5

Design History System: Data Models & Prototype Implementation

Jami J. Shah[1], S. Rangaswamy[1], Sohail Qureshi[1], Susan D. Urban[2]
[1]*Dept. of Mechanical & Aero. Engineering, Arizona State University, Tempe AZ85287, USA*
[2]*Dept. of Computer Science, Arizona State University, Tempe AZ85287, USA*

Key words: Design history, Product data management, Ontology

Abstract: This paper presents ISO/STEP compatible data models for digitally archiving the history of design projects. The two main components are a product core model and a process core model. The former relates product data to design process rationale and the latter encapsulates the design process itself. The process core model can represent process hierarchy, structure, logical and temporal relationships between processes and design steps. The core models are domain independent; specific schemas can be derived from the core models. A prototype Design History System has been implemented which can be accessed over the Internet with a web based user-interface. A relational-object database is used for design history archival. The system supports browsing, various types of queries, and dynamic schema evolution.

1. BACKGROUND

In today's large Corporations, product development teams are faced with many challenges. Under severe time and cost constraints, they must produce a design for a complex product that requires considerable resources to produce, often incorporates new technology and processes, and is expected to perform according to specifications with minimum cost and risk over its life cycle. George Santayana noted that "those who cannot remember the past are condemned to repeat it,"(Santayana 05). Thus, it makes sense for a

S. Finger et al. (eds.), *Knowledge Intensive Computer Aided Design*

design team to research similar past projects before and during product development. It is estimated that designers spend over 40% of their time looking for information. Some of the knowledge they require is archived in textbooks, journals, or company proprietary documents, etc. However, a large portion of it, comes from the company's past experience, and it is this knowledge that is often poorly documented..

Information that designers are interested in fall into three categories: physical attributes, performance characteristics, and design rationale. Since they are not trying to replicate the previous product, but to use it in designing a new one, they would like to understand not only the major hardware features, characteristics, and design parameters, but even more importantly, the rationale behind the choice of those features, characteristics, and parameters. In particular, they would like to see the analyses and design calculations that supported the choice of a particular set of characteristics.

In most large companies, product data is currently maintained in Product Data Management Systems (PDMs). Current PDMs are for the most part file-oriented and do not either store or reveal the semantics of the file contents. As a consequence, they provide a very meager means for relating product data with process or rationale. Also, PDMs do not capture much of the data they maintain. They obtain much of their data—e.g., manufacturing bills of materials (BOMs) and plans—by duplicating this data from other systems, where it is initially captured, and where it is also 'managed'. We need to go beyond PDM technology to create more comprehensive Design Information Systems (DIS) to support the work of design teams distributed in space and time. During the design process the DIS could provide support for better coordination of between team members. This would facilitate design change propagation throughout the project and provide an understanding in the future of the rationale used in design decisions. Before such information systems can be built we need to formulate data schemas that are rich, flexible, and extensible. This paper will discuss both the derivation of such a schema and the implementation of a prototype based on it.

2. LITERATURE REVIEW

Formal models for information representation have been developed in a fragmented manner. There is a set of ISO standards for representing (exchanging) detailed level geometric data and assembly structures (ISO 96). The standard is being extended gradually to include parametric geometry, constraints, construction history (Pratt 97) and some analysis data like FEA (ISO 96). There is no standard representation of performance parameters,

function and behavior, or design process and rationale. Stanford's ontology editor uses KQML to specify domains of discourse among intelligent agents in terms of definitions of shared vocabulary (classes, relations, functions, and object constants) from a knowledge engineering point of view. Experiments at Stanford were conducted to determine what information should be captured on the basis of relative frequency of use (Baya92).

There are several computer-aided engineering design systems based on the hypertext paradigm (Brown89, Fu90, McCall89, Thompson90). Design rationale is built up by creating documents and links based upon hypertext. Other systems have tried to capture the "design trace", which is an execution trace of the CAD tools (Casotto94). DEDAL/Weblibrarian provides the means for a system model based indexing that is used in searching through hypermedia documents but it does not present the product or other types of data in a structured way (Baudin 93). *PartNet* uses the WWW for access to component catalog data; it assumes data retrieval and interpretation by human users and lacks structured representation of associated design procedures.

One argumentation model is the Issue Based Information System (IBIS) developed by Rittel for organizing the deliberation process that occurs during complex decision-making (Rittel 73). An IBIS model views the decision-making process in terms of issues (tasks, questions or problems), alternatives (proposals or concepts), arguments (evaluations) and decisions. It is possible to find whether an issue has been resolved or not, what decisions have been made, and the reasons for them (Chen 91, Conklin 88, Lubars 91). An extension of IBIS which would make it more applicable to the description of mechanical engineering design processes was proposed in (Ullman 95) and (Nagy 92). Lee (Lee 96) has provided a framework DRL (design representation language) that extends IBIS by adding more concepts such as goals, questions, procedures and viewpoints. Somewhat similar to IBIS is the QOC model proposed by MacLean (MacLean 96) in which, the logic that leads to each decision is modeled in terms of Questions (key issues), Options (possible answers) and Criteria (for comparing options).

A number of hypertext systems were developed to support frameworks for argumentation. Such systems included PHIDIAS (McCall et al. 94) and gIBIS (Conklin, 88) for the IBIS framework as well as EUCLID (Smolensky et. al. 87) and SIBYL (Lee 90) for other argumentative frameworks. PHIDIAS combines computer-aided design, argumentation, and multimedia capabilities (McCall 94). There have been attempts to combine argumentative approaches to capture design rationale and other design aids to improve the design process. Another system is the KBDS (King 95) system in which, a knowledge-based design system (KBDS) was used along with IBIS to develop a tool to support chemical plant design using design

history. A system for conflict resolution in collaborative design called SHARED-DRIMS was proposed by (Pena-mora, 1996).

Case-based reasoning (CBR) is a general paradigm for problem solving based on the recall and reuse of specific past designs (cases). The three basic steps in a typical CBR cycle are: *find, retrieve, revise, and archive.* Many methods have been developed for each of these issues, such as *Archie* (Domeshek, 97), *Cadsyn* (Maher, 91), *Casecad* (Maher, 95), *Kritik* (Goel, 96), *Cadre* (Faltings, 97) and *Cadet* (Narasimhan, 97). *Casecad* and *Cadsyn* represent subcases with the Function Behavior Structure (FBS) framework which classifies the variables used to describe design cases into three semantically different categories. *Kritik*'s developers directly addressed the representation of causal behavior as sets of device states and transitions linking the states; these behaviors explain how the physical device's structural elements achieve its functions.

Commercial Product Data Management (PDM) systems do not support retrieval based on file contents and provide very limited support for design process representation. Current approaches for computer-based design retrieval fall well short of answering the types of questions that one human designer asks of another. The reason for this lack of sophistication is that design information is archived as either text/hypertext or data indexed in terms of a few pre-defined attributes.

