EDA and the Network

Mark D. Spiller and A. Richard Newton
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

Abstract

Digital computer networks are playing an increasingly important role in the evaluation, distribution, integration, and management of EDA systems. Tools, libraries, design data, and a variety of both design and manufacturing services are accessible today via networks. Networks are also playing a central role in the integration of system design teams, teams that involve a variety of both business and technical disciplines as well as widely distributed geographical locations. Throughout the history of EDA, the architectures used to integrate and distribute computation and interaction have played a central role in the overall design methodology and so have had a major, indirect impact on the choice of the most effective tools, algorithms, and data structures. In this paper, a number of the factors involved in the choice of a suitable architecture for EDA integration are reviewed and a number of ongoing developments and challenges are presented.

1. Introduction

From the very earliest days of Electronic Design Automation (EDA) more than thirty years ago, the particular *computing infrastructure* available to chip designers, EDA tool developers, and EDA system integrators has played a key role in determining overall chip design methodology. In so doing, it has also played a major role in establishing the resulting tool architectures and algorithms used to implement design systems.

By computing infrastructure we mean all of the general-purpose facilities, both hardware and software, used by EDA professionals and designers to do their job. Today, these include operating systems, storage systems, programming languages and development environments, user interaction technologies in both hardware and software, and, of course, computer networks.

Over the past three decades, there have been a number of significant changes in computer infrastructure and every change has brought with it a significant change in research emphasis as well as to the EDA industry. Computer-Aided Design (CAD) programs were originally designed for batch-input, line printer output (e.g. Spice[1]), and formed the basis of the CAD tool industry. With the advent of multitasking operating systems and direct inter-program communication (e.g. Multics[2], Unix[3]), along with "personal minicomputers" (i.e. workstations), the concept of the *software tool* was introduced to electronic system design and the EDA industry was established in the early 1980s. We have seen the impact of similar changes in user interaction technologies. For example, symbolic layout entry moved from character-based CRT approaches[4], to stick diagrams on vector displays[5], to today's raster fill abstractions (e.g. [6,7]). In each case, the layout methodology that could be expressed by the display hardware and associated software ultimately determined the layout data structure and most suitable compaction algorithm.

This infrastructure is always evolving and at certain times the changes are so profound that they have an almost revolutionary impact on EDA. We believe that the explosive growth in the development of wide-area network infrastructure over the past few years and into the next century, especially the Internet, along with significant developments in user interaction hardware and software (e.g. high-performance video and 3D graphics technologies) can fuel another revolution in the way EDA systems are organized and so how the industry itself is structured. We say "can" rather than "will" because history has demonstrated that the ultimate outcome of such a change is also a function of where industry and government choose to spend their research and development dollars. The way in which the EDA, semiconductor, and chip design industries are organized and relate to one another *will* find another "fixed-point" in the future--the changes are already apparent. However, the particular form of that structure and its overall technical as well as business efficiency will be a very strong product of the attention the industry pays to the rapid changes in infrastructure occurring around it and

how it chooses to invest in taking advantage of these developments.

We envisage the *entire EDA community organized as a single, integrated, distributed environment that is continually evolving and adapting*, much as the basic Internet infrastructure is organized today. Groups may choose to publish or protect particular aspects of their work but *the overall community is fully integrated at the service layer*, facilitating a tight coupling among EDA vendors, their customers, and research communities throughout the world. With appropriate investments in basic engineering and EDA software services, EDA users should be able to *construct their own* "virtual" EDA systems that couple tools, libraries, design and validation services, as well as manufacturing, component acquisition, and product distribution, from internal developments, other companies, universities, and individuals throughout the world. Globally-distributed product creation teams consisting of hundreds or perhaps thousands of professionals, from many different organizations and disciplines, from engineering to marketing and sales, should be able to collaborate closely on complex design tasks and bring new products to market quickly and reliably.

One way of organizing the major aspects of network and visualization technologies are shown in Figure 1.

