-Information	reference	Identifies an organization or individual where appropriate responsible for the content of the information asset
Programming-Characteristics*	structure	Describes general programming characteristics of a model
Programming-Level	string	Describes the Programming Level for the model as an enumerated string, including: Object-Code, Micro- Code, Assembly-Code, High Level Language (HLL) Statements, DSP Primitive Block-Oriented, Major Modes
Programming-Language	string	Describes the language in which the model is written (e.g., C, PCL, various graphing languages)
Is-Synthesizable	string	Indicates whether or not the model is synthesizable (yes/no)
Execution-Scripts-Available	string	Specifies whether or not execution scripts ("run files") are available for the model (yes/no)
Standards-Compliance	string	Specifies the modeling language compliance, and to which standard (e.g., Verilog, VHDL-93)
Test-bench-Available	string	Specifies whether or not a test bench is available for the model (yes/no)
Test-Vectors-Available	string	Specifies whether or not test vectors are available for the model (yes/no)

Table 4 - 6: Attribute definitions for the Simulation Model class

Reuse Methodology and Implementation Appnote

5.0 The RASSP Reuse Data Manager Architecture

As described above, sources of reusable designs may include those created within an engineering design organization, CAD tool libraries, CAD tool-independent libraries, released designs managed by product data management (PDM) systems across the enterprise, related business and engineering information, and component vendor data, among others. To query and access these heterogeneous sources of design data, the RRDM provides designers with a common view of the virtual reuse repository. Syntactic and semantic differences among the various source file systems, libraries and repositories are resolved through a common vocabulary representing their union and mapping algorithms that relate the source vocabularies to the common view. The underlying knowledge engine that manages the common vocabulary, maps that vocabulary to the various distributed sources of reusable engineering design data in the environment, and provides a common user interface / single entry point from which users can search for, view and access candidate designs and related information is known as the Sandpiper Software Intelligent Information Broker (IIB). This section details the high-level architecture and concept of operations of the IIB as applied in the RASSP environment.

The environment in which the RRDM operates is illustrated in Figure 5 - 1, and consists of the following components:

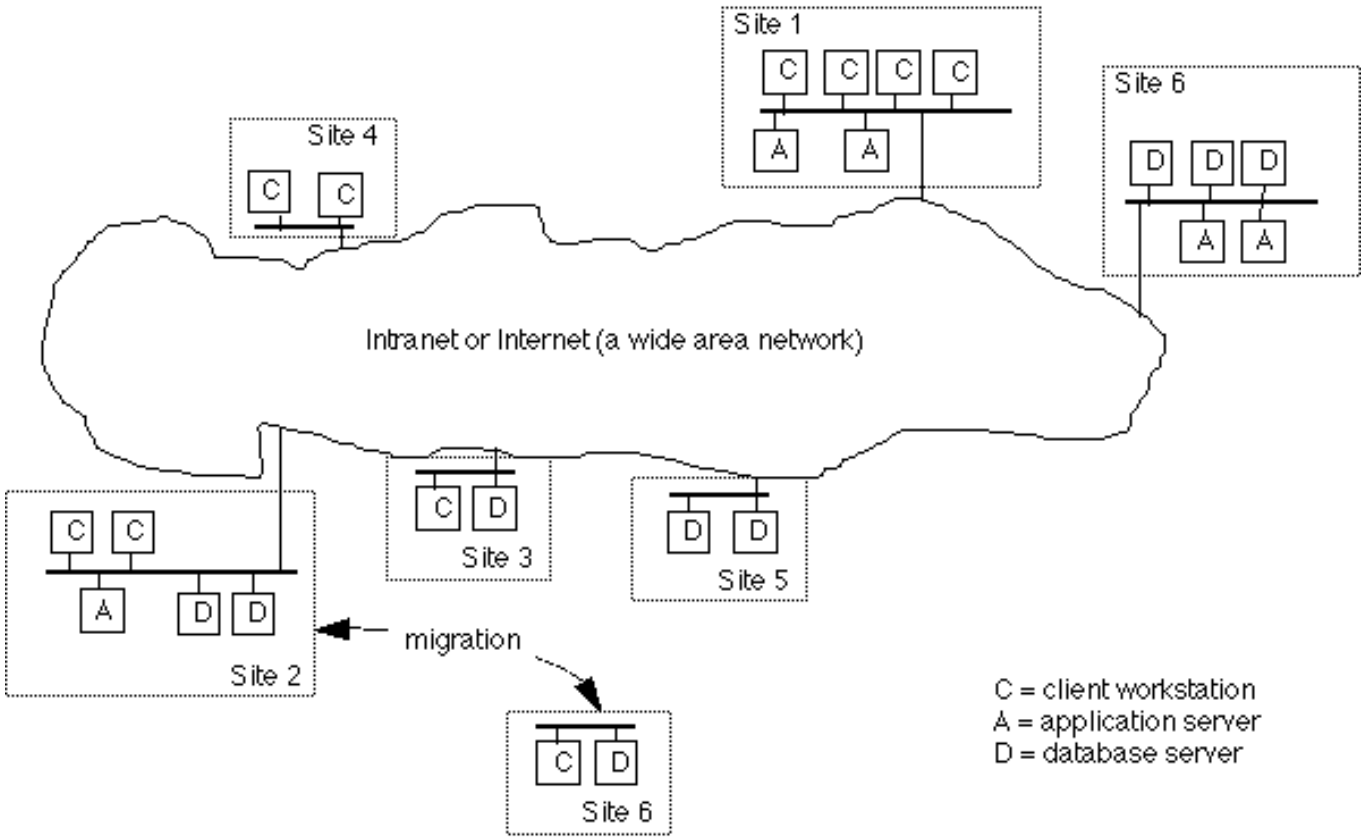


Figure 5 - 1: RASSP Reuse Data Manager Environment

- one or more sites, where users and/or source databases, applications, file systems, or other source information resides, connected by local and/or wide area networks
- multiple client systems and one or more back-end database servers which contain hardware and software from a variety of vendors
- back-end databases of varying content, organizations (i.e., schemas) and underlying DBMS engines (relational, object-relational, object), related to one another in both tightly-coupled (i.e., with similar schemas) and loosely-coupled (potentially having little or no similarity among them) federations

End users include, among others, designers who interact with the system directly or through integrated applications such as product data management (PDM) systems, workflow managers, CAD tools, or other systems in their environment.

The intelligent information broker (IIB) provides the capability for users to search for and retrieve design knowledge (or other related engineering and business information) stored in their distributed environment through the common vocabulary. This vocabulary is implemented as a group of internal repositories of descriptive information that describe the terms defined for the domain, the locations and characteristics of source repositories, characteristics of the mapping from source repositories to the common vocabulary, user authentication information, and so forth. The IIB provides the capability to search stored meta-data for design knowledge, regardless of whether the objects themselves are managed by the broker or externally.

The information broker (shown as an "application server") acts as middleware, integrating end users (or "clients") with the various sources of design knowledge in their virtual environment. IIB functionality has been partitioned between the application server, Java[a] applets integrated with the user's Web Browser, and the source database servers that comprise the cooperating environment. Integration of external applications is enabled through Common Object Request Broker Architecture (CORBA) application programming interfaces (APIs).

Figure 5 - 2 highlights the prototype IIB architecture as implemented in the RASSP environment. As mentioned above, users interact with the system either directly through a Java[a]-enabled web browser or through other applications that have been integrated with the IIB through CORBA-compliant APIs. As stated in the Architecture of the RASSP Enterprise System, section 2, a number of enhancements are planned over the course of the next 18 to 24 months that include commercialization of the broker and development of ancillary tools to support its implementation in various environments.

