Software architecture to facilitate automated message recording and context annotation

David T. McWherter       Vera Zaychik       Erik E. Hayes       William C. Regli
Jonathan Sevy

Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street, Philadelphia, PA 19104
http://gicl.mcs.drexel.edu

ABSTRACT

The complexities of product engineering frequently require teams of agents, both human and digital, working together to take a product through its life-cycle. Increasingly, manufacturers, designers and project planners belong to different design teams, in different corporate organizations in a supply network, separated by large geographic distances.

This paper describes the software architecture of the Collaborative Design Studio environment we are developing to support Networked Engineering. We provide the framework enabling communications exchanged during design and manufacturing activities to be annotated with design context information and archived for future reference. We present an XML-based message model to encapsulate relationships between design context and designers’ communications. As a proof-of-concept, are creating a Networked Engineering Environment with integrated email, instant messaging, collaborative work, text-editing and CAD/CAM packages.

Our architecture provides a wide range of benefits for network-based engineering. Transmitting design context with informal communications establishes frames of reference among designers. Recording informal communications aids searching and analyzing the contents of Design Repositories that archive a product’s life-cycle, documenting the rationale behind decisions. Additionally, data-mining techniques can be applied to the communications network in the Repository to analyze agent interactions—working toward case-based indexing of the design process.

Keywords: Networked Engineering, Design Rationale, Knowledge Capture, Informal Media, XML.

1. INTRODUCTION

It has long been established that engineering design is a process that involves collaboration, over distance and time, among many people, who often have very different roles and goals.1–3 As a design evolves, a great deal of knowledge and information is produced. Only a small portion of the information relating to a design can be found in the direct products of the design effort—the CAD, physical models, or standard documentation. Design Rationale, the motivations, difficulties, and failures encountered during the design typically remain undocumented, despite the fact that it is essential for future modifications of a product and the development of new products.

In this paper we describe the software architecture for a Collaborative Design Studio for Networked Engineering. This environment facilitates the capture of Design Rationale and the construction of Design Knowledge Repositories. Primarily, we concentrate on the archival of electronic communications and the association of the designer’s context with these communications.

The remainder of this paper will focus on three major components of this architecture:

Further author information:
David McWherter: E-mail: udmcwher@mcs.drexel.edu
Vera Zaychik: E-mail: uzaychik@mcs.drexel.edu
Erik E. Hayes E-mail: uehayes@mcs.drexel.edu
William C. Wregli: E-mail: wregli@mcs.drexel.edu
Jonathan Sevy: E-mail: jssey@mcs.drexel.edu
• Representation and Capture of Design Communication and Context
• Representation and Capture of Design Process History
• Storage and Retrieval for Design Repositories

We suggest an integration of design and communication tools central to industrial design into one system. We intend to make use of primarily passive techniques to gather enough information during the course of a design project so that the design rationale can be reconstructed.

2. BACKGROUND

Our first assumption is that many design projects are conducted in a distributed fashion, with groups and members of the design team located in geographically different sites. As the complexity of modern design increases it becomes more difficult to find the specialists and subject-matter experts required to design a product within a single organization. In this situation, designers must rely on electronic, Internet-based communications to bring the design to fruition. We believe that traditional design and manufacturing data (CAD, CAM, CAPP, CAE, etc.) must be combined with communications and the design process histories to create Design Repositories.8–7 We believe that Design Repositories will prove critical in supporting future Internet-based manufacturing and design.

It has been repeatedly demonstrated in the literature that a great deal of design information can be found in informal communications such as e-mail and video-conferencing.8 Unfortunately, in present design and manufacturing processes most of these communications are not recorded or archived. As a result, crucial information regarding product development, design rationale and intent will be lost. It is our goal to implement a Design Repository capable of recording and analyzing these informal communications in an attempt to preserve the design information that they contain.

We also make the assumption that most designers are reluctant to spend sufficient time documenting their work. As a result, many of the tools for design rationale that rely on manual data entry remain extremely unused in the industry. In order to ensure that our system is effective, we put a strong emphasis on reducing the amount that the system interferes with the user. There is a critical tradeoff that must be made, however, in that low-level information collected without interaction with the user lacks enough semantics to be of practical use. In addition, the lack of structure in data captured non-invasively makes it very difficult for analysis and information retrieval. There is limited prior art for purely automatic capture and retrieval of design rationale (for example, Myers et al.9) and manual techniques are much more developed in the research literature. For more in-depth surveys of design rationale research, readers are referred to.10,11

Our approach builds on insights gained from the SHARE project12,13—focusing on network-oriented distributed and collaborative design. We both concentrate on archiving a combination of informal and formal design knowledge, with different focus. A notable difference is that SHARE is process-oriented, centering on a personal design notebook where a designer enters relevant information by choice. Our approach to Networked Engineering is communication-centered—focusing on the messages exchanged in the process of design. Ideally, we would like to transparently and automatically archive all communications, requiring limited or no action on the part of designers.