Our previous ASU Design Information System (DIS) (Shah, 96) supported four information elements: product data, design steps, relationship of design steps to product data and rationale. The product data model is imported from STEP with modifications made to include design with features. The system was implemented on the World Wide Web and the data is stored in text files. This is a disadvantage because, complex data types cannot be handled and the system slows down as the text files get bigger. The work reported in this paper extends the ASU DIS system in several directions.

3. DIS CLASSIFICATION

In order to present the different research approaches to Design Information Systems under a common perspective, a taxonomy is presented here. The classification is based on six generic criteria:

1. Level of dynamic functionality: This criterion is a measure of both the extent of data processing and the control that the system has over the design process. At the lowest level is the archival retrieval of files like a PDM system which knows little about the contents. At the highest level is a pro-

active system that monitors conflicts, constrains violation, and automatically propagates design changes.

2. Level of abstraction: This criterion compares the levels of abstraction at which product data is captured. A system may support one or multiple levels. These levels may also be perceived as phases of the design process: the longer one gets into a project, the more information there is; for example, there is more detailed information available at the embodiment stage than at the early conceptual design stage.

3. Data organization: This criterion compares the level of formalization and structure of archived information. Unstructured text may require human interpretation while formal models may be thought of as computer understandable.

4. Granularity: Indication of the various levels of detail at which the design process information is captured, from microscopic actions, such as program log files to high level project planning information.

5. Data integration: This criterion refers to the data sharing architecture: centralized, distributed, or federated.

6. Schema flexibility: This categorizes information models according to the freedom they provide to the users to incorporate new applications via changes to the schema.

Table 1 is a comparison of five of the many design history systems described in the sections above: *PHIDIAS, DRIMS, C-DeSS, ASU-DHS* and *Cadre*. The Table shows that only passive design information systems have been built as prototypes. Most only capture a small fraction of design information and still lack formal structure. Also, systems either assume ownership of all data (centralize), or generate their own "overview" data, such as the IBIS approach. An integrated view of all design data in a distributed environment is done superficially today.

4. OVERVIEW OF RESEARCH

Except for highly routine designs, the product structure and design procedures are not known *a priori* ; this implies that the information schema evolves during the design process. Therefore, support for dynamic schema evolution and continual transformation of instance data between schema changes are fundamental requirements of any active or pro-active design archival system. A domain independent query language is another basic requirement for design retrieval. An indexing scheme is required to support user defined queries. Therefore, we need both topical models and integration infrastructure. Topical models are focused on specific aspects of design information (e.g., geometry, stress analysis, project management,

etc.), while the integration model is needed to bring together product and process topical models without eliminating the flexibility to extend and develop models in localized domains. Various standard and pseudo standard topical models already exist but integration models for relating such information, at various levels of abstraction, to process information do not exist. The need for data organization, query language, and schema evolution all point to the use of database technology in the implementation of design archival systems.

Table 1 Comparison of Design Information Systems

Classification criteria	PHIDIAS	DRIMS	C-Dess	ASU-DIS	Cadre
Dynamic functionality					
Static file retrieval/ Browsing	•	•	•	•	•
Unstructured search	•	•	•	•	•
Structured queries	-	•	-	•	-
Reactive: Input checks	-	-	-	-	-
Proactice: Change propagation	-	-	-	-	-
Abstraction level					
Perceived need	•	•	•	•	•
Design specifications	•	•	•	•	-
Functional design	•	•	•	•	•
Embodiment design	•	•	•	•	•
Artifact type design.	•	•	•	•	•
Artifact instance design	•	•	•	•	•
Manufacturing process data	-	-	•	•	-
Production data	-	-	-	-	-
Data Organization					
Unstructured text	•	-	-	-	•
Partially structured	•	•	•	•	-
Fully Structured	-	-	-	-	-
Geometric rationale	•	-	•	-	-
Level of Granularity					
Commands/ actions log	-	-	•	•	-
Work session summaries	•	•	•	•	-
Design tasks	-	•	•	•	-
Design sub-project	-	•	•	•	-
Design project	-	•	•	•	-
Level of Integration					
Centralized (single source)	•	•	•	•	•
Distributed	-	-	•	•	-
Federated	-	-	-	-	-
Schema flexibility					
Pre-determined/ fixed schema	•	•	•	•	-
Branching in schema	-	-	•	•	-
Synthesis from a fixed set	-	-	-	-	-

Synthesis from an open set	-	-	-	-	-

The steps followed in the system development were as follows:

1. Study the morphology of design information and select topical models
2. Define domain independent integration model
3. Derive domain specific models for airframe design and analysis
4. Implement a Design History System (DHS) for archival and retrieval of design cases
5. Populate and test the DHS with specific cases

We examine the morphology of design information in Sec. 5 and then present an integration model in Sec. 6. Implementation issues are discussed in Sec. 7. Space does not permit us to discuss topical models; the reader is referred to (Qureshi 97a, 97b, Shyam 98).

5. MORPHOLOGICAL ANALYSIS

By morphology we mean the scope, content, internal structure, external relationships, temporal relations, role, usage and semantics of design knowledge, data, and information (information) needed to support the design process in a real-world corporate environment. We investigated the creation, organization, archival, communication, usage and management of product design data, information, and knowledge during the product development cycle in an industrial setting (an aircraft manufacturer). Both product and process knowledge were studied. To keep the project manageable, the emphasis was on the design phase; manufacturing and other downstream applications were out of scope, although any life cycle data used in design decisions (such as manufacturability) were within scope.

Basic information element-types include system constraints, product data elements, product data features, and design process steps. Also of interest are relationships among these elements—e.g., relationships between process steps and product data elements, relationships among system constraints, product data features, and rationale. We need to represent this information in a formal way. It is best to first consider one aspect or domain/task at a time (topical models), without worrying about the inter-relations (integration). Topical models will be discussed in this section, while the integration will be discussed in the next section.

Topical models for product data, design procedures, rationale, project management, and organizational data are needed for developing our system. Widely accepted topical models for product data already exist under the ISO/STEP standard for geometry, geometric tolerances, assembly structure, finite element analysis, manufacturing plans, and material specifications.

However, product models at higher abstraction levels, such function and behavior are either non-existent or not widely used. The same applies to design process and rationale, although IBIS may be considered a pseudo-standard. We needed to develop the process ontology further.