At the heart of the IIB is the Perspectives Server, which provides the intelligent search and information brokering function in support of user requests. From these user requests, it generates queries to the associated source repositories. The Perspectives Server maps the user's query from either the common vocabulary or a customized, user-specific vocabulary to the physical back-end repository implementation notation (e.g., SQL, C++), returning the requested data to the user in a user-specified format.

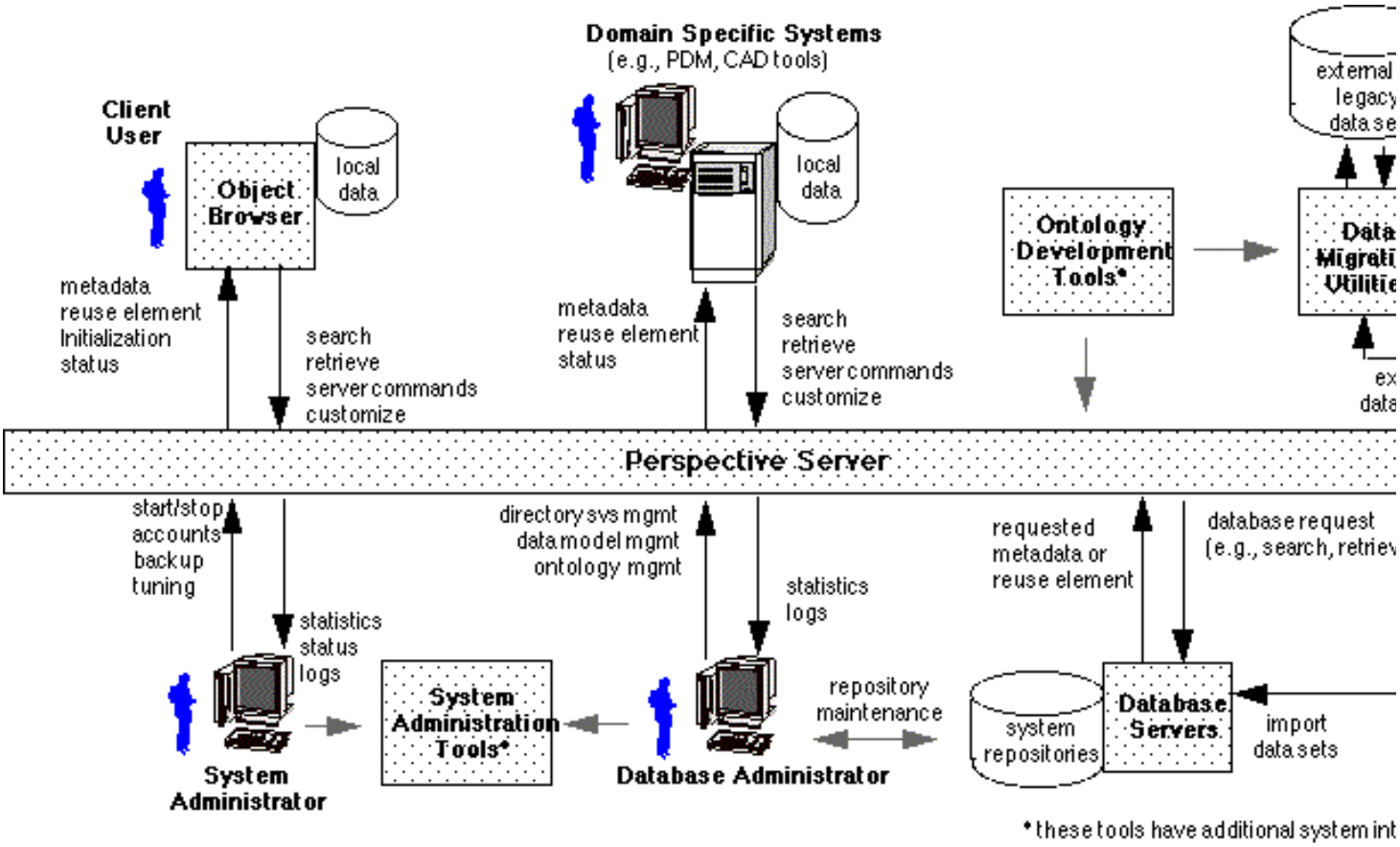


Figure 5 - 2: Sandpiper Software Intelligent Information Broker Context

The IIB provides native support for sets of tightly-coupled back-end database servers. These tightly-coupled federations have similar schemas and may be mapped to one another and the common domain vocabulary through a single source vocabulary. For source repositories that are sufficiently dissimilar, separate (distinct) source vocabularies will be integrated with the IIB. A single federation of source repositories may be represented by multiple vocabularies, each providing different views (e.g., by a standard, by taxonomy, by function, by product) of the underlying data. User selection between these views enables access to the data in a

manner most appropriate to the task at hand.

The Perspectives Server also describes the mapping between the users' preferred view and the common vocabulary for a domain. The user's custom view may include attributes from multiple classes, constraints on display characteristics, such as preferred currency or units of measure, and so forth. These views are also encoded as ontologies based on Ontolingua and KIF.

Reuse Methodology and Implementation Appnote

6.0 Application of the RASSP Reuse Approach

In order to demonstrate and benchmark the process, the RASSP Reuse Data Manager (RRDM) was implemented for a Satellite Communications Payload application (interim prototype demonstration), and for distributed access to multiple VHDL model repositories (final prototype demonstration). As a result of designing these prototypes, the RASSP reuse classification hierarchy was developed for the respective domains. Extension of the classification hierarchy either in these domains or related areas is planned with additional applications of the reuse manager.

6.1 Satellite Communication Reuse Application