3. APPROACH

Our belief, in constructing our design environment, is that designer’s processes and design rationale can be automatically archived and analyzed in a non-intrusive manner. We make the assumption that, through the use of software for CAD/CAM or email, designers will provide clues to understand the rationale that is applied to the project at different points in time. With little or no required interaction with the designer, a wealth of raw information can be harvested about the design process. This information can be used to yield better designs through understanding both current and future project design processes.

Figure 1 shows a conceptual view of our environment. We see design and collaboration ongoing in the traditional manner. Agents are inserted into these loops to (a) observe and archive design events and (b) embed design context into messages to capture reference points for the communications.

There are two methods for embedding design context into messages. First, involving human intervention, designers can point-and-click in the CAD system, selecting components or features to attach to a message context. The selection
Figure 1. The overall conceptual view of our approach to Networked Engineering.

is embedded into the message using an XML message model discussed in Section 3.2. The message then is forwarded to the archival server and deposited into the Design Repository. The message recipient can display the selection that the sender made by clicking on a link contained within the message. In addition to archiving the messages, this architecture also enhances the message, allowing the sender to describe in much less detail the component(s) of a model that they are referring to.

Our second approach to context embedding is to observe the user’s actions, and tag this information with timestamps to enable messages to be associated with primitive actions in a CAD/CAM environment. This method is less intrusive, but it does lack formal user cues to provide any information as to their motivation. After performing user studies and evaluations, we anticipate that the implementations based on the purely non-invasive techniques will be insufficient for capturing adequate design rationale information. This method is discussed in Section 3.3.

3.1. Architecture

Figure 2 depicts the internal software structure of our Networked Engineering environment, consisting of eight major components organized into three functional groups:

1. **Client Tools** for collaborative work and design authoring:
   - (a) CAD/CAE environments, along with their Application Programming Interface (API) layers;
   - (b) Computer Supported Cooperative Work (CSCW) tools, including email, and multimedia conferencing software and an Internet client/browser;
   - (c) Middleware Integration Layer, including the software infrastructure to hook into specific applications;
   - (d) Design Process Capture Agent, an event monitor for the design environments in the network;

2. **Collaboration Server** for multi-way audio/video and support of distributed collaboration applications;

3. **Design Repository and Server Tools** for archival and retrieval of the Design process data and collaboration, including:
   - (a) Archival Server;
   - (b) Collaboration Archive;
   - (c) Design Process History Archive;
   - (d) Retrieval Server.
The **Client Tools** consist of the suite of applications that currently make up the design environment in which designers operate. In our design domain, this will consist of the design authoring environment such as a CAD/CAE system with API services to allow applications to query and send commands to the CAD software. This API layer typically allows the tracking of user selections, changes, and other operations. In addition to the primary design environment, the client tools consists of a suite of communication tools such as email, chat, and audio/video-conferencing software.

To enhance the available Client Tools, we provide a custom middleware layer that integrates the CAD system and the communication tools, allowing information to be exchanged between them. This integration layer enables all messages exchanged through the system to be augmented with information detailing the design context in which the message was formed. The message model is discussed in Section 3.2.

The **Design Process Capture Agent** is a software agent that observes the user’s activities in the design authoring environment and records the design process history in the Design Repository. This is discussed in Section 3.3.

For the system’s back-end, we have been working on a package of database servers and software interfaces to handle the storage and retrieval of information collected during the design process. The **Archival Server** receives messages from the email and collaboration servers, and inserts them into the Design Repository’s collaboration archives. Once in the archive, the messages are analyzed and attached to the parts and changes in the project closely associated with this message. The analysis of these messages is difficult, and part of our future work is to investigate algorithms to accomplish this task. We believe that through sufficient annotation of the messages, we can limit the amount of reconstruction that needs to be done through natural language processing and extensive parsing, which is prone to significant error.