Since design processes can represent different types of activities, an ontology should be extensible in that the user can specialize the process class with user-defined processes. Subtype processes and supertype processes are specializations and generalizations of a process respectively; decomposition of a process is orthogonal to specialization or generalization of processes. The ontology framework must contain these capabilities. Process attributes can be categorized into: 1) attributes of processes, 2) process to information element relationships, and 3) and process-to-process relations. The class of attributes that describe a process represents important information about the process itself. Of interest is the status that a process is in. A process can be active, suspended, or resumed. In addition, a process can be tentative (the designer is exploring alternatives), committed (the process is chosen as part of the whole design process), or abandoned/failed (the process generated an unsatisfactory design). The input/output objects represent the objects that a process uses, creates or modifies. 'Uses-objects', 'creates-objects', 'modifies-objects', and 'deletes-objects' are the essential process-to-data relationships. For a particular process object and associated data objects (design specifications) there can be associated rationale and constraints. Sub-process and super-process attributes of a process represent the 'part-of' relationship between processes. A process is composed of a set of sub-processes. The sub-process and super-process attributes give information about the temporal relationship between processes (i.e., a sub-process will be performed within the time-frame of the parent process). But in general, additional temporal information is needed. Such information is provided by the dependency and temporal relationship attributes. Temporal relationships and dependencies among a related set of processes are: *depends-on, independent, parallel-with, mutually-exclusive-with,* and *synchronous*. Processes also require the definition of specific process operations. These operations model the various abstract and primitive activities that occur during design. An operation such as 'decompose' needs to be associated with the process meta-class object. The operation 'decompose' will take any process and relate the process to a set of new sub-processes or sub-tasks which is at a lower level of abstraction.

Two types of constraints must be supported for the description of design histories: Static constraints are those constraints that are typically associated with structural data (i.e., constraints expressible in a first-order logic constraint language, possibly supplemented with application-oriented functions). Dynamic constraints are important since they constrain

relationships between processes or between processes and other information objects. Pre/post condition attributes of a process are one example of dynamic constraints.

A simplified form of IBIS is needed to capture rationale of a design and relationships of rationale to design processes, constraints, and data. A possible strategy is to scale down the IBIS structure to represent only issues, positions and arguments. In a simplified view, the justification may be considered to be the design rationale object. As an optional function, the post-condition attribute of a process may trigger a prompt to the designer for a rationale for the particular design action. Thus, rationale could be associated with design processes or objects at any level of abstraction and scope of the design. As with constraints and processes, rationale will be given object status to maintain uniformity of the information model and facilitate the definition, expression and manipulation of rationale.

The ontology given above for design process and rationale is domain independent. One may need to derive more specific models for domains of interest. We have derived ontologies for design and analysis of airframe structures, but this aspect of the research is not described in this paper; the reader is referred to (Qureshi 97b, Shyam 98) for more information.

6. INTEGRATION SCHEMA

The aim of the integration model is to provide a framework to relate topical models included in the design information capture system. The framework is developed by providing a basic skeleton called the "Integration Core Model" (ICM) which is a formal representation of the definition of a product in terms of one or more topical models. There are two major sub-divisions of the ICM: *Product Core* and *Process Core.* The Product Core Model provides the high level structure needed to associate process information to the product data and the Process Core Model provides the generic process information model that is needed to capture and organize process information. The ICM is developed to play the role of a generic resource similar to the integrated resources series of ISO 10303 standard.

6.1 Product Core Model

The Product Core Model defines the data-types necessary for establishing a framework for relating product data such as geometry, design procedures, function and behaviour attributes, and environmental information. The following is a description of selected Product Core Model

entities. An EXPRESS-G representation of the Product Core Model is shown in Figure 1.

***concept*:** Everything that is designed/developed is called a concept, meaning that each product would be a concept. By calling it a concept and not a product at this level, it becomes a generic representation without concern to the level of detail. Concepts have views, versions, context, relations, and description.

relations: This abstract entity represents the relations between concepts. The concepts may be related or relating to other concepts through explicit relationships in both sides. The relationships that a concept can bear with other concepts may be different each. For this reason, *relations* is an abstract class.

description: Such information includes, but is not limited to, properties associated with the product, their usage etc. This entity provides the attributes needed to uniquely identify a product. There is no commitment to any particular definition and the users of this model have the freedom to supply their own attributes that can uniquely identify a product in their environment.

product-process relations: link product models to process models.

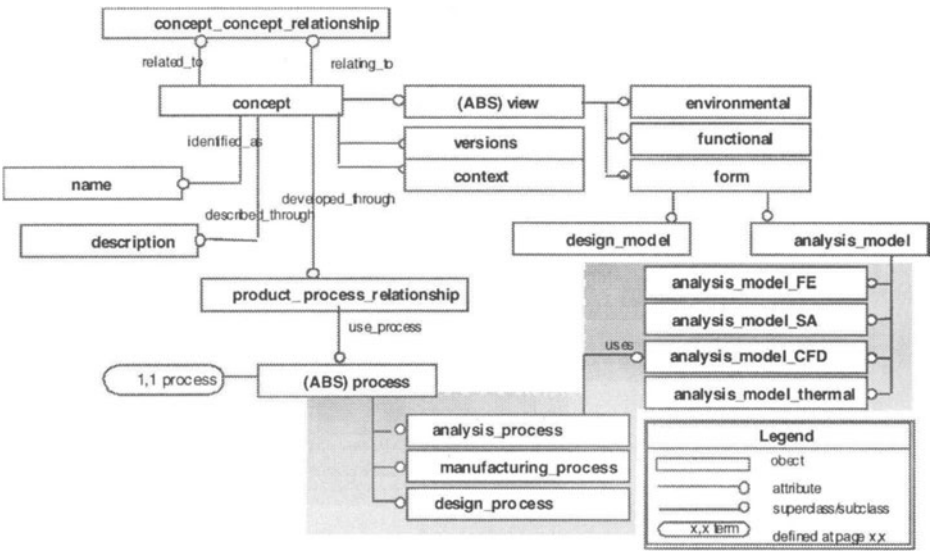


Figure 1 High Level Entities of the Product Core Model

(EXPRESS-G notation: boxes are entities; dark lines represent super-subclass relations; light lines represent attributes; little circles show direction of relation)

6.2 Process Core Model

A process can be described as it evolves and no *a priori* knowledge is necessary. Specialized processes are captured through survey studies along with the generic information that is common to all processes. The model also provides freedom for inclusion of specialized activities associated with the specialized processes. For example, a computational fluid dynamics (CFD) process would have specialized activities such as selection of flight parameters, evaluation of pressure distributions, specific to the CFD process. The EXPRESS-G representation of the model is shown in Figure 2.