The Satellite Communication Payload application was associated with an internal project at Lockheed Martin Astrospace Company at East Windsor, New Jersey. Our goal was to build a design reuse repository from the legacy data sources at Astrospace, and maintain it over the course of the demonstration period. Two incremental prototype RRDMs demonstrated some of the basic features of the Information Broker in the Astrospace environment, including the object-oriented class hierarchy, attribute-based searches, and web-based access.

The CPC was in a position to provide objects in a wide variety of functional categories which could be used both to augment the RASSP RDOCH and upon which to implement the demonstrations. The metadata about these objects promised to be rich in terms of the variety of naming conventions, data values/enumerations, units, data types, data formats and such. This was considered to be very important from the interoperability perspective. The data also demonstrated the concept of representing multiple implementation levels (e.g., concepts, designs and implementations from components to systems). Use of this diverse data set showcased important ways of accessing information to solve design problems at multiple levels like satellite beam coverage analysis down to communications circuit design. In addition, the ability to access information on concepts, trade studies, rules of thumb and detailed design from distributed sources is extremely useful. This showed how an organization can enhance the engineering process by encouraging reuse.

The CPC problem was a real world case for an integration of interest to Lockheed Martin, the Government and aerospace in general. The CPC had developed potentially reusable components in the past and was about to embark on many more to support the new generation of digital communication satellites. CPC was developing a "configure-to-order" to replace the traditional "design-to-order" concept for satellites. "Configure-to-order" required CPC to maximize the use of reusable components, assemblies and even systems. It required the ability to catalog and easily access information residing in a wide variety of sources including databases, product data managers, component information systems and document management systems. The system would have a wide range of users (e.g., systems engineers, designers, procurement specialists) each requiring a different "view" of the multiple data sources. This demonstration showed the proof-of-concept for providing these views and highlighted the types of features that would be necessary to integrate distributed heterogeneous sources for a engineering reuse application.

