The Case for Design using the World Wide Web

Mário J. Silva and Randy H. Katz
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776

Abstract

Most information and services required today by designers will soon become available as documents distributed in a wide area hypermedia network. New integration services are required from the design environment, supporting business transactions with design information providers, automatic exchange of design data between independent groups, and integrated support for new forms of collaboration. We discuss design using electronic commerce and other services based on the Internet, and propose a hypermedia system organization for a new generation of CAD systems, conceived to make efficient use of that infrastructure. We also describe our experience as designers of an integrated design and documentation system that interfaces existing design and documentation tools with electronic commerce services based on the World Wide Web.

This project was supported in part by NSF under grant MIP-9002962.
Email contact: msilva@cs.Berkeley.EDU.
This document is available as UC Berkeley Technical Report number UCB/CSD-94-837

1. Introduction

Industries are adopting new business models characterized by close cooperation between independent organizations, known as virtual corporations [8]. Key to this concept is the rapid exchange of services between organizations. This implies the existence of an ubiquitous infrastructure that makes it possible to perform business transactions, advertise and distribute information products on a common network. There are already many initiatives dedicated to creating these infrastructures around the globe. One example, focused on providing these services to the electronics industry, is Silicon Valley's CommerceNet[1].

The World Wide Web (WWW) is becoming the *de facto* standard for providing information on the Internet [2]. The WWW software is based on the exchange of electronic documents using a client-server communications protocol. This protocol is well suited to transport the information manipulated by designers across organizations. The primary information format is derived from SGML, the ISO standard for on-line documentation representation [9]. However, any other data type can be transferred when encapsulated according to Internet conventions for encoding multimedia data. As a result, we can easily adapt existing WWW software to transport design data. CAD interchange formats can be defined as new media types, and included in multimedia documents.

In an earlier prototype, we have used multimedia documents as a common front-end to multiple design tools [13]. This was built as an open hypermedia system supporting remote command executions and hyperlinks between heterogeneous tools. In our system, the files containing these links are called *active documents*. A hyperlink in our system can contain data and a program to be executed upon activation. We call these links *active messages*. Designers use an electronic notebook to navigate in the web of information, add annotations and create links. This provides them with an integrated view of all design related information and tools as documentation (see Figure 1).

1.The URL for CommerceNet is http://www.commerce.net/

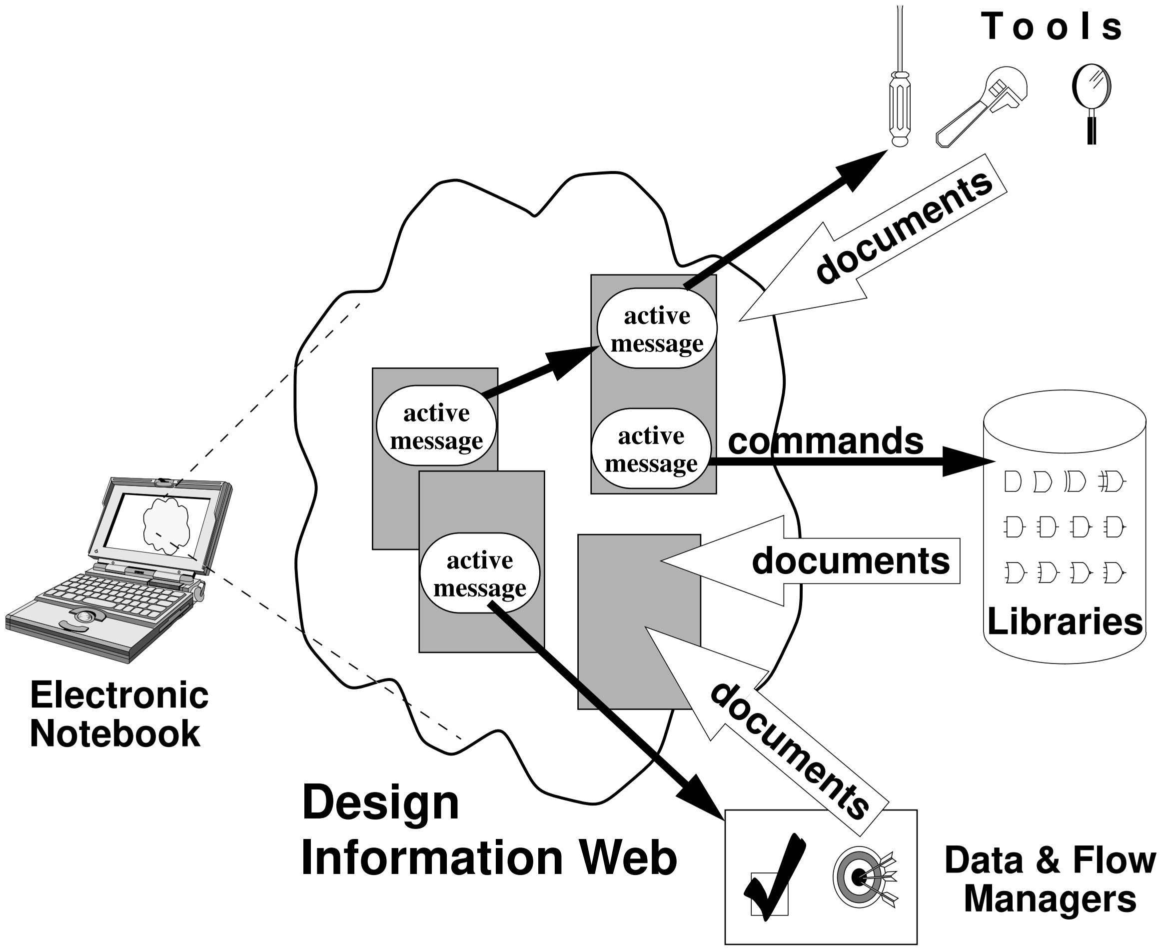


FIGURE 1. Information flow in the Henry System. In the information-centered environment of the Henry System, designers access tools, libraries and design data and flow managers via active documents. These are connected into an information web through active messages stored within the documents. Active messages fire the commands to the tools. Command executions create additional data, which is integrated into the web as new documents.

The documentation paradigm for interacting with design information has the additional advantage of being well adapted to interface with the new services offered to system designers through the WWW. In this paper, we describe the new architecture of Henry. This has evolved to support design methodologies that include the use of electronic commerce and support concurrent design involving independent groups using heterogeneous CAD systems. We discuss protocols, exchange formats and tools that we adapted for supporting the design process in the new kind of internetworked environment. These are presented in the context of our experiments extending the use of hypermedia as the integration tool for the construction of integrated design and documentation systems.