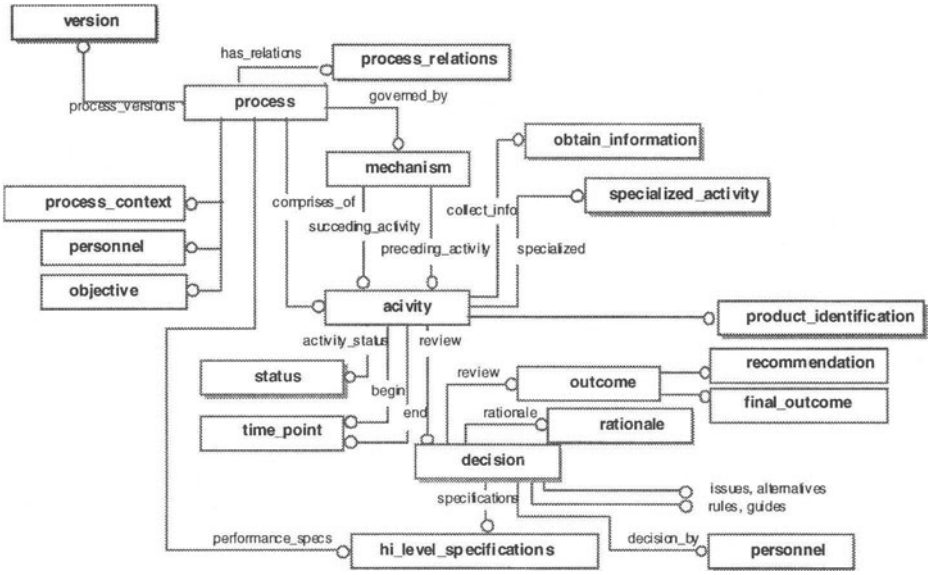


Figure 2 High Level Entities of the Process Core Model

The following information categories are used to describe the generic process information:

Mechanism: information about the sequence of *activities* that has to be maintained in order to run a process. The mechanism defines the relationship between various activities that constitute a process.

Personnel: information about the personnel involved in the process.

Decision Information: includes technical description of alternatives considered, outcome, personnel involved in decision making. The information is linked to rationale and outcome.

Rationale: Explanation of decisions made.

Time point: This information keeps track of what action was taken at what time so as to keep track of the process and plan for workflow.

Processes are associated with product data through relations defined in the product core model. The product data objects are used as inputs/outputs of a process activity. There is also a set of process attributes that define the relationships between various processes and these are expressed using mechanism information or process dependencies.

7. DESIGN HISTORY SYSTEM: IMPLEMENTATION

The following sub-sections describe the design and implementation of the second generation ASU design information capture system, DHS-2.

7.1 Database issues

Three requirements of the database management system are to faithfully represent the entities and complex relationships in the data model; to handle dynamic schema evolution; and to support user defined queries. Object oriented databases (OODB) can be used to support schema evolution, though they typically require recompilation to accommodate schema changes. In OODB all the data in a complex entity is stored in a single object. Therefore, the data in an entity is retrieved and modified as a single object. In a relational database, the same aggregate would be stored in multiple tuples scattered across multiple relations, requiring several operations to retrieve and modify it. It was thus decided to use an OODB.

The DBMS considered were ROSE, ITASCA and PSE-Pro. ROSE (Hardwick, 89) was selected because it was more stable, allows creation of objects in C++ or Java, and has a built-in connection to EXPRESS, the data modeling language of the ISO/STEP standard. ROSE also supports multiple versions of data and provides tools to aid the process of detecting differences between versions and merging versions of product data (Hardwick, 91). ROSE files can be converted to STEP Part21 format and vice-versa, a feature that can be very useful for a neutral storage in a central database.

7.2 DHS Architecture

Three types of interfaces need to be accommodated: schema definition/editing, populating the database with case histories, browsing/querying of design histories. All 3 functions could be carried out by the same user (designer), or by different people. Initially, a schema can

be loaded from an EXPRESS file and subsequent changes made interactively. Also, the current system is based on interactive instantiation for archiving case histories using forms.

It was decided to build the DHS on top of the World Wide Web because it is a familiar interface to most people and designers feel comfortable working on it. The software for the client and the server is easily available for almost all platforms and is of a low cost or shareware. The web allows for remote access easily. Distance querying of the database information over the Internet does not require high bandwidth communication links. External security features can be easily implemented by using Javascript. There is also an option of building the system over an Intranet that is not accessible outside the organization. There is a possibility of referencing multimedia records such as graphics files, audio and video. Using the web for the UI, however, also has some disadvantages. There are only limited number of interaction techniques, such as "form input" and limited graphics input. Any higher level interactions have to be provided by external programs such as applets. Security is an important consideration in industrial applications. However, space does not permit us to describe this aspect of the implementation – see (Shyam 97) for details.

A high level architecture of DHS-2 is shown in Figure 3. The web server provides a link between the various clients accessing the system and the underlying application that connects to the DHS-2 database. The DHS-2 application contains a user interface engine to generate forms for user input based on the EXPRESS schema, and a backend to access the database for populating it and for queries. The following aspects of the DHS implementation are discussed below. form generation, the backend and web gateway to ROSE, history archival process, the designer's palette, dynamic schema evolution, and the query module.

7.3 Dynamic Form Generation

The UI, which is based on HTML forms is implemented over the World Wide Web. The UI is designed to perform the following functions:

1) Instantiate the Design History Database.
2) Delete an entity from the database.
3) Browse the instance data, the metadata and the EXPRESS schema.
4) Query the database.
5) Support dynamic schema evolution.