The process of building the reuse repository consisted of the following steps:

1. Interview design engineers at Astrospace to identify sources of legacy reusable design data.
2. Source the information, which consisted primarily of presentations, drawings, and some

documentation from legacy sources and on-going projects
3. Develop a class hierarchy for the data available in a bottom-up fashion, identifying leaf-level classes and their attributes first, and abstracting common facets to form abstract superclasses
4. Populate the data in a repository with a web-based front end

The first interim implementation of the RRDM was created with classes and instances stored as static HTML pages. The documents were obtained either in electronic format or were scanned into GIF and PDF formats from paper documents. HTML links were provided from the design object class instances to the data objects (documents) that described the design objects. Documents could be viewed or downloaded as either HTML files, Adobe PDF files, or in their original format -- typically Microsoft Word or Interleaf format. Security was provided using the Netscape Secure Socket Layer (SSL) features. Only people accessing the system from authorized machines (verified by IP addresses), with an authorized user name and password were allowed access to the system.

This interim implementation of the RRDM lacked scalability, as the only means for searching for data was by browsing or doing text searches on the HTML pages. Sophisticated searches using attributes of the classes were not possible. To overcome these limitations, an object-oriented approach was used for the second RRDM prototype engine. An object-oriented database system was selected over a relational database for the following reasons:

1. The object-oriented reuse classification hierarchy could be mapped directly to C++ classes for implementation purposes, rather than simulating each class as several tables in a relational or object-relational environment (cumbersome and costly).
2. An object-oriented database system uses pointers to represent relationships between objects rather than object-ids and multiple table joins as in an object-relational DBMS. The RRCH implements numerous relationships between classes and between instances of classes. An object model seemed to be most appropriate from a performance perspective.
3. The object-oriented model is more expressive than other modeling methodologies such as relational, network, or hierarchical. It allows natural modeling of complex data types such as sets, arrays, and collections, and facilitates the definition of the behavior of the objects using methods derived from the constraints specified in Ontolingua or defined in source applications and repositories.

The second prototype for the satellite communications RRDM was implemented using ObjectStore, an object-oriented database management system with a C++ -based API. The class hierarchy was implemented and maintained through a C++ -based programmatic interface. The database was populated manually through a graphical user interface and through scripts with load files once the schema became stable. Complex attribute-based querying was supported in this implementation. Inverse pointers between design objects and the related data objects were implemented to maintain database integrity. Thus when new design objects were added or when a design object was deleted from the repository, the back pointers from the related data objects were automatically updated by the ObjectStore system.

Key lessons learned in the course of this effort are:

- **Guidance for engineers is needed to clarify the definition of a reuse item and improve the reuse library population process**
    - A considerable amount of data generated during the course of a design that is not normally considered reuse data should be, either on its own merit, or becuase it enhances the usefulness of other reuse items. This includes such things as descriptions of design intent, trade-offs performed, and candidate design selection criteria. Rejected approaches and the reasons for rejection are almost as important to a document describing the selected approach as the selected approach itself. Giving designers access to the thought processes of the reuse item creators enhances their ability to make good decisions about using the item. Although this type of data exists, because it is not considered part of a traditional data package, it is often hidden in reports and presentations, or worse, in somebody's notebook. Designers must be constantly aware of the reuse process and what types of data objects are of interest from a reuse point-of-view so that appropriate data is collected, described and stored over the

course of the design cycle.

- **The process for collecting and documenting reuse data must occur in parallel with the design process**
  - Collecting data at the end of a project is a time-consuming and slow task. Once a project is over and people are assigned to new tasks, it is very difficult to recreate quality reuse data. A significant amount of data will be lost if the engineers are not consciously and continuously collecting what is needed to create a robust reuse object. Each organization must define what documents are important to the reuse community, basing their decisions on both the type of information normally generated during the course of a design, and the type of additional information that must be collected to support a reuse library.