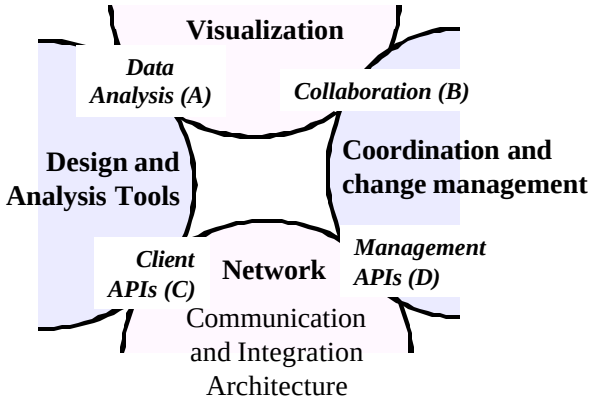

Fig. 1: Major Aspects of an Integration Architecture

New visualization hardware, including both input as well as output technologies, and associated software support can deliver significant advances in our ability to visualize complex data sets and their evolution (A) as well as aid in the management of collaborative work distributed over geographic space as well as time (B). However, it is the network and communication architecture that will play the most fundamental role in determining the overall organization of the electronics industry and so we have chosen this aspect as the emphasis of this paper. It is the most fundamental building-block upon which the design, analysis, and service tools are layered, as well as the necessary design coordination and work-flow management systems.

Innovation in the application of the network in EDA, for commercial as well as university research and instruction, is occurring at a very high pace today. For that reason, we have not attempted to record all of the ongoing activities in the area but rather focus on a number of specific architectural issues central to sustaining broad development and growth in the community. Where appropriate, we reference representative work in an area but are not able to refer to all of the excellent ongoing activities, especially on the client side, due to space limitations.

In Section 2, the main requirements of a distributed service infrastructure are defined and a number of architectural approaches to meeting these requirements are reviewed in Section 3. Selected examples of services that are under development today and would benefit from such an infrastructure are presented in Section 4.

2. Network Services

As mentioned earlier, many of the basic ideas behind the transformation of a traditional operating system to an *infrastructure of computing services* were introduced in the Multics project[2], an inspiration for the later Unix developments at Bell laboratories. Unfortunately, that project was hampered significantly by the lack of a reliable high-speed networking infrastructure. As is already apparent in EDA, network-based applications are becoming significantly more popular than their stand-alone desktop counterparts as they are a lot easier to maintain (e.g. for version management, service, use tracking for planning purposes, and platform management). In addition, queuing theory demonstrates that centralized (virtual) serving of application cycles is more efficient in both cost and utilization than a collection of smaller servers, where desktop systems represent the degenerate case of one "server" per user. These arguments are part of the thrust for the adoption of Network Computers[8].

Unfortunately, network-based services remain very difficult to deploy and maintain. Network services can experience a wide range of loads, with very high burst requirements at times, and there are a number of very challenging system design issues that must be overcome in order to maintain a high response time, constant availability, and meet capacity requirements. Such services must be able to adapt well to an ever-evolving environment, with an increasingly varied client population (in terms of software, hardware, and network connectivity). Because of the potentially slow connectivity to clients and lengthy delivery times, coupled with the potentially long transaction times encountered in EDA applications, such services must be

able to handle hundreds of outstanding tasks simultaneously.

If our vision of a single, integrated EDA community is to be achieved, the following challenges must be met:

- **Scalability:** Network services must support exponential growth in the number of users (service providers as well as designers) and servers (tool, information, data storage, etc.)
- **Availability:** Access must be uninterrupted and any service degradation due to catastrophic failure in the infrastructure should be "graceful" and its impact limited in scope.
- **Adaptability:** Users, services, and servers will be added and removed incrementally. The infrastructure must be able to adapt automatically to such continuous changes without the need to maintain a consistent global state for all users.
- **Robustness:** Any failure which may result in loss of data or other aspects of design state must be recoverable efficiently and incrementally. No failure should be able to compromise the security of information stored in the network or in transit.
- **Cost effectiveness:** All of the above attributes must be achieved cost effectively and such that any economies of scale in the network can be exploited efficiently.