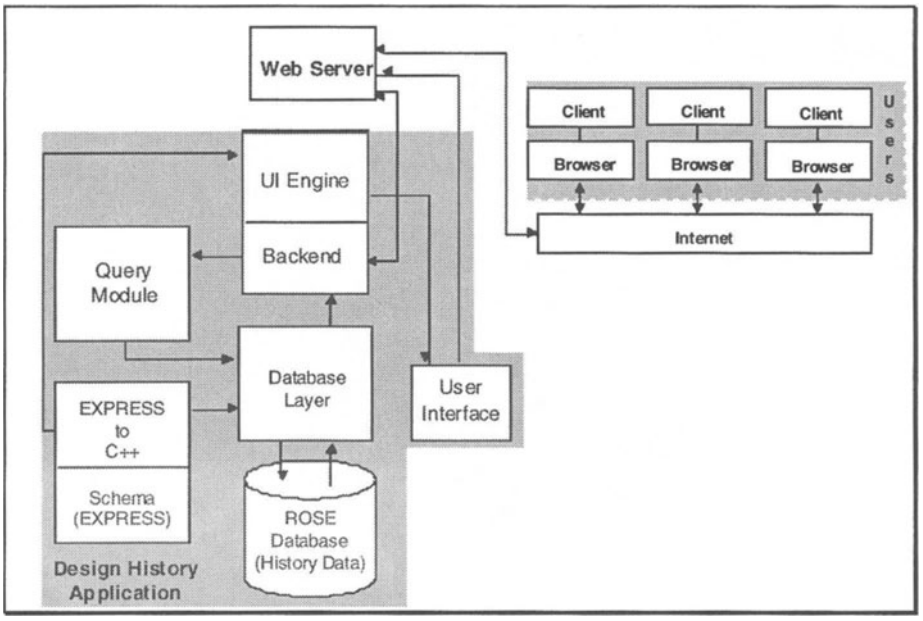
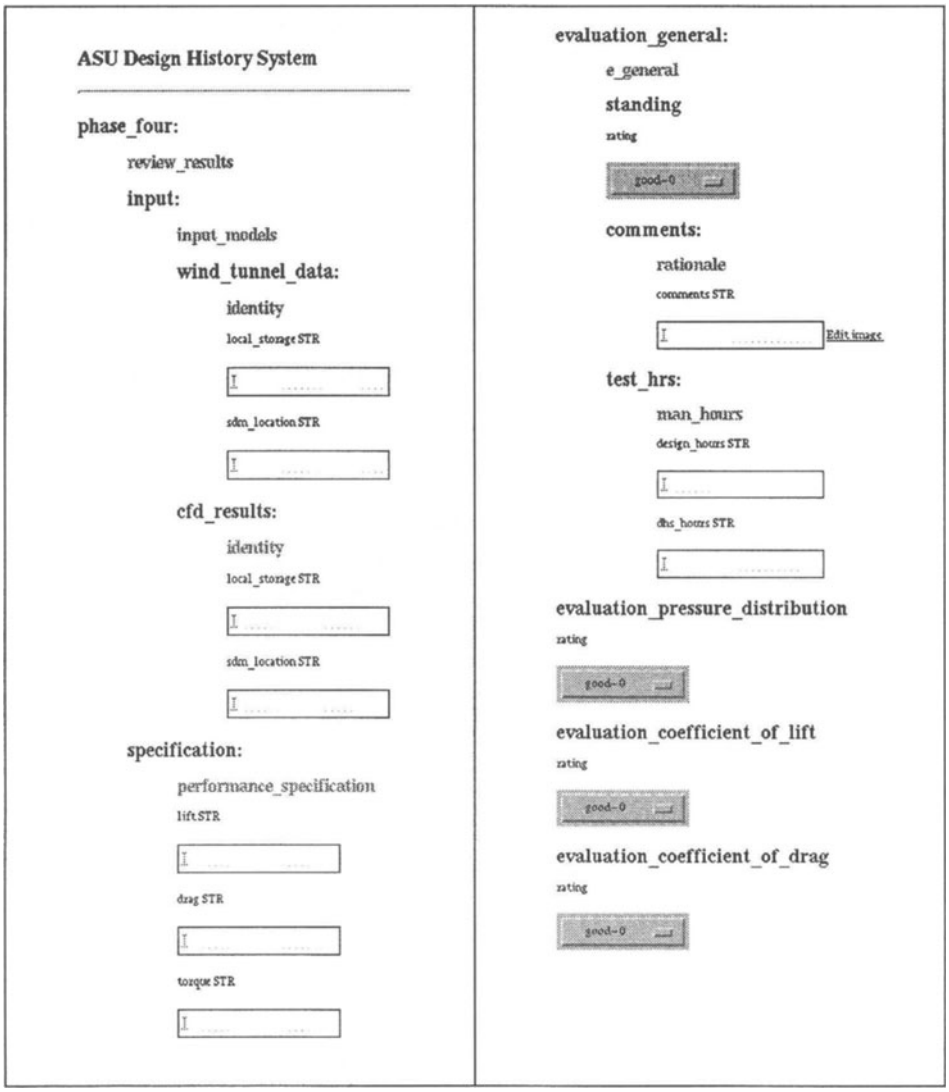


Figure 3 Conceptual Architecture of DHS-2

The forms that allow the user to populate design histories are generated automatically from the EXPRESS schema by the User Interface Engine. A typical instantiation form is shown in Figure 4. The UI Engine (Figure 5) contains an EXPRESS to HTML mapping tool. For every entity in the EXPRESS schema, HTML forms (like the one in Figure 4) are generated with input text fields, selection boxes etc. using the UI Engine. These HTML forms are used to populate the database based on the designers' input. The mapping has to proceed in a specific order because there may be complex entities that may reference other entities. First, all simple entities that have EXPRESS primitive types only as attributes are mapped. Next come all complex entities with references to other entities that have already been mapped. The third set is all supertypes with simple or complex entities or EXPRESS primitive types as attributes. The fourth set is all select types and enumeration types. The last to be mapped is the set of subtypes. A script has been implemented to create a list of entities in the order mentioned above. This list is the input for automating the entire process of form creation. This ordering is implemented in the Pre-Processor.

The output from the UI Engine consists of two files for each entity: one is a procedural file that contains the HTML definitions and the other is a script file that references the procedural file. When a form is requested by the client browser, the web server loads the script file to display the form.



7.4 Backend and Database Gateway

The backend is responsible for initial processing of the user input and depending on the user input, either more forms are displayed or a gateway is opened to the ROSE database. The backend is implemented as a CGI (Common Gateway Interface) program written in PERL (Practical Extraction and Report Language). The gateway to the ROSE database is implemented as a shell script for setting up environment and special ROSE

variables, as well as, paths that have to be set before any ROSE operation can be performed.

For history archival (instantiation), which is described in detail in the next section, the backend program checks to see if all the information is received, then passes the data on to the database layer. For example, if the input form contains a select type and the selection made by the user is another EXPRESS entity, the backend has to display a new form for the selected entity. For deletion of an instance of an entity, a form is displayed that takes the name of the entity through a text box. Next, the form for that entity is displayed and the user has to put in one or more attribute values that uniquely identify that entity in the database. The particular instance is then removed from the database.

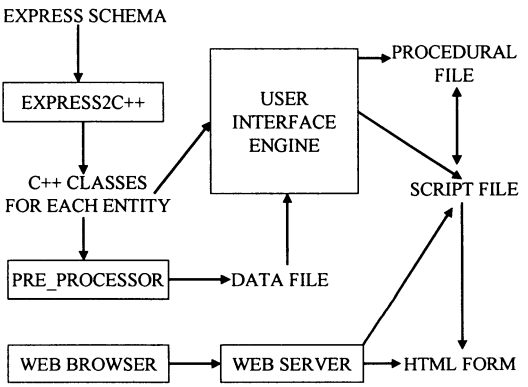


Figure 5 Generating html forms from schemas with the UI Engine

7.5 History Archival (Instantiation) Process

Histories can be archived based on the EXPRESS schema made current in the database. This does not mean that the schema itself cannot be changed; we will discuss dynamic schema evolution in a later section. For active projects, the company will need to determine the frequency of populating/updating the history. This could be at regular intervals, (e.g., once a week) or at the completion of significant sub-tasks, etc.