- **Domain experts must spend some time learning the classification process in order to be effective in their role in defining the class hierarchy and class attributes**
  - For example, hardware engineers tend to classify objects on the basis of a physical hierarchy, instead of a functional hierarchy. Thus, a typical hardware hierarchy might be: System => Box => Board => Chip. However, from a functional point of view, "System" , "Box", "Board", and "Chip" have similar attributes: function, throughput, latency, power, etc. From a structural point of view, they also have similar attributes: size, weight, number of I/O, etc. As such, they would not belong in separate classes. Rather, "System", "Board", etc. would be a value for an "Implemented-as" attribute of the Design Object class. Familiarity with the classification concepts, along with examples, will assist the domain expert in defining a quality class hierarchy.

6.2 Distributed VHDL Model Repository

The final reuse prototype demonstration featured the integration of two distributed VHDL model repositories. It demonstrated support for uniform user access to source repositories with diverse implementation schema via the information broker.

The focus of the final demonstration was two-fold:

- to highlight the productivity gains realizable with the Sandpiper information broker as compared to conventional approaches or current best practices for queries against multiple heterogeneous databases, and
- to show the generality of the solution and its relevance to other government and commercial applications.

Two databases on separate machines were used to represent a distributed database. Both databases were searched with a single query from the Sandpiper information broker. Corresponding searches were performed on the individual databases using native search methods to verify the correctness of the results.

The first database was developed from Lockheed Martin Advanced Technology Laboratories-developed VHDL models primarily associated with the RASSP benchmark program and other internal libraries. These model libraries include performance and detailed behavior models for the UYS - 2 and SAR benchmark programs, and register transfer level (RTL) models from the RASSP Model Year Architecture effort. Metadata about these models is accessible through a keyword search engine developed by ATL. (The vocabulary for the keyword search engine was significantly enhanced as a side benefit of this effort.)

The second database was developed using VHDL models associated with the RASSP legacy information project managed by SCRA. These models are available through an HTML-link-based search engine developed by SCRA.

The class and attribute definitions for the VHDL data and design objects were developed based on the GILS ontology, the RASSP VHDL Modeling Terminology and Taxonomy document, and the bus attributes table developed under the Model Year Architecture effort. Additional attributes were derived from common

datasheet usage.

Productivity gains with the Sandpiper information broker as compared to conventional approaches or current best practices for queries against multiple heterogeneous databases were realized in two areas.

- **Resolution of ambiguities in terms**
    - Search results improved for searches using attributes that had the same name, but different meanings. This is an area where it was expected that the Information Broker would excel: a direct result of the more rigorous modeling provided by the RASSP Reuse Classification Hierarchy.
- **Improved query quality for reduced search time**
    - A significant improvement was realized in the quality of the search that could be specified and the accessibility of the information.

An example of the problem with searching in a heterogeneous environment where a word can have multiple meanings clarifies the issue. The SCRA website has models of FPGAs. These are models of the FPGA themselves as opposed to the logic implemented in the FPGA. ATL uses the term FPGA to refer to the logic that is to be modeled and/or implemented in an FPGA. Using the native search engines of each database, one would find both models of FPGAs (on the SCRA site) and things implemented in FPGAs (on the ATL site). Because of a lack of semantics on "FPGA", there is no convenient way to distinguish the FPGA models from the FPGA implementations. Through the common vocabulary, the Information Broker allows the semantics of FPGA to be captured through mapping to the "function" class. The semantics of "FPGA" for ATL are mapped to the "implemented as" attribute. This difference enables Sandpiper to differentiate the two and provide accurate search results.

The quality and accessibility of the design objects found by the Information Broker is enhanced in two ways. First, when the recommended design-for-reuse approach is used, the information is complete and meta-data about the information significantly more detailed than is traditionally supplied. Being able to search and compare on the meta data and understanding the semantics of meta-data from multiple sources is the value added by the information broker. Second, a special "Info" file and the use of viewers allows the designer to do a quick preview of the model before downloading it. The designer spends less time downloading, unzipping and untarring unwanted models and more time searching for an appropriate model. Although this second aspect was not demonstrated, the information was captured to enable demonstration at a later time.