In the early part of this decade, EDA systems were built upon a framework model that could be characterized as shown in Figure 2[34].

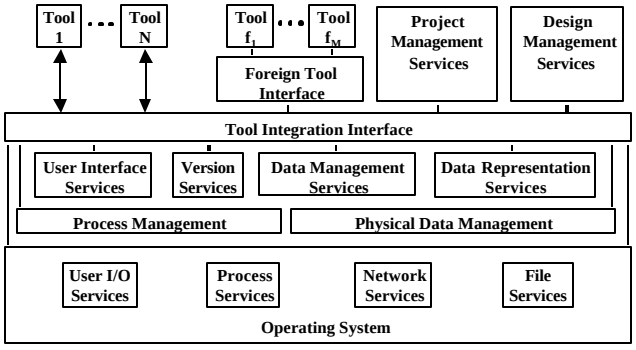

Fig. 2: EDA Framework Model[34]

Operating system services were abstracted as supporting *user interaction, process services*, *local storage*, or *network services*. These facilities were then grouped into those that supported *process management* and those that supported *physical data management*, including distributed data management on a network. This underlying classification of services is well-suited to a multitasking process/communication model as presented by traditional Unix systems, for example, and as used extensively for EDA systems integration today. Other services, usually specialized in some way for the peculiarities of the EDA problem, are shown at the next layer. These include the services related to advanced user interface, design versioning and configuration management, and EDA data representation semantics. *Applications*, whether design tools, project management applications, or users themselves, can apply these facilities to implement their particular task.

When the network is viewed as the central aspect of an overall EDA integration architecture, the major change to this diagram is that the local distinction between process, network, and secondary storage services is eliminated. All of these services can be considered simply as different aspects of the abstraction of services provided via the network.

The two basic components of such a modern distributed architecture are then the *computational nodes*, often referred to today as *clients*, *servers*, or *peers*, depending on their respective responsibilities in the overall system, and the *interconnection network*. For the purposes of this analysis, we define the *service layer* as representing all the facilities used to link these two worlds, as illustrated in Figure 3. For simplicity, we will use the term *client* here to represent any user of the network: human, tool, and information or manufacturing service and assume that most of the more EDA-specific *other services* referred to above are provided as part of a client Applications Programming Interface (API).

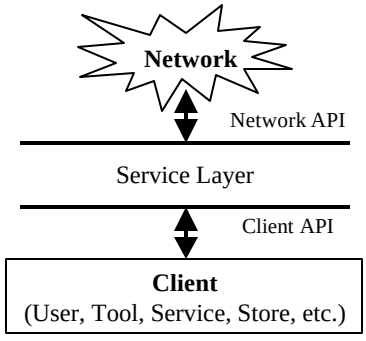

Fig. 3: The Service Layer

The different architectural approaches under development for implementing the infrastructure needed to meet the requirements outlined above can be seen as different approaches to the implementation of this service layer.

2.1 Design Transaction Management

The traditional transactional database world is primarily concerned with the ACID properties (atomicity, consistency, isolation, and durability)[9], essentially providing strong semantics but at very high cost and implementation complexity. As pointed out in [10], ACID makes no guarantee regarding availability and an ACID service would rather be unavailable than relax the ACID constraints.

In the EDA world, there are almost always a very complex set of dependencies to manage: among a design and its sub-components, different views of the design, analysis and synthesis tools, design services, and human collaborators in the design and manufacturing process.