The **Retrieval Server** interfaces with the design repository to provide query and retrieval services over the information stored within the repository. The retrieval server supports queries based on selecting parts, features, authors, recipients, or other information found in the database.

### 3.2. Context representation and communication capture

To represent the communication and its design context, we have developed an XML-based *message model*. To capture communications, we developed agents to monitor and archive all inter-agent exchanges occurring during the design process.

As a result of choosing a multi-agent architecture for implementing our Networked Engineering Design Environment, we are faced with the potentially challenging dilemma of integrating a large number of separate tools into one unified system. Given the difficulty of constructing ad-hoc translation facilities to enable each program to
communicate with one another, we instead choose to focus on developing a common ontological representation for inter-agent communication. We make use of XML as the encoding language to form the basis for a shared ontology for collaborative exchange and design context.

The message model is an XML-based schema (DTD) to represent raw collaborative exchanges and associate them with design “context.” The message model is used as an ontology to describe semantics of all types of communications. All message share a common set of attributes which include participants in the communication, date, subject, and the actual content (text) and other fields of importance to distributed design (in particular of electro-mechanical components).

The goal of the captured design context is to unambiguously identify the design attributes (part, feature, parameters, etc.) being discussed in the collaborative exchange. This information is extracted from the CAD system and, ideally, includes semantic information about the state of the design space at the time of the collaborative exchange. Currently, our model supports capabilities and features available in off-the-shelf CAD environments. For example, we consider the design context to be defined by a tuple consisting of PROJECT, and a sequence of MODEL, ASSEMBLY and PART elements. A MODEL describes a name for a particular instance of a modeler data file. An ASSEMBLY consists of the name of a collection of solid modeler parts, such as an “Engine.” A PART represents an individual (usually a static solid model) component in a system, and is made up of a semantic name, number, version identifier, label, and a list of features if any were selected. However, this does not have to be the only design context representation. Our model can be extended with XML to accommodate other design contexts such as structure-function-behavior (SBF) representation. 

Our communications XML schema has facilities to represent the intent of a speech act. In general, this is a very complex concept, and we limit ourselves only to discerning those intentions relevant for understanding rationale in the industrial design domain. In our current model, we allow communications to be tagged as requests for change, inquiries, inquiry responses, advisors, solutions, problems, or information exchanges. This information can be, ideally, inferred from the communications, but in the meanwhile the user is prompted to choose the most appropriate one.

The approach capturing communications will depend on the mode of collaboration and its media types. For example, email is relatively straightforward: we provide a plug-in for the CAD/CAE system that enables the design environment to be synchronized with the email messages that are sent and received; and these messages are automatically carbon-copied to the Archival Server. For an asynchronous chat application, we provide an option of including design context at any point in the conversation (via a user prompt for selection). The chat transcripts are saved by the Collaboration server and forwarded to the Archival Server. Other media types, such as streaming audio and video, can be handled in a similar manner. We have also explored non-invasive recording of context and communications, using agents to monitor exchanges and cross-reference them to currently active components in the design environment.

3.3. Design process history representation and capture

Most existing design environments record design histories as procedural descriptions of how to create a final model of an artifact via a series of primitive operations and parametric refinements. The design process history is the design space explored by the agents working on a design task. In contrast to existing design histories as implemented in commercial CAD systems, the design process history would include all of the backtracking, dead-ends, changes, deletions, modifications, and discarded paths explored in the creation of the final design. This information provides a wider view of complete design space and can provide a reference point for the captured context and communications.

3.3.1. Design process history representation

The process of designing within a CAD environment primarily consists of three types of operations: the creation of objects, the deletion of objects, and parametric changes to the attributes of existing objects. Recording these events provides the ability to maintain a large amount of knowledge about the evolution of components over time. A set of requirements suitable for representing the design process history in a CAD/CAE domain includes:

1. Completeness: The structure must be able to describe the complete design history of a product. In particular, deleted and modified elements that are no longer directly found in the final model must be accessible in the structure.
2. **Temporal Validity**: The structure must be able to provide the means to construct a chronologically ordered view of all of the design events (object creations, deletions, or modifications) along different phases of the product’s development cycle. The order in which events occur is essential to understand the rationale associated with the design.

3. **Extensibility**: The structure should be flexible enough to enable various components of Networked Engineering environments to make practical use of it. Applications should be able to manipulate the data structure to suit their needs without having to construct a new data structure. Potential candidate areas of use include tagging design team communications to such a structure and design rationale extraction.