As previously mentioned, the RRCH is compliant with the RASSP VHDL Modeling Terminology and Taxonomy. The Virtual Socket Interface Alliance (VSIA) is adopting the RASSP VHDL Modeling Terminology and Taxonomy as the basis for its own taxonomy to describe intellectual property available from multiple vendors, and to act as a clearing house for that information via the Internet. They have also incorporated portions of the bus attributes table in their bus attributes document. The detailed ontology definition and corresponding meta-data templates developed for this prototype are relevant not only to VSIA but to numerous collaborative, concurrent engineering applications.

One lesson from the Satellite communications reuse application was reinforced and several new lessons were learned in the course of developing this demonstration.

- **The importance of collecting the information as the models are developed was reinforced**
    - This reduces the risk that portions of models will be missing or modified due to uncontrolled reuse. It also increases the retained design knowledge should a designer leave the organization. The quality of the information will be significantly higher if it is developed while it is fresh, and not recreated by someone else after the fact.
- **Clearly defined organization of data during the data mining/collection process is critical.**

- **Designate a reuse area and define its structure. The following suggestions may help:**
  - Figure out how you are going to organize the models; By program, By function/abstraction, By developer or By group. Draw a diagram of the hierarchy. At the bottom of the hierarchy you will have a directory for each model and its associated data objects.
  - Avoid the proliferation of nearly identical packages with the same name.

    We developed the following naming convention for packages. Put all the widely used packages at the top of the hierarchy. Any packages derived from these (copied and modified) should be renamed in some standard way and reside in the same directory as the model that they are used with. If a package is at the top of the hierarchy, no other package in the hierarchy should have that name. The names of packages specific to models which reside in a particular model directory should include the name and the source of the model. Since models are copied flat into the user's local space, this will reduce the chances of packages being overwritten.

- **Run all "extract" and "compile" scripts to verify proper execution.**
  - "Extract" scripts copy all of the files required (including packages) for model execution to a user directory. "Compile" scripts compile the files in the proper order. Ensuring that these are correct will simplify incorporation into the database, later.
  - Do not use hard paths in the "extract" or "compile" scripts.

- **References in the VHDL code to designer-created libraries need to be handled appropriately.**
  - In this environment, packages are copied into the user area and compiled into the WORK directory. Where necessary, references in the VHDL code to these packages were modified to reference the library WORK.

- **Provide the developers with templates defining the type of data that needs to be collected.**
  - Review the collection and documentation process periodically.

B. Database schema

- Clearly, if the models are already in a database, the schema is defined. However, if the legacy data is not in a database, this is a good opportunity to do a good job creating the schema. In particular, staying close to the common vocabulary will reduce the time it takes to integrate to the Information Broker.

C. Functional and domain specific schema

- Finally, when creating a functional schema, create it in the context of the world of engineering functions. That is, separate out general functionality such as arithmetic functions, filters and multiplexers, from the domain specific functionality. Look at the common vocabulary: chances are, the more general functionality has already been classified. The careful development of the domain specific schema will ensure good search results while minimizing the effort required to define the schema.

Next Up Previous Contents

Next: Conclusion **Up:** Appnotes Index

Reuse Methodology and Implementation Appnote

7.0 Conclusion

An application such as the RASSP Reuse Data Manager and its underlying Sandpiper Intelligent Information Broker (IIB) can save organizations large sums of money by simplifying and enabling knowledge integration tasks. The number and variety of potential applications for this technology is broad. Due to the level of complexity, a significant applied research investment will have been made prior to productization of the IIB, however. This kind of application also requires large amounts of test data and a number of test domains in order to validate the approach and functionality of the solution. We have successfully proven, though, that the broker technology is capable of resolving conflicts among terms, enabling integration across multiple, diverse sources of information, and supporting collaborative engineering and design reuse on the RASSP program.