On the other hand, at a number of points in the design process, it is more important to maintain high availability and low latency access to information than to ensure strong consistency or durability of the data. Approximate answers, based on stale information[11] or incomplete soft state[12], that are delivered quickly may be more valuable than precise data delivered more slowly or requiring some form of transitive closure across the entire design state. The semantics resulting from these requirements have been referred to as BASE[10]--Basically Available, Soft State, Eventual Consistency, and we will use this definition here. There is a strong analogy between the application-level needs of BASE systems and the traditional requirements of implementing high-performance software on "bad" multiprocessors (i.e. processors where remote memory access latencies are significantly longer--often orders of magnitude longer--than cache miss latencies on a uniprocessor.)[13]

In the presence of a potentially unreliable and unpredictable network fabric like that of the Internet, any successful EDA service layer will require both ACID and BASE component services. As distinct from the traditional Web-based services of today, where the balance is very much in the direction of BASE requirements, the implementation of distributed design and manufacturing will require a more intimate relationship between these two worlds, both of which are undergoing significant research and development (e.g. [14-21]).

In a Client-Service-Network (CSN) approach, illustrated in Figure 2, the particular characteristics of the service layer in its implementation of both ACID and BASE modes of operation is likely to be application dependent (task domain dependent). Therefore, the design challenge is finding an architecture that can address as broad a range of EDA requirements as possible. The service layer can be thought of as providing a client APIs for the delivery of abstract datatypes (of all types, from video services to software programs and data components) to and from the client. A key requirement of such a standard API is that the abstract datatype handlers implemented in the service layer not change very rapidly over time since such changes are likely to affect many clients. There are generally two approaches in use today to deal with the evolution of such datatypes--either provide *plug-in* datatype support to facilitate evolution of the interface or try to be sure the datatypes served are as application-independent as possible. For example, selecting standard MPEG2 for all video services, no matter what the particular application, would be such an approach. In either case, there is no ideal solution and the mechanisms used to implement the interfaces must be selected with evolution in mind.

3. A Proxy-Based Approach

If the service layer is viewed as the interface between clients and the network as mentioned above, it can also be seen as the client's *agent*, or *proxy*, in negotiations among the various players involved in the EDA virtual network. As such, it plays the important role of orthogonalizing the local needs of the client (user, service, tool, information store, etc.) from the network infrastructure. Such an approach is illustrated schematically in Figure 4.

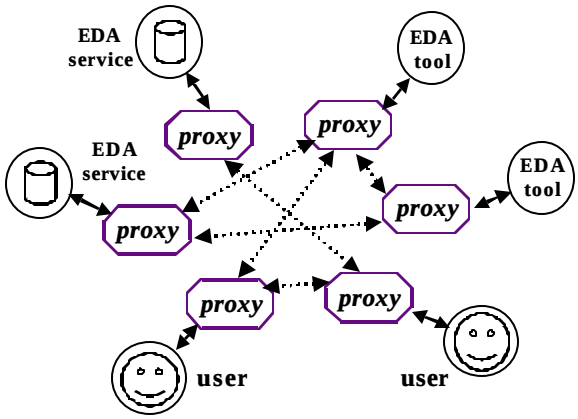

Fig. 4: A Proxy-Based Distributed Architecture

Proxies can implement many roles on the client side in such an architecture. They can play the role of the conventional legacy tool "wrapper"[34] by hiding a custom interface for a particular tool or class of tools on the client side, presenting instead the generic system interface. They can also be used for data distillation[18] or datatype transcoding to provide a better match between the needs of a particular client (a user on a low-speed mobile link, for example) and the virtual server infrastructure the client is using. Finally, proxies can of course play their most common role in use today, that of data hiding and security management.

On the network side, each proxy shares the same view of the world. The negotiations for services and service quality among proxies must demonstrate high availability, security, and reasonable performance for both ACID as well as BASE operations. It is via the network interface that the system must meet its goals of scalability, availability, robustness, and adaptability. This is also an area under active research and one which lends itself to many modern innovations. For example, in the WELD project[42], we are exploring the use of probabilistic learning, based on Bayesian reasoning[40] as an approach to managing aspects of the adaptability and robustness concerns. The implementation of an efficient BASE technology