The data structures we are examining to capture model design process history and satisfy these requirements are based on *directed acyclic graphs* (DAGs). The details regarding the nodes and edges in the design process history graphs are as follows:

- **Nodes**. We expect three basic types of nodes to be stored within the DAG. These are feature nodes, operation nodes, and change nodes.
  1. **Feature Nodes**. Feature nodes describe features present in a solid modeler. Items such as holes, pockets, or cuts will be represented as feature nodes. All of the relevant feature information, such as name and dimensions, will be stored within these nodes.
  2. **Operation Nodes**. Operation nodes reflect actions that relate feature nodes within the CAD environment.
  3. **Change Nodes**. Change nodes are extensions of feature nodes. They represent a modification made to the data contained within a feature node, such as a change of dimensions, or name. Any time a feature in the modeler is modified, a change node is added to the structure indicating the type of modification made.

- **Edges**. Edges within the design history structure are directed in nature. We are still examining rationale for structural conditions on the existence of edges in the system. One of the edge conditions we are examining is based on inserting edges to impose a chronological ordering of the vertices in the graph. We maintain a linear history of the design history, and as deletions and changes are made, the DAG branches at the node where the deleted/changed object was created, and the remaining tree is deleted or duplicated in the new branch as appropriate. We anticipate that we will require that the in and out degree of each node in the graph will need to be limited to a small number, in order to ensure the development of tractable algorithms for this structure.

### 3.3.2. Design process history capture

Much of modern design now takes place within some sort of CAD environment. We propose to create CAD Agents to acquire process information through event-based communication with the design environment. Many CAD systems on the market provide an application programming interface through which it is possible to interact with the design environment. Through the use of these interfaces, the desired information can be extracted from the environment. Agents can thus be programmed that are capable of running in the background of the CAD system. Because interfaces vary with respect to different CAD systems, the information contained within the graph structure described above may have to be updated to be more universal once the discrepancies between the interfaces have been examined in greater detail.

CAD Agents developed to capture design process information should be capable of fulfilling the following requirements:

- The agents should be capable of detecting a certain set of events. These events generally include creations, deletions, and modifications. Upon detection of these events, the agents should record the necessary information.
- Design process capture agents should extract the desired information with the minimal amount of aid from the designer. Past attempts at capturing a detailed design history have imposed too heavily on the designer and were thus met with little enthusiasm. Therefore, the agents should strive to be as independent of the designer as possible.
• The agents should be programmable via an abstract interface so that an agent can interact with any CAD environment equipped with the appropriate interface code. This enables the agent to behave in similar fashions in various environments and produce identical data structures regardless of the CAD environment being used.

At present, we have created CAD Agents to automatically capture design process history graphs for single-user CAD sessions.

3.4. Storage and retrieval of design communication and process histories

Consider an small-scale, workgroup sized design project. During the course of such a project, an extensive amount of data related to the design process will be created. We see this data, consisting of design data, design process history graphs, email and other communications, as being archived in Design Repositories for efficient search and retrieval. In the long term, Design Repositories will become the platform for future knowledge-based design systems and support case-based retrieval of past design experience. Towards this goal, we are developing graph-based data representations to store this data in Design Repositories. Design process history graphs are directly representable in this form. Archived communications stored in the Design Repository can be represented using a variety of graph-based structures (for example, call-graphs, semantic networks or conceptual graphs).

Users of the Design Repository will need to be able to construct custom “views” of raw data collected from designers’ communication in order to analyze and understand different aspects of the design process. For instance, to understand large-scale role assessment of team members, a less detailed communication graph that captures information at the conversation level may be appropriate. On the other hand, to understand the complex form of questions and responses between designers resolving a conflict in the product design, a much finer resolution communication graph is likely to be needed.

To make this functionality possible, the Design Repository needs to be able to efficiently store complex graph-based data structures. Graph structures are frequently represented in conventional relational database systems (RDBMS) as a relation between two vertices in the graph, and an edge type. For instance, a simple graph may be represented as the binary relation $\text{HasEdge}(\text{vertex}_1, \text{vertex}_2) \in \{0, 1\}$, in which the relation is 1 if an edge is between two vertices, and 0 otherwise. This treatment, however, fails to take into consideration the structure of individual graphs in a collection, making efficient queries very difficult to perform. Some research is being conducted on storing collections of highly structured graphs in a system known as GRAS, although we believe this system provides services that are generally too low-level for use in the Design Repository.