The rest of the paper is organized as follows. In Section 2, we present an overview of the new services that can be offered to designers based on electronic commerce. Section 3 discusses the limitations that would be faced by designers and system integrators using the available frameworks in an internetworked design environment. In Section 5, we describe the architecture of Henry. In Section 6 its implementation. Section 6.2 describes design scenarios involving the use of Henry in electronic commerce. Section 7 closes the paper and presents directions for future work.

2. Electronic Commerce Services for System Designers

With the infrastructures for electronic commerce in place, new design methodologies based on the outsourcing of design and manufacturing will become possible. We anticipate that electronic commerce will make it possible to offer multiple new services to electronic design and manufacturing organizations:

CAD outsourcing. There will be specialized CAD systems for specific design tasks available. For instance, a company may sell the use of dedicated hardware and software for performing large and expensive simulations. The advantages of using this service would be those associated with outsourcing in general: economies of scale, focus on core competencies and the flexibility of usage based costs.

Collaborative Design and Design/Manufacturing Integration. This involves adopting standards for conferencing, shared editing and exchange of design information. These will enable much closer interactions between contractors and sub-contractors, speeding and increasing the quality of the artifacts produced.

On-line Component Information Services. This will offer the ability to quickly retrieve datasheets and select components for a specific purpose. A great deal of component related information, mostly digitized from paper databooks, is already distributed in CD-ROM. This information will eventually move to on-line services. The Pinnacles specification of a SGML document type for this purpose represents an important advance in this domain [1]. It may also contain schematic symbols, simulation models, and application notes. Information will be much easier to incorporate in design frameworks and more up-todate. New billing methods, based on the actual information retrieved, will be possible. Component information will become affordable to smaller organizations.

Broker Services. New brokerage services that can search for specialized information will be available. These services will make appraisals and give recommendations about design services. They may provide application notes for application-specific designs with embedded requests to access entry points to endorsed service providers. The ISI has a research project in this area [10].

Business Services. These will be the non-design specific services that will form the backbone of electronic commerce infrastructures. They will include electronic Yellow and White pages, electronic payment services using Electronic Data Interchange (EDI) standards and certification authorities for authentication.

3. Limitations of Current Generation of CAD Frameworks

The current generation of framework-based design environments makes the assumptions that (1) a single framework controls the entire design process, (2) all information is available locally to the design team, and (3) the design is the product of a single organization In addition, frameworks only manage design specific tools and artifact data. In our view, these assumptions are no longer valid. The next generation of *information centric design environments* will have new basic requirements. They must integrate multiple, independently managed, heterogeneous frameworks, and be capable of accessing on-line services available through electronic commerce. They also should have a flexible structure, adapted to new business models and concurrent design involving independent organizations.

Current CAD Frameworks attempt to integrate the entire design process. They offer multiple common services, such as design methodology management and inter-tool communication. However, implementation of a common design Framework supporting all the tools used in the design environment is hard to achieve. In a typical CAD environment, the vendor supplying the logic synthesis tools is not the same as the vendor that provides the best printed circuit design or chip layout tools. As a result, we observe that system design environments have multiple sets (or clusters) of integrated tools in use. Good progress in the standardization of common data representation formats, such as VHDL and EDIF, has been what makes it possible to build environments composed of multiple frameworks. In each framework, the tool set can operate satisfactorily as an ensemble. Interchange formats make it possible to move the design data from a tool set to another. We are at a point where integration is good within each cluster, but poor when passing of control information between tool sets is required.

In addition, system design involves many non design-specific tools, not integrated within any design framework, which have a crucial role in the design process. Examples are the FAX and electronic mail processing tools. These have always been used in the design process and are increasingly becoming fully integrated components in collaborative design environments. Despite all the standardization attempts, it remains hard to integrate all the new tools that are constantly being added to design environments.

4. System Design Using an Integrated Design and Documentation Approach

One approach for integrating the heterogeneous data, frameworks and on-line services that designers must cope with is by providing a new viewport into the design environment, based on an active document manipulation metaphor. We developed Henry, a prototype VLSI design system built as an open hypermedia system supporting remote command executions and hyperlinks between heterogeneous tools. The architecture of the system has been re-designed to provide an interface to WWW protocols. This makes it possible to use the system for automatic exchange of data between different groups and to access electronic commerce services.

What follows is not a report of our experience using Henry in actual designs, but the description of sequences of design operations that can actually be performed with the existing prototype, reflecting realistic usage scenarios. The design of the Henry architecture proceeded through the development of these mock-ups. We used them to validate the architecture and inter-operation between the various components of the environment.

Creating usable electronic commerce services to designers is not simple. There are several problems in multiple domains that need to be considered, including:

Authentication of clients and security of communications. For these, we can use cryptography techniques. Certification authorities, digital signatures and Privacy Enhanced Mail are developed technologies that could be used for this purpose [6].

Billing. For this we could use existing software for automatic placement of orders and payment, based on EDI, the Electronic Data Interchange standard.

Intellectual property protection. This is of major importance to clients, as they need to have guarantees that the models loaded into the remote server will not be used by someone else. We believe that a combination of legal mechanisms and technical barriers could be created to provide the necessary protection.

Our goal in developing scenarios involving the use of electronic commerce by designers focused on finding the appropriate flow of information and sequencing of tool invocations required to implement the services. For the development of these, we assumed that the above problems could be addressed by re-using existing software. When in the remaining of this section we mention billing, authentication or encryption, we refer to the point where these operations should be performed. In the mock-ups we built using the tools integrated in Henry, these are not actually executed.

We have developed two scenarios. On the first, we prototyped the operation a SPICE simulation service that would be accessed over the Internet. This is organized as follows. There is a WWW home page that advertises the service. From there, it is possible to retrieve the terms and conditions for its use. When a designer decides to use the service, the contractual forms are sent to be filled-in interactively. Once completed and authenticated, the designer receives a document with a digitally signed contract. The document includes a URL that can be used in the future to request simulations. To post simulation decks for processing, designers use the *postmessage* script. Simulation requests received at this URL are authenticated and billed to the client when completed.

For the second scenario, we considered a designer selecting and ordering an off-the shelf chip, its documentation, models and application notes from a catalog on the WWW. We describe the operation of this in more detail in the remainder of this section.

4.1 The On-line Component Selection and Ordering System

Our goal for this scenario was to develop a mechanism for selling complex VLSI components on the Internet. Information would be presented in a similar way to that used by the MSU Microsystems Prototyping Lab library project[1]. However, we made different assumptions about how this information would be available. Access to part of the information would be restricted and given for a fee. The business transaction would be performed automatically using electronic commerce. In addition, instead of providing bitmaps of the layouts in GIF format and delay information as tables, we wanted to be able to send the layouts in a CAD interchange format and simulation models along with propagation times tables. In addition, we wanted to have the files automatically installed in the clients databases via active messages.