Archival requires interactive input via forms, such as the one in Figure 4. The EXPRESS schema is used as an input to the *express2c++* tool to generate the C++ class for each entity and a ROSE file that contains the EXPRESS Data Dictionary. After translation the C++ classes can be used in application programs that need to interact with the database, retrieving, manipulating and storing entities defined in the schema. Depending on the

user input and the entity instantiated, the backend dynamically creates a C++ program, based on the constructor for that entity in the C++ program generated by the *express2c++* tool and the user input from the form. This is an important feature as it is required for the dynamic schema evolution also.

Figure 6 shows the process of design archival, i.e. database instantiation. When the user submits a data form, a gateway to the ROSE database is opened. The dynamic C++ program generator creates a new instantiation program based on the constructor and the user input. The program is run from within the web browser. The data goes into the ROSE database and the updated ROSE instance file is displayed on the web browser.

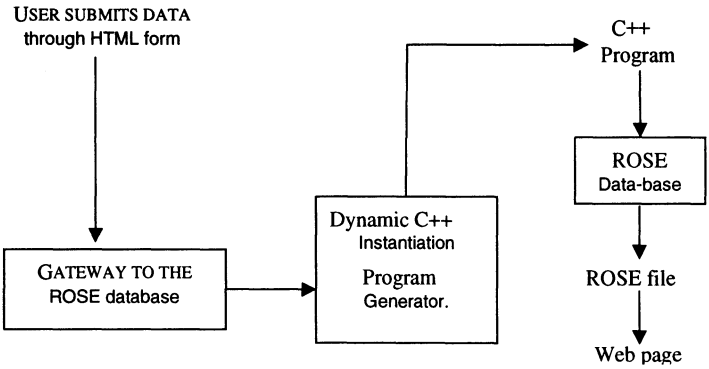


Figure 6 History archival (Instantiation) Process

During archival, the user also has the option of selecting entities for which earlier instances can be used. When this selection is made, the user can select the previous instance data and the UI is regenerated to include them as select types.

7.6 Database Layer

The database is stored as a ROSE file. There are two ROSE files for each EXPRESS schema definition. One stores the EXPRESS data dictionary and the other contains the instance data. Both ROSE files can be browsed from the user interface. New persistent objects are created in an application. ROSE applications read EXPRESS-defined objects into memory and write them to storage in clusters known as "design" objects. These RoseDesign objects hold the data objects and define operations that work on multiple objects. Objects are considered persistent if they are contained within a RoseDesign, because they can be written out to secondary storage. Objects that are not owned by a RoseDesign are considered non-persistent because

they will be lost when the program has finished running. An object may belong to only one RoseDesign at any given time. In addition to belonging to a design, a persistent object has an EXPRESS data-dictionary definition, called its domain, and a unique 160-bit object identifier, called an OID. This extra information, along with a reference to the owning design, is stored in a helper object called a manager. Instances of the RoseManager class are associated with each persistent object. Within a schema design, EXPRESS definitions are stored as instances of the RoseDomain and RoseAttribute classes. Each RoseAttribute object describes a storage slot. Depending on the EXPRESS definition, this storage slot could be an entity attribute, one component of a select, or the elements of an aggregate. The EXPRESS Data Dictionary is a compiled description of the data structures defined by an EXPRESS schema. The description is stored as a RoseDesign containing RoseDomain and RoseAttribute objects.

7.7 Designer's Palette

Non-geometric rationale is entered through the text boxes provided by the user interface. Geometric data, which is defined via CAD systems, is translated into 2D (GIF) or 3D (VRML) formats. These images can be annotated and comments can be added to express rationale, by using the "Designer's Palette". The "Designer's Palette" is a Java applet that can be viewed from within the web browser. The modified images, once entered into the Design History database, can be viewed over the Web and provide a visual browsing of geometry rationale. An advantage of using image files is that it conveys the message more clearly than simple text. The image shown in Figure 7 was annotated using this functionality.

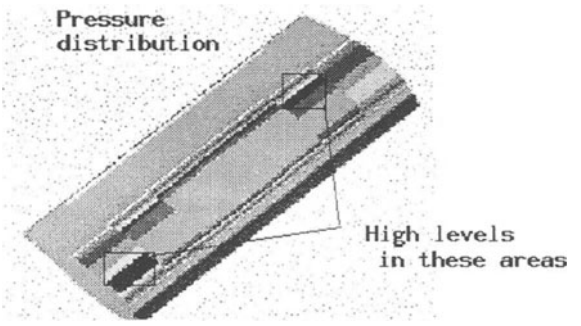


Figure 7 Archiving and annotating graphical data

With every occurrence of rationale in the schema, the UI Engine puts a hyperlink to the Java applet, along with the regular text-box for input of rationale as text. The applet allows the user to load any image file in *gif* or

jpeg format. Portions of the image can be highlighted and comments can be added or fresh drawings can be created. Once the comments and alterations have been made, the new modified image file is stored and its location is put into the text area for rationale in the original form from which the link was made. The modified image file is displayed when the ROSE file is loaded by the browser.

7.8 Query module

The querying facilities provided in DHS-2 to extract useful information from the ROSE database are based on the various functions provided by the ROSE library to traverse, search and locate objects in memory. As in the case of instantiation, C++ programs are generated based on the type of query and the backend runs these programs to give an output. Since a RoseDesign object is the basic unit of I/O for STEP data, most of these services operate within the scope of a single design.

When the user submits a query through the web browser, it goes to the server. The server runs the backend, which does the initial processing and selects the appropriate query handler from the query module. Then, the database layer runs the query using the form input, the query handler and the relevant C++ classes. It fetches the data from the database and displays it back to the user.

Queries in DHS-2 are divided into three categories: Standard database queries, pre-defined queries, and user-defined queries.

Standard database queries (domain independent)

1. Generate a list of all instances of an entity along with its attributes.
2. Objects that match certain constraints: Operators $<$, $>$, $\geq$, $=$, $\neq$, have been implemented. These are particularly useful to locate values of numerical attributes.
3. AND/OR constraints: Queries such as "display a list of Points where x=0 AND/OR y>0" have been implemented.

Pre-defined queries (specific to design rationale)

1. What was the objective of design task "x"?
2. Why was/was not alternative "x" approved?
3. What were the results of the evaluation "type y" of design "x"?
4. What were the recommendations to solve problem " y" in "x"?
5. What were the alternatives generated for design "x"?
6. What kind of analysis was performed on design "x"?
7. What was the overall rating for design "x" and why was it rated so?
8. What were the performance specs. and how did design "x" perform?
9. What were the rules that governed design "x"?
10. What were the design constraints?