We attempt to develop a fairly generic technique for efficiently storing and indexing graph databases through the use of “graph similarity” measures. In our architecture, we construct one or more functions that compare two graphs, and produce a number representing a distance between the two graphs. For maximal performance and consistency, we require that these functions meet particular requirements, such that they are metric. In practice, however, we expect that most domains will require the use of functions that approximate these requirements.

Through the use of similarity measures which are metric, we can make use of a number of indexing techniques for metric spaces. The indexing techniques that we plan to make use of are those of either metric trees or generalized hyperplane techniques. These indexing techniques enable efficient performance for executing queries for structures within a radius around a model graph. As a result, our database will be able to efficiently handle queries of the form “Show me all of the graphs in the database which are most similar to graph $G$,” which we believe is a very powerful tool for searching large complex data such as the communications graphs. We are also exploring techniques from spectral graph theory to compute an approximate distance metrics for graphs.

We anticipate that a large number of distance metrics will find uses in enabling case-based retrieval from Design Repositories, especially those that do significant domain-specific analysis between the communications and design process graphs. Currently, however, we focus primarily on the problem of performing comparisons between graphs based solely on their structural similarities.

4. STATUS AND IMPLEMENTATION

We implemented the initial prototype of the system using Structural Dynamics Research Corporation’s (SDRC) I-DEAS Masters Series CAD system which provides a CORBA-based API layer called Open I-DEAS. We used Netscape Messenger as the email client and Mitre’s CVW as a collaboration server. CVW provides client-side audio/video
conferencing and shared whiteboard. Our client communication tools have been designed as modules which work
in concert with standard communication software, so that users will not be required to switch from those they are
already using to gain the benefits of archiving. The email archiving client extracts project information from the CAD
package, encapsulated it as a partial XML message (to be completed by the archiving server), and starts a chosen
email client with this message specified as an attachment and the archiving server as one of the recipients. This
sequence can be initiated through an entry in I-DEAS Application Launcher, making this functionality immediately
accessible from within the CAD environment.

5. CONCLUSION AND OPEN ISSUES

In this paper we presented our vision and software architecture for creating a Collaborative Design Studio for
Networked Engineering. This environment integrates CAD, CSCW and Repository services and facilitates automated
message recording and design-context annotation, as well as tracking of the evolution of project designs. In addition,
we store and archive this information using techniques to enable efficient retrieval of critical information.

The Collaborative Design Studio is a work in progress. There are still many issues that need to be addressed
before a functional integrated design studio is able to be available. The question of evaluating Networked Engineering
Environments is extremely complicated. The sheer complexity of the number of systems required to complete a
project like the design studio results in a huge number of parameters that can be adjusted to alter the system. How
much interference is too much? How much information can be extracted without any user intervention? Are the
standard CAD environments integrated with standard communication tools the way to go or maybe a new paradigm
is in order22? Techniques to evaluate the performance and perceived benefits on the part of the users, as well as
evaluation of final designs are required to determine the effects altering these parameters will have on the design
environment.

In addition to the human-computer interaction questions remaining to be answered in the Collaborative Design
studio, a number of implementation-based issues remain to be addressed. The development of similarity metrics
used in indexing our graph-based data storage still needs much work, especially for facilitating queries that exploit
domain-specific information. In addition, the primitives we use in constructing the representations of design histories
in the system need to be evaluated to determine the “universal” set of design operations that need to be recorded.

We anticipate this research will further our understanding of how design activities can be effectively supported
with cooperative work tools and build the foundation needed to create large-scale Design Repositories.

ACKNOWLEDGEMENTS

This work was supported in part by National Science Foundation (NSF), Knowledge and Distributed Intelligence
in the Information Age (KDI) Initiative Grant CISE/IIS-9873005; and Grants ENG/DMI-9713718 and CISE/CDA-
9729827. Additional support was provided by AT&T Labs.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s)
and do not necessarily reflect the views of the National Science Foundation or the other supporting government and
corporate organizations.

REFERENCES

   *IEEE Internet Computing* 4, pp. 54–62, September-October 2000. Special Issue on “Knowledge Networking.”
   and J. C. Weber, “Pact: An experiment in integrating concurrent engineering systems,” *IEEE Computer* 26,
   Technical Conferences, 19th Computers and Information in Engineering Conference, ASME, ASME Press, (New
   York, NY, USA), September 12-16, Las Vegas, NV 1999. Special Track on Internet-Aided Design, Manufacturing
   and Electronic Commerce.*