To order a component, a designer first consults a manufacturer's database with their specifications and application notes illustrating their use. Once connected to the database, he receives a document with a catalog of the available information. From the catalog, he can retrieve a *preview*, containing publicly available information about the component, such as its basic characteristics, cost and usage terms. Next, if the designer decides to order it, he fills-in an electronic form containing the company's identification, type of framework where the component models and schematics will be installed, interchange formats accepted, address and payment method. In return, the designer receives a document. This contains the transaction receipt and information on how to retrieve the information.

1.The Mississippi State University Microsystems Prototyping Lab Standard Cells Library is located at URL *http://www.erc.msstate.edu/mpl/libraries/stdcells/*

Clients can retrieve the information in several forms. The simplest way is by activating the hyperlinks to the URLs[1] in the library server pointing directly to the simulation models and schematic symbols for the purchased component. With minimal extensions to the configuration files on both sides, we can have the appropriate tools invoked to display design files directly. However, as in this method we retrieve the files one at a time, activation of links between the files that constitute the component's information package is not possible. This is because the links use relative addressing to refer to other files.

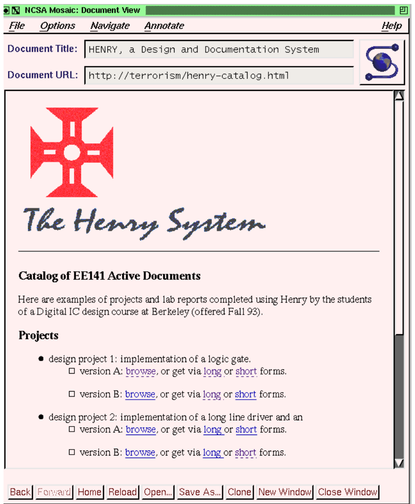
Full browsing capability only becomes possible when clients have the Henry tools installed. These may download all the component's information in a single active message. This contains the complete set of files for that component plus a script to install them in a directory structure reflecting that of the server. Links between the files in the package can then be directly activated. This results in the invocation of the design tools to browse the received data and install it in the local project database. Figure 2 shows the windows seen by the user when retrieving a component's information package with the Henry System.

5. The Henry Integrated Design and Documentation Environment

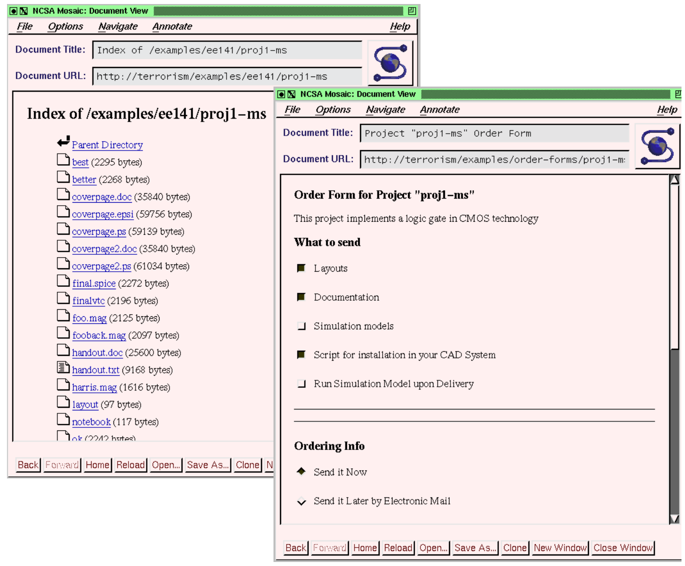
The new electronic commerce infrastructure creates the demand for new design environments providing support for simple and efficient use of the services it offers. New design environments will integrate multiple frameworks and will provide support for automated exchange of data [7]. It will be possible to create automated design and manufacturing processes with control information passed across organizational boundaries. Simulation environments spanning multiple organizations, each specialized in a different domain, will allow for rapid prototyping of very complex hardware/software systems.

We envision design data distributed across a wide-area network, organized as a web of related pieces of information. CAD systems will integrate tools designed to communicate on distinct protocols. In addition, they will have to support user-mediated asynchronous communications. When we consider inter-organiza-

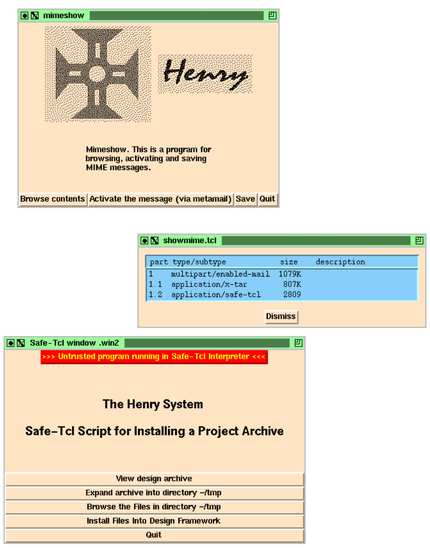
1.URL — Universal Resource Locator, the specification of the address of an object in the WWW.



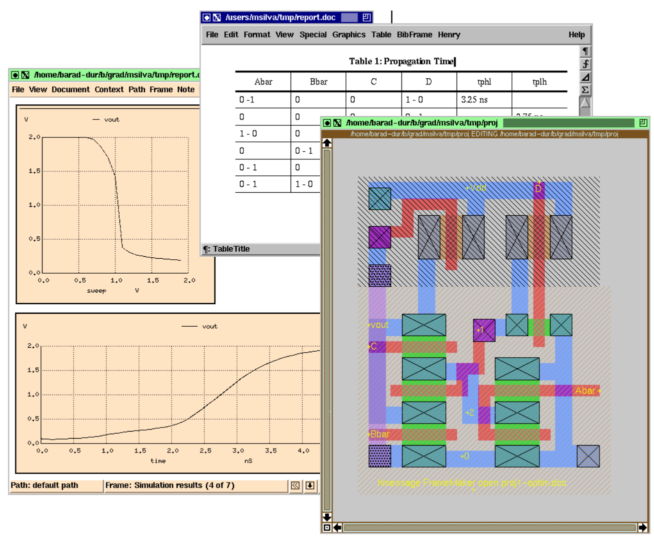
1) Electronic Catalog Home Page



2) Browse the Catalog, then fill the form to place an order



3) An active message with the information package requested and a script to browse and install it is received by Mosaic. Henry's active message browser is called to evaluate it.



4) The active message script then starts the tools to browse the files in the component's information package. Links between the files may be activated. Finally, the designer may go back to the active message program's window and give a command to install the files into the local project database.