An example of results of query type #4 is shown in Figure 8. In this example, "y" is aerodynamics and "x" is a specific wing design.

User defined (Dynamic) Query generation

There is an option available under 'DHS specific queries' that can be used for dynamically generating new queries that could help other designers. The user can formulate a query question that would be visible to other designers accessing the query page. Then, the user enters the entity and attribute information for the query. When the user hits the submit button, the new query is generated, and when the query page is reloaded, this new query becomes visible. The person setting up the query drafts the text in which entity and attribute names are embedded.

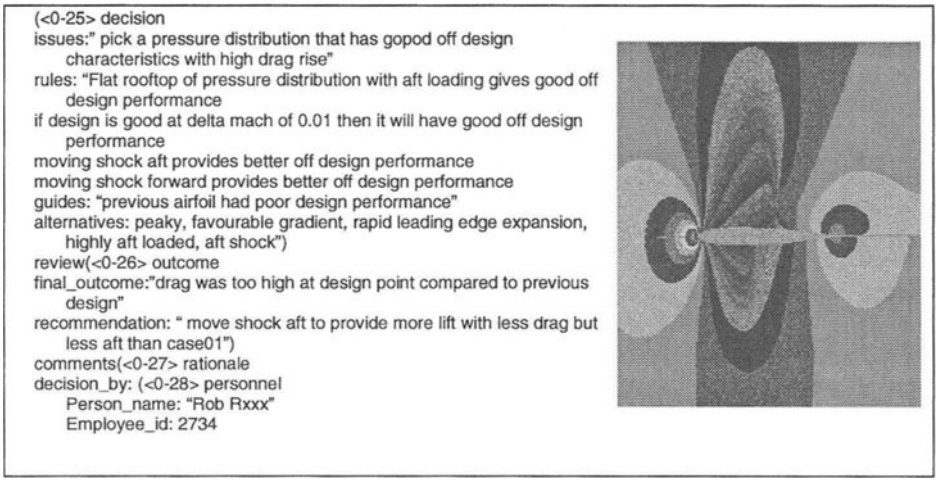


Figure 8 Results of query #4 on design 1

7.9 Dynamic Schema Evolution

The designer can modify the schema or add new entities by using the *text editor* or the *expgedit* tool provided by *Steptools*. Dynamic schema evolution provides the following three options:

1) *Attribute changes only:* Addition or deletion of attributes.

2) *Entity changes*: Addition or deletion of entities; also attribute changes.

3) *New schema*: For an entirely new schema or major schema changes.

The "attribute changes" option recompiles the new schema and generates a new set of procedural files and script files that reflect the changes made. (The operations are shown in Figure 9) It also generates a new backend for processing. If the new schema is instantiated with data, the new data goes into the same database file as the earlier data. If attributes are added, the new instance data for the new attributes is copied over to the older instances.

The "entity changes" option proceeds in the same manner as before, except that the preprocessor file is changed to include the new entities. Here again, if instances are created, the new entity instances are copied over to the previous instances.

In the "new schema" option, the database is created at a new directory location and a new set of user interfaces and backends are created at that location. The user interface then performs the same actions as before at the new location and will initialize the database and make necessary changes to the user interface. The process is shown in Figure 9. All actions can be performed from the web browser itself. Another useful feature is that the system can display the difference between two schemas. The interface takes in the names of two schemas and gives a ROSE difference file as an output which shows the difference between the two schemas.

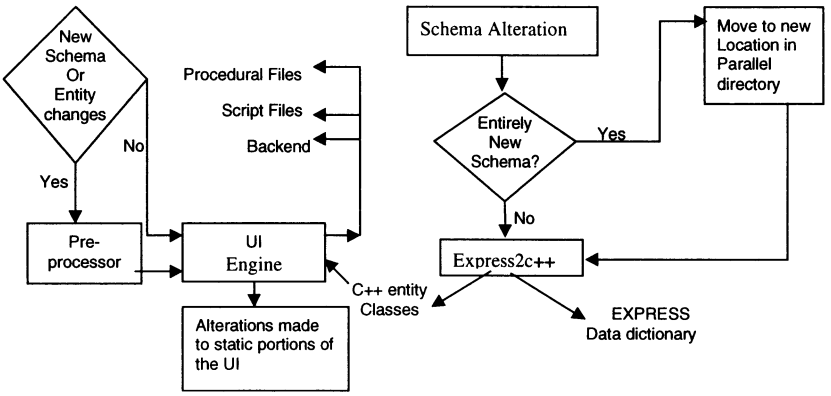


Figure 9 Dynamic Schema Evolution

8. DISCUSSION

We implemented the DHS as part of a larger project to create an Integrated Product Design Environment (IPDE) in which several design and analysis applications interact with a common data repository through Design Access Interfaces (DAI). All product data is based on STEP Application protocols AP203, AP224, and AP209. The Product and Process Core Models were domain independent and designed to be compatible with STEP. Domain specific schema for airframe design and analysis was the derived from the core model. The prototypes DHS uses this derived model and interacts with the IPDE in the same way as other applications used in the pilot project, such as CATIA for geometric design and PATRAN for FEA.

Thus, apart from having its own private database, DHS gets and puts data in the shared database at the start and end of each session, respectively. The role of DHS in the IPDE is to explain what each of the design iterations achieved and how. Two design case histories were then documented in DHS and Company personnel were able to access the system remotely via the Internet. Even this limited experiment revealed the problems of design information availability. Also, the instantiation process was quite labor intensive.

There are three areas in particular that industry may find appealing: (1) use of STEP standards in data modeling, (2) application of commercial database technology in data archival and retrieval, and (3) use of the WWW in generating the user interface. Despite these features, we are a long way from getting industry to shift from passive file management to active information integration. Barriers preventing this from happening include the following: large volumes of data produced by diverse application programs used by different groups of engineers and other involved in product development; lack of time and/or motivation on the part of designers to document design rationale; lack of commercial systems taking this approach; and heavy investment needed to set up and maintain design information systems.

One solution for reducing the volume of data to archive is to be selective, i.e., we could store only the most critical data based on frequency of re-use or the relative importance for cost or performance. Studies are needed to determine what data should be archived for most benefit.

REFERENCES

Baudin, C.; Underwood, Jody G.; Baya, Vinod, (1993),"Question –based Acquisition of Conceptual Indices for Mutlimedia Design Documentation", In proceedings of the 11th National Conference on Artificial Intelligence AAAI-93, Washington, D. C., pp. 454-458, July.

Baya, Vinod and Leifer, Larry, (1995), "Understanding Design Information Handling Behavior Using Time and Information Measure". *ASME Design Engineering Technical Conference*, Volume 2.