The technology prototyped as a part of this effort has far reaching implications for the development of enterprise systems in the future, for the meaning of the terms open system and interoperability, and for any situation requiring integration of multiple applications, sources of business and technical knowledge, or a combination of the two. The application of knowledge representation concepts to problems of concurrent engineering, collaborative work, and integrated manufacturing is relatively new to the commercial community, though work has been funded in this and related cognitive science areas for a number of years. The problems related to information access and more importantly, decision support, in a collaborative engineering environment are difficult to solve due to the diversity and potential nature of the data and applications present in the environment. Issues involving information access across databases and applications in general are extremely complex, in part because each database instance or application is in all likelihood distinct, frequently poorly documented, and the individuals who developed them may no longer be available.

A significant portion of the work involved in establishing requirements for a system capable of enterprise-wide collaborative engineering and design reuse support, in soliciting feedback from the user community, in defining the ontologies and processes and in preparing the demonstrations described herein was accomplished essentially as an applied research and consulting task. This task required resources well beyond what was originally anticipated by the program, as is frequently the case with data warehouse and other database applications development. As the requirements became clearer and the task more and more well-defined, the work required to fulfill those requirements and our collective vision increased as well. The knowledge gained through this process, however, was also significant and will be applied to many other programs and opportunities in the future.

Reuse Methodology and Implementation Appnote

8 References

Armstrong Laboratory, "IDEF3 Process Description Capture Method Report", AL-TR-1992-0057, Wright Patterson Air Force Base, Ohio, 1992. [AL_92] (reference is not available)

American National Standards Institute, ANSI/NISO Z39.50-1995, Information Retrieval (Z39.50): Application Service Definition and Protocol Specification, 1995. [ANSI_95]

International Electrotechnical Commission, "Standard Data Element Types with Associated Classification Scheme for Electric Components -- Part I: Definitions, Principles and Methods," Draft International Standard 1360-1, Netherlands, September 1994. [IEC_1994] (reference is not available)

A. Farquhar, R. Fikes, J. Rice. "The Ontolingua Server: a Tool for Collaborative Ontology Construction". Proceedings of the 1996 Knowledge Acquisition Workshop (KAW96), KSL-TR-96-26, November 1996. [KSL_96a]

R. Fikes, A. Farquhar, W. Pratt. "Information Brokers for Gathering Information from Heterogeneous Information Sources", Proceedings of the Ninth Florida Artificial Intelligence Research Symposium (FLAIRS '96), John H. Stewman [ed], pp.192-197, Key West Florida, May 1996. [KSL_96b]

Lockheed Martin Advanced Technology Labs., "Domain Independent Reuse Methodology Development", Camden, New Jersey, 1996. [Lockheed Martin_96a]

Lockheed Martin Advanced Technology Labs., "The OMT Model of the RASSP Design Object Class Hierarchy", Camden, New Jersey, 1996. [Lockheed Martin_96b] (reference is not available)

Meta-data Coalition, Version 1.1 Meta-data Interchange Specification, August, 1997. Available on the Internet at http://www.he.net/~metadata/standards/toc.html [MDIS-11]

Open Systems Environment Implementors Workshop / Special Interest Group on GILS, "Application Profile for the Government Information Locator Service (GILS)", November, 1997. Available on the Internet at http://www.usgs.gov/gils/prof_v2.html. [OIW_SOG-LA_97]

"REUSE Methodology and Implementation Requirements Version 1.0 ", Lockheed Martin ATL, November 1995. [REUSE_RQTS_95]

SDRC, "Metaphase 2 -- Object Management Framework User's Manual", Milford, Ohio, 1997. [SDRC_97]

Sandpiper Software, Inc. and Stanford University Knowledge Systems Laboratory, "Application of Ontology-Based Knowledge Representation to Design Reuse", Saratoga, California, 1997. [SSI_97]

Appnotes

Enterprise Application Note
Process Modeling Application Note

Next Up Previous Contents
Next: Up: Appnotes Index **Previous:**7 Conclusion

Approved for Public Release; Distribution Unlimited *Dennis Basara*