FIGURE 2. The User Interface of the Electronic Component Library. The figure shows the sequence of windows presented to the designer when ordering through a catalog available on Mosaic from a service supporting active messages. Component information is packed into an active document, which is browsed locally with the Henry System Tools

tional communication related to collaborative design, the traffic will also contain unsolicited messages that, for trustability reasons, may require user inspection before actual delivery.

In this section, we present the main architectural concepts of the Henry System, an initial prototype of a CAD system that integrates documentation into the design environment.

5.1 Communications in Henry

The Henry System uses communication protocols suited for the new active document-based design environment. Active documents have the capability to send and receive commands and data. We call these commands active messages, as they resemble the messages used in active mail systems [5]. These extend electronic mail to transport not only data but programs that can be activated upon reading.

Active messages contain data and commands to be performed on that data upon delivery. Active messages in Henry can be transported via SMTP[1] and handled by conventional mail readers, as in active mail systems. However, we use them not only for communication between end-users, but fundamentally for intertool communication. For instance, a user browsing an active design document may generate an active message requesting a database to return the layout of a circuit being described. The layout could come in the form of another active message addressed to the layout editor, that would in turn display the data contained in the message.

Remote procedure call (RPC) protocols used for inter-tool communication in CAD Frameworks, such as Tooltalk [15], are optimized for activating commands remotely with small latencies. Data is assumed to be available via a common file system implemented using NFS[2]. However, this combination of protocols does not scale well when we consider larger networks consisting of multiple organizations exchanging commands

1.SMTP — Simple Mail Transfer Protocol, the Internet standard for exchanging electronic mail messages between hosts.
2.NFS — Network File System, the Internet Standard for accessing remote hosts's file systems transparently.

and data, as they were not oriented to support the transaction-oriented paradigm for accessing information required by our application.

A protocol more adapted for the exchange of active messages is HTTP, the client-server communications protocol used in the WWW. In Henry, we use HTTP to transport active messages. HTTP uses MIME[1] as the encoding mechanism to pack information into messages [3]. Communication is handled by HUBs, message servers that communicate with tools and exchange active messages (see Figure 3).

The organization of the design environment based on a web of HUBs has the flexibility required to adapt to the dynamic constellations of business units that characterize the virtual corporation. In the Henry System, each user has an associated HUB running on his workstation. These manage the activation and inter-tool communication between the tools run by each user. In addition, groups of users can set up a HUB for handling messages for which the dispatching procedure requires knowledge of the group organization, such as broadcasts of messages addressed to team members assigned to a specific task. In an electronic system design team, group HUBs would resolve message addresses like “logic designers” or the “PCB design manager.” In a similar way, deeper hierarchies could be established to support larger groups with multiple teams.

As HUBs use the Internet message exchange protocols and formats, it is possible to create design environments with very heterogeneous frameworks and many levels of integration. The possibility of exchanging design objects and commands to remote design systems via electronic mail, makes it possible to create multi-organizational design environments operating at various levels of integration. At one site, processing of a given active message could consist in forwarding the embedded commands for execution by a running tool. In another less automated environment, the same message could be placed into a user’s mailbox to be handled manually. To complete processing, the designer at the receiving site would have to examine the contents of the message, retrieve its contents, call the appropriate tools and return the resulting data formatted according to the conventions in use.

1.MIME — Multi-purpose Internet Mail Extensions, the extensible Internet standard for formatting electronic messages containing not only text but other types of data.

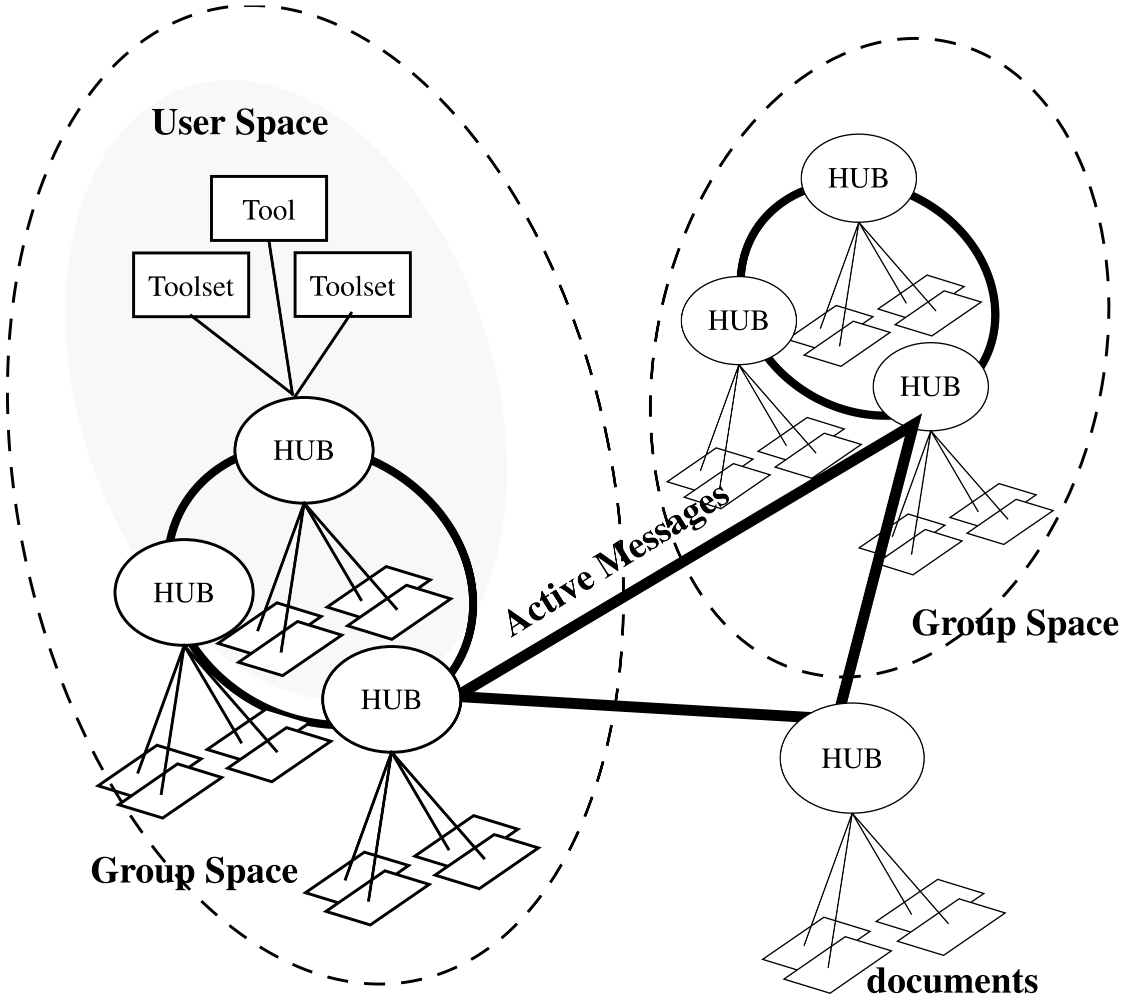


FIGURE 3. The information centric design environment organization. All design information is viewed as a web of active documents, including design files and scripts of commands. Tools are used to manipulate documents. Documents are packed into *active messages*, containing data and operations to be performed on the data by the receiver of the message. Active messages are exchanged via specialized message servers, called HUBs. HUBs stand between document manipulation tools and the information web. HUBs can be setup to manage the exchange of information between the ensemble of tools run by a user or between groups of users. The format of active messages and the message exchange protocols used are defined by Internet standards. We use MIME for active message representation and HTTP as the message transport protocol.