Brown D. C.; (1989), "Using Design History System for Technology Transfer", Proceedings of the MIT JSME Workshop on Cooperative Product Development, November 1989, pp. 545-559.

Casotto A.; Newton A. R.; Sangiovanni-Vincenetelli A.; (1994), "Argumentation-Based Rationale: What Use at What Cost?", *International Journal of Human-Computer Studies*, v40, n4, pp. 603-652.

Chen A., (1991), "A computer-Based Design History Tool", Thesis for the Department of Mechanical Engineering, Oregon State University.

Conklin, J., M. Begeman, (1988), "Gibis: a Hypertext Tool for Exploratory Policy Discussion", *Proceedings of the Conference on Computer Supported Cooperative Work*, September 1988, pp 140-152.

Conklin, Jeffrey E. & Burgess-Yakemovic, KC. (1996) "A Process-Oriented Approach to Design Rationale". Thomas Moran & John Carrol eds. Design Rationale: Concepts, Techniques, and Use. Lawrence Erlbaum Associates: New Jersey.

Domeshek E. and Kolodner J., (1997), "The Designer's Muse," in Issues and Applications of Case-Based Reasoning in Design, M.L. Maher and P. Pu, eds., Lawrence Erlbaum Associates, Hillsdale, N.J., pp. 11-38.

Faltings, Boi, (1997), "Case Reuse by Model-based Interpretation." In: Issues and Applications of Case Based Reasoning in Design, Maher M. L., Pu P., eds. Lawrence-Earlbaum Publishers.

Fu H.; Bansal, R.; Haggerty C. M.; Posco P.; (1990), "Hyperinformation Systems and Technology Transfer", Proceedings of the 1990 ASME International Computers in Engineering Conference and Exposition.

Goel, Ashok. et al. (1996), "Explanatory Interface in Interactive Design Environments". Artificial Intelligence and Design Conference. Stanford, July.

Hardwick M. and Spooner D., (1989), "The ROSE Data Manager: Using Object Technology to Support Interactive Engineering Applications", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No.2, pp. 285-289.

Hardwick, M.; Spooner, D.; (1989)," The ROSE Data Manager: Using Object Technology to Support Interactive Engineering Applications", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 2, pp. 285-289.

ISO Ind. Automation Systems, (1996), "Part 209 Application Protocol: composite and metallic structural analysis and related design", IDO/CD 10303-209.

ISO. (1996) "Guidelines for the Development and Approval of STEP Application Protocols. Volume 1.2,ISO TC 184/SC4 N433. Pages 4-5.

King, J.M.P., René Bañares-Alcántara, Ballinger and Lakshmanan R., (1995), "Using Design History to Support Chemical Plant Design", *The 1995 IChemE Research Event, First European Conference*.

Lee, J., (1990), "SIBYL: A Qualitative Decision Management System." In Artificial Intelligence at MIT: Expanding Frontiers, P. Winston, S. Shellard, Eds. The MIT Press, Cambridge, Mass., 1990, pp. 104-133.

Lee, J., Lai, K., (1996), "What's in Design Rationale". Thomas Moran & John Carrol Eds. "Design Rationale: Concepts, Techniques, and Use". Lawrence Erlbaum Associates: New Jersey.

Lubars, M D., (1991), "Representing Design Dependencies in an Issue-Based Style". *IEEE Software*. Pages 81-89.

MacLean, et al. (1996), "Questions, Options, and Criteria: Elements of Design Space Analysis". Thomas Moran & John Carrol Eds. Design Rationale: Concepts, Techniques, and Use. Lawrence Erlbaum Associates: New Jersey.

Maher M.L., Balachandran B., and Zhang D.M., (1995), Applications of Case-Based Reasoning in Design, M.L. Maher and P. Pu, eds., Lawrence Erlbaum Associates, Hillsdale, N.J.

Maher, M.L., and Zhang, D.M., (1991), "CADSYN: Using Case and Decomposition Knowledge for Design Synthesis". In Gero, J.S. (ed.). Artificial Intelligence in Design.

McCall, R. J.; (1989), "MIKROPOLIS: A Hypertext System for Design", Design Studies, V10(4), October 1989, pp. 228-238.

McCall, R., Bennett, P., and Johnson, E., (1994), "An Overview of the PHIDIAS II HyperCAD System." In Reconnecting: ACADIA '94, A. Harfmann and M. Fraser (eds.) *Association for Computer Aided Design in Architecture*, pp. 63-74.

Nagy, R.L., et al., (1992), "A Data Representation for Collaborative Mechanical Design". *Research in Engineering Design*, Volume 3, pp. 233-242.

Narashiman S., Sycara K., and Navin-Chandra D., (1997), "Representation and Synthesis of Non-Monotonic Mechanical Devices," Issues and Applications of Case-Based Reasoning in Design, pp. 187-220.

Ng, Jason L., (1993), "GSQL: A mosaic gateway to SQL", http://www.ncsa.uiuc.edu/SDG/People/jason/pub/gsql

Pena-Mora, F. & Vadhavkar, S., (1996), "Design Rationale and Design Patterns in Reusable Software Design". *Artificial Intelligence and Design Conference*. Stanford.

Pratt, M.; (1997), "Parametric Representation and Exchange (PAREX), Part 1: Overview and General Principles", ISO 14959-1: Draft.

Qureshi, S., (1997), "Integration Framework for Electromechanical Design Information", PhD Thesis, Arizona State University.

Qureshi, Sohail M., Shah, Jami J. et al. (1997), "Process Modeling to Support Design Process History Databases". *ASME Design Engineering Technical Conference*.

Rittel, H.W.J. and Weber, M.M., (1973), "Dilemmas in General Theory of Planning", *Policy Sciences*, pp. 155-169.

Santayana G.; (1905), "The Life of Reason".

Schenck, Douglas & Wilson, Peter., (1995), "Information Modeling the EXPRESS Way", Oxford University Press: New York.

Shah, J., Urban S., Bliznakov., (1996), "Database Infrastructure for Supporting Engineering Design Histories", *Computer Aided Design,* Volume 28, pp. 347-360.

Shah, J.J. et al. (1993), "Development of Machine Understandable Language for Design Process Representation". ASME, Design Theory and Methodology Conference. Albuquerque, NM.

Shyam R., (1998), "System for archival & retrieval of design history", MS thesis, Mech. Eng., Arizona State University.

STEP, (1996), http://www/scra.org/uspro//projects//std00026.html, 1996.

Thompson J. B.; Lu S. C.; (1990), "Design Evolution Management: A Methodology for Representing Design Rationale", 1990 ASME Design Technical Conference, Chicago, IL, Sept. 1990, pp. 185-191.

Ullman, David G., Paasch, Robert, (1995), "Design History, Design Rationale and Design Intent System development issues", *Design Engineering Technical Conference*.