5.1.1 Conceptual Model for Message Handling

The format of active messages exchanged between HUBs and the conceptual model for message activation on delivery that we adopted is based on Enabled Mail (EM) [4]. EM extends the MIME format and the conceptual model for processing electronic mail in the Internet to support active mail systems. However, there is a significant difference in the paradigm used to transport active messages in Henry and active mail systems.

The former uses HTTP, the client-server protocol used in the WWW, while the later uses SMTP, the Internet mail transfer protocol. Henry uses a RPC-based protocol to “pull” information from information servers, while electronic mail is designed to “push” information to receivers of information.

Enabled Mail assumes an environment for activating messages consisting of two interpreters of the Tcl language [11]. One runs Safe-Tcl, a restricted sub-set of the commands of the Tcl language, while the other fully supports it. The former operates as an untrusted interpreter that evaluates the commands embedded in incoming active messages; it has no access to any system resources. To do anything meaningful, it has to send commands to the latter. This is programmed to execute only pre-defined commands that give access to system resources under user control.

We have extended Enabled-Mail by adding new commands to the Safe-Tcl language. The new commands define an interface to access a library of message handling functions for sending commands to design and documentation tools. In the Henry System, the HUB also runs the two interpreters required in the Enabled Mail model. The message handling functions that communicate with the tools run in the trusted Tcl interpreter. On the other hand, the scripts embedded in active messages run in the untrusted interpreter (See Figure 4).

5.1.2 Operation of the HUB

The HUB is structured in software layers, as is common in communications systems. There are two main layers, (1) the Message Transport Layer (MTL) and (2) the Message Handling Layer (MHL).

The functions that support the operations of converting MIME messages to commands and data objects, as well as those for evaluating the active part of messages and associated handlers, constitute the Message Handling Layer. The Message Transport Layer, consists of the functions that perform the low level interface to start the tools and send them the commands and data [14].

The interface between the two layers is defined by a new Tcl command, *hmessage,* used to call the operations that can be performed on every tool. The general form of the *hmessage* command is

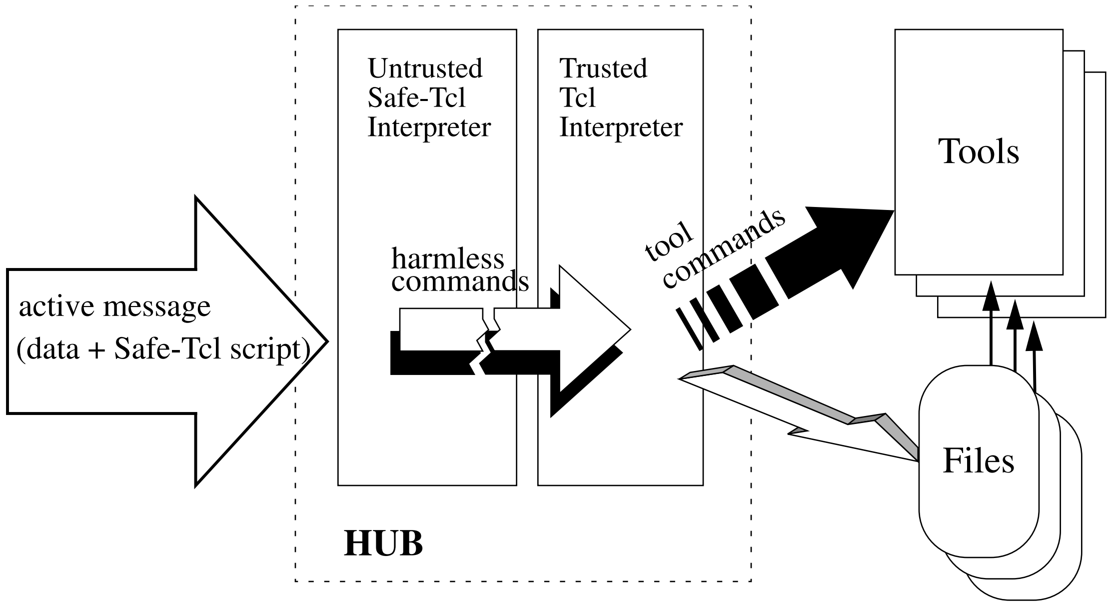


FIGURE 4. Message Handling in Henry. We use the Enabled Mail conceptual model for Message Handling. The Safe-Tcl commands embedded in active messages are evaluated in an untrusted interpreter. This interpreter cannot access any system resources, only the information contained in the message. When part of the contents of a message needs to be saved into a file or a command has to be sent to a tool, the unsafe environment has to use the commands available in the trusted Tcl interpreter. These only give access to compromising system resources after prompting the user.

hmessage tool-address operation operation-parameters

where the *tool-address* field is a 3-element list containing (1) the name of the application to which the message is directed, (2) the Internet address of the user running the tool, and (3) the display where the tool should run.The *hmessage* command defines an essential interface in the HUB architecture. It has two major roles:

1. Defines the point of transition between the tool independent message handling software and tool-specific message processing.
2. Defines the point of transition between the untrusted execution environment for active messages and the trusted environment. A wrapper that prompts the user for confirmation before executing an *hmessage* is available for execution from Safe-Tcl untrusted interpreters used to evaluate active messages.

The importance of designing a common interface to abstract the tools at this level is also an essential aspect of the Henry architecture. In a system comprising heterogeneous tools, it becomes necessary to find a common framework for supporting the different command syntaxes used by the tools. For instance, to read a file into an application's address space, we observe that SPICE3 uses the command *source*, whereas Magic uses *load* and FrameMaker *open*. We have identified the common operations supported by the tools to which we interface. These are listed in Table 1. By creating a uniform syntax to invoke these common operations, we

Command	**Function**
ping	check if a tool is running
start	send the ping message to a tool and start it if no answer is received
open <object>	send the start message and open, source or load the object given as argument
do <command>	perform the command in the tool's command language syntax. This provides the "escape" function to execute any tool specific command not offered by this interface.
quit	terminate execution of the tool

TABLE 1. Common operations supported by the tools integrated with the HUB

make the tool interface uniform to higher layers of software. This uniformity also makes it easier for integrators to create hyperlinks to tools that have different command language syntaxes and terminologies.

5.2 Henry as an Open Hypermedia System

From a designer's perspective, the Henry System operates as follows:

1. The designer selects a piece of design related-information;
2. When he activates the selection, he sees a list of descriptors for other pieces of information. These are related to the object upon which the operations are being performed.
3. Activation of one of the operations, launches the invocation of another tool. The new tool fetches and/or generates other pieces of information.

A similar type of interaction is already used with some combinations of tools by VLSI designers. For instance, there are commercial versions of integrated simulation systems containing a schematics editor, waveform displayer and circuit simulator. In these systems, a user can select a net on the editor and then request the waveform displayer to show the last simulated signal associated with the net. Our goal is to generalize this interaction, so that users can define and associate multiple actions with any design object, select one and invoke it. In the same example, we would like to extend the schematics editor menu with the operations that can be applied to a net. The new operation would open a document describing the circuit in the section that specifies the function of the associated signal. In Hypermedia terminology, we call these operations *live link* activations. We use this term because they do not simply cause the display of other information, but rather send an arbitrary command to a running tool.

Our goal for the Henry environment is to create a framework where hyperlinks are as easy to do as cut and paste within personal computer software. This has to be achieved in a heterogeneous environment where each application is developed using a different set of user interface and inter-tool communication libraries. In Henry, live links exploit active messages to define the actions and the link anchors. These are sent between applications using the HUB services. This interpretation of open hypermedia merges very well with the concept of an information centric design environment.

6. Implementation of the Henry System

The existing prototype of the Henry System consists of a set of design and documentation tools that communicate with a an initial implementation of a message HUB. Some of the tools had to be modified to communicate with the HUBs. These incorporate a collection of inter-tool communication interfaces for sending and receiving commands. Henry already contains a diverse collection of commonly used design and documentation tools. These include,

- FrameMaker, a documentation processing system with hypertext support.
- Magic, a VLSI layout editor.
- SPICE3, a circuit simulator which is linked to *nutmeg*, a front-end for waveform displaying.
- GNU Emacs, an extensible text editor. GNU Emacs runs also as a front-end to a very sophisticated software development environment.

- VEM, the front-end to the Octtools VLSI Design Framework.
- The tools developed at the NCSA to interface with the WWW, Mosaic and the *httpd* server.
- New tools we wrote to support integrated design and documentation.

This list gives a good coverage of the different types of interactions performed by current system designers. It includes tools used for information retrieval, software development, integrated circuit layout and simulation, and documentation. We believe that other tools addressing design aspects not covered by these, such as logic synthesis and printed circuit board design, use fundamentally the same types of interactions and could be integrated in similar ways.

6.1 Interface with the WWW

The organization of the software in Henry's implementation of the interface with the World Wide Web does not follow the organization suggested by the system architecture we described. However, from the functional point of view, it appears to designers as such. For instance, our HUBs do not run two interpreters in a single process. When an active message is received, it is parsed in a separate process that runs the existing Safe-Tcl software. If a tool command has to be executed, the Safe-Tcl interpreter passes the associated message to the safe interpreter, which forwards it in turn to the HUB process. From there, the message is then dispatched to the final destination (see Figure 5).

In the Henry design environment there are two gateways for communication with external services. One is based in electronic mail. The other uses the client-server protocol of the WWW. We describe their implementation in the remainder of this section.

6.1.1 Active Messages Transported by Electronic Mail

To send an active message by electronic mail from a tool, a user of the system presses a button or highlighted text in one of the tools. This has the effect of sending a *hmessage* to the HUB. As the HUB runs a Tcl interpreter, it is straightforward to send a file to another user. The following *hmessage* would do it:

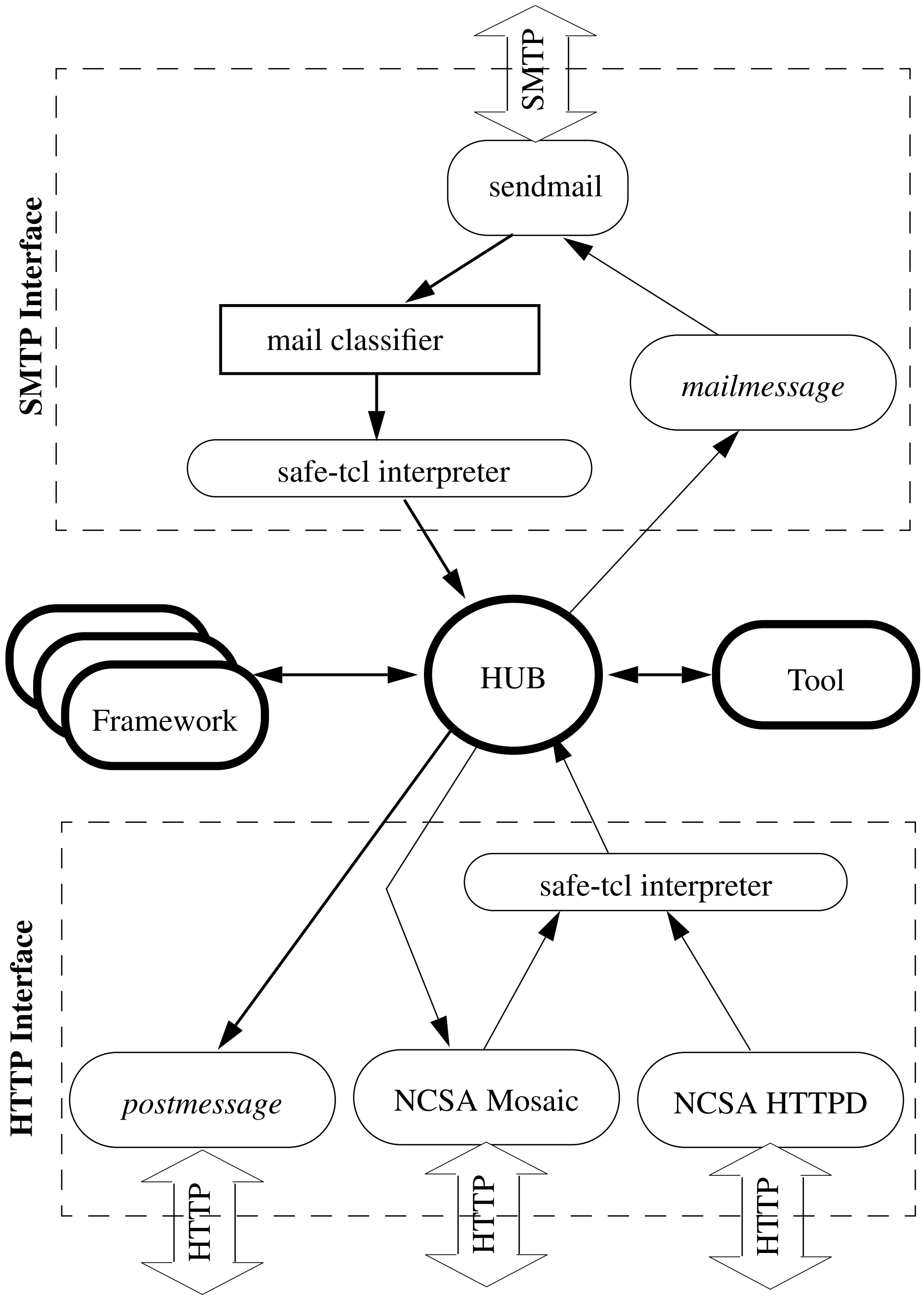


FIGURE 5. The software modules used to communicate with external HUBs and Internet-based services. There are two interfaces based on two Internet protocols. One uses SMTP, the other HTTP. Postmessage and Mailmessage are scripts that we wrote to send active messages from the tools through the HUB using these interfaces.

hmessage HUB [exec mail user@host < file]

We also have *mailmessage,* a script that we wrote to generate active messages from *hmessages* and send them by electronic mail. *Mailmessage* generates a MIME message with the files indicated as arguments, and hmessage commands to the tools to operate on them at the destination. Then, it pipes the resulting active message to *sendmail*, the UNIX program to send mail over the Internet.

To deliver active messages received by electronic mail in the Henry environment, a user needs to configure his mail agent program to automatically dispatch these to the Henry tools. This is achieved through a mail classifying program[1]. HUBs are assumed to be running while the associated users are in session. If a HUB to which a message has to be relayed is not running at the time of delivery, the mail classifier simply places the message into a special folder. The message can then be read and possibly re-activated at a later time.

6.1.2 Active Messages Transported by the WWW Protocol

The sequence of operations for sending an active message using the WWW interface is similar to the one used to send it via electronic mail. The difference is that in the present case a different script, called *postmessage*, is used. While *mailmessage* takes a user's electronic mail address as argument, *postmessage* takes the URL of a remote program to receive a process the document. *Postmessage* spawns a sub-process running the *telnet* program which in turn connects to the WWW server of the URL and sends the generated active message using the *POST* command of the HTTP protocol.

Users may receive active messages from the HTTP interface in three ways:

- As a reply to posting in a URL using the *postmessage* script. HTTP servers in general return a HTML document with information about the result of the execution of the commands they receive.
- As a reply to retrieving the contents of a URL when using Mosaic to browse the WWW. Mosaic can receive commands to get the contents of URLs not only from its the user interface but also from other tools, via its HUB interface.
- Through the HTTP server running in their environment. In this case, we adapted the same Enabled-Mail support software used to dispatch active messages received by electronic mail to a HUBs to interface with NCSA's *httpd* server.

1.There are several programs available for this purpose. An example is *slocal*, which is part of the MH mail handling system [12]

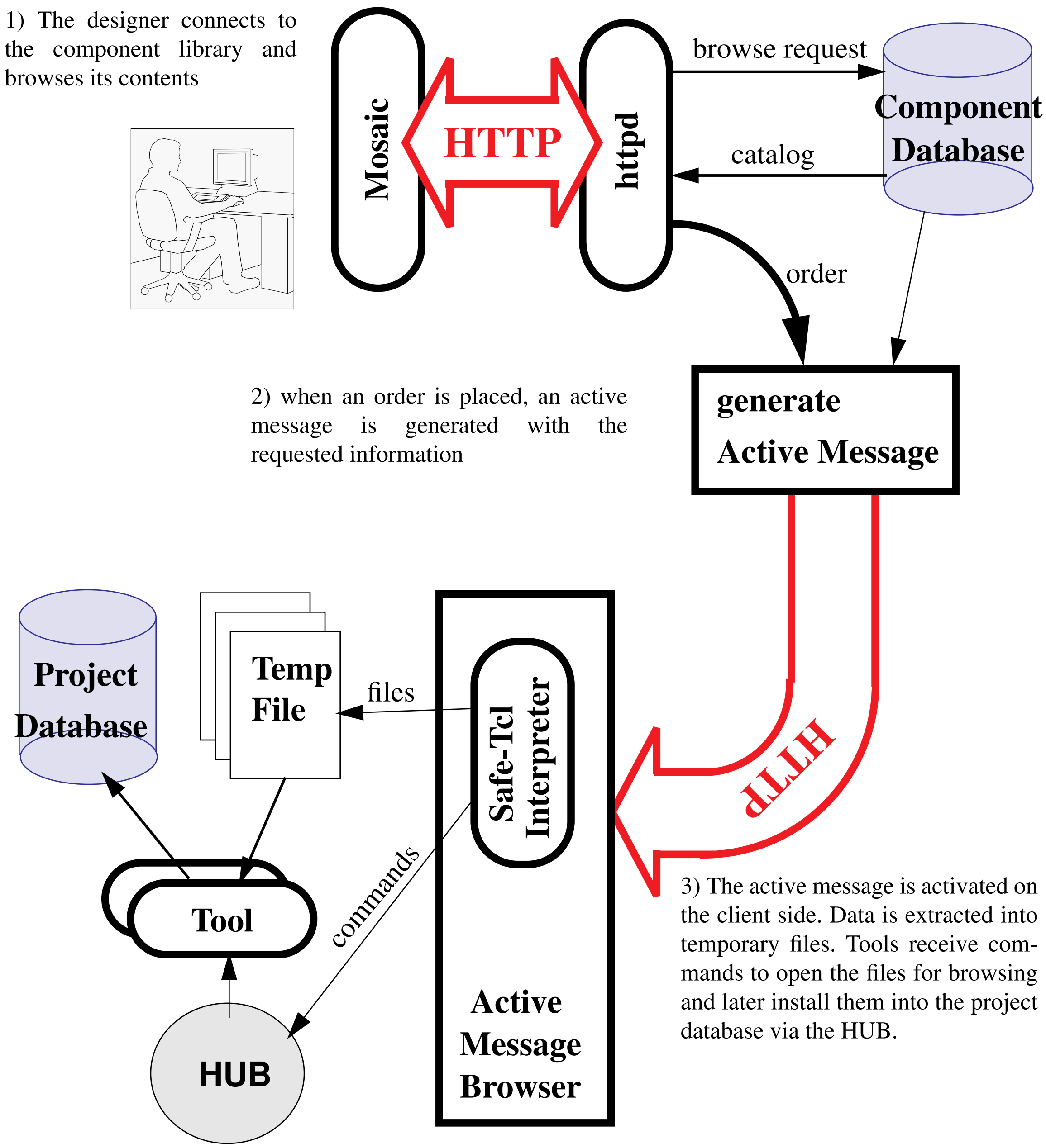


FIGURE 6. Information Flow in Transactions with the Component Library. The figure shows the flow of information between the Henry design environment and an Electronic Component library, from selection and ordering to installation into the local project database.

6.2 Component Library Scenario Implementation

In the Henry prototype, the component library runs the NCSA *httpd* WWW server. Clients access it using Mosaic and retrieve the design information as active messages (see Figure 6 for a diagram with the information flow). The component catalog and order forms are written in HTML, the WWW document format, derived from SGML. The library is simulated with directories containing different implementations of various class projects in CMOS technology. The designs and associated documents were produced by the students of a VLSI design course who used an initial prototype of the Henry System. Each directory contains files in various formats, including FrameMaker documents, Spice3 simulation decks and Magic layouts. As a result, each project's information package is an active document, with files of various types containing hyperlinks between them. The active message with the information for a component is formatted as an Enabled Mail message containing 1) a MIME composite message, whose elements are the individual design data and documentation files to be installed and 2) a Safe-Tcl script.

The advantages in terms of speed and work required to retrieve this information in an environment where this setup could be in real use are obvious. Once the standard protocols and appropriate tools are in place, we can replace paperwork and many commands in a large number of tools with a few button-clicks and the filling of an electronic form.

One way to protect information in a component library, while giving designers the possibility to use it, is by supplying clients the component's interface specifications and documentation along with an authorization to use a remote simulation service, such as the one briefly described above. Only the simulation service has access to the detailed model and this can be set up to restrict its use to simulations submitted from authorized clients. This way, designers can see how their systems would work with a component without actually having access to its model.

The implementation effort to prototype this scenario using the software of the Henry System was rather small, around two weeks. During the implementation process, the main limitation we encountered was the unavailability of a tool capable of sending a large file, such as a simulation request, to a HTTP server. This is not possible with the current version of Mosaic. The form-based user interface of Mosaic when user input request is required is also somewhat limited. In our view, this is one argument for organizing design systems as an ensemble of tools capable of accessing the WWW instead of having one single tool that centralizes all the data presentation and communications with Internet services.

7. Conclusions and Directions for Future Work

Electronic commerce networks will soon be a reality for a large number of electronic companies. New services will be offered to electronic system designers, based on the Internet standards. It will also radically change many of the functions currently performed by designers, such as collaboration between design groups, component procurement, and CAD systems and services outsourcing.

Based on our experience with the Henry System, we advocate using an open hypermedia based architecture for the new generation of design systems integrated with electronic commerce networks. The existing implementation however would have to be extended and improved to provide real services to electronic designers. Some of the HUB's message exchange services could be implemented using Tooltalk, the COSE standard for inter-tool communication adopted by CFI. This would also make integration with commercial tools and Frameworks more easy. Henry, currently does not include software for handling standard CAD interchange formats, encryption and authentication, or to process electronic orders.

Many issues still need to be addressed before electronic commerce is widely accepted among designers. Although our experience tells us that it is possible to add the capability to exchange commands between most design and documentation tools, we are still far from being able to offer an open hypermedia system for electronic design with a single consistent and easy to use interface to all the tools. We anticipate a whole new generation of CAD tools and automated design methodologies for design process involving multiple frameworks and the use of information available to designers on the Internet. Interaction with design systems needs to be based on a new paradigm supported on an information centric user interface. In particular, design management tools will need to be adapted to interact with designers through active documents and manage the design data and process based on the messages exchanged between the tools.

Although good privacy and authentication can be provided with current software, these are only the initial security issues that need to be addressed. Currently most design environments are maintained behind firewalls that isolate their databases from the Internet. Providing really useful services to designers will imply moving the information in those databases into publicly accessible places. However, trustability of design services seems to be much harder to ensure. In a simulation service as the one described, the simulation files could easily be duplicated and made accessible to a third party without knowledge from the client. Separating the design process into independent design services' providers may be much harder to implement than the separation from design and manufacturing of VLSI circuits we observe today. There is incomparably much more knowledge in a VHDL simulation model of a system than in the fabrication masks.

8. Acknowledgment

We thank Prof. Jan Rabaey and Ole Bentz for the interesting discussions and ideas for this project. Jan Rabaey coined the term *information centric design*.

References

[1] ATLIS Consulting Group, Inc., 6011 Executive Boulevard, Rockville, MD 20852, USA. *Pinnacles Component Information Standard*, March 1994.

[2] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The World Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.

[3] N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. Technical report, Bellcore, Innosoft, September 1993. Internet RFC 1521.

[4] Nathaniel Borenstein. Email With a Mind of its Own: The Safe-Tcl Language for Enabled Mail. Submitted to Proceedings of ULPAA'94, 1994.

[5] Nathaniel S. Borenstein. Computational Mail as Network Infrastructure for Computer-Supported Cooperative Work. In *CSCW'92 Proceedings*, pages 67–73, November 1992.

[6] Patrick W. Brown. Digital Signatures: Are They Legal for Electronic Commerce? *IEEE Communications*, 32(9):76–80, September 1994.

[7] CAD Framework Initiative, Inc., 4030 W. Braker Lane, Suite 550, Austin TX 78759. *CFI Architecture Revision*, version 0.06 edition, March 1994.

[8] William H. Davidow and Michael S Malone. *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century*. Harper Collins Publishers, New York, 1st ed. edition, 1992.

[9] International Organization for Standardization. *Information Processing, text and office systems, Standard Generalized Markup Language (SGML)*. International standard 8879. International Organization for Standardization, Geneva, Switzerland, 1st edition, 1986.

[10] Robert Neches, Anna-Lena Neches, Paul Postel, Jay M. Tanenbaum, and Robert Frank. Electronic Commerce on the Internet. Technical report, USC/Information Sciences Institute, May 1994.

[11] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[12] M. T. Rose and J. L. Romine. *The Rand MH Message Handling System: User's Manual*. Department of Information and Computer Science, University of California, Irvine, January 1985.

[13] Mário J. Silva and Randy H. Katz. Active Documentation for VLSI Design. In *30th ACM/IEEE Design Automation Conference*, pages 654–660, 1993.

[14] Mário J. Silva and Randy H. Katz. Ensembles of Interactive Tools and Hypermedia Documents in VLSI Design. In *Proceedings of the 1994 Electronic CAD Interoperability and Integration Conference, EII'94*, pages 23–32, Oakland, CA, May 1994.

[15] SunSoft. The ToolTalk Service. Technical report, SunSoft, Inc., SunSoft, Inc, 2550 Garcia Avenue, Mountain View, CA 94043, October 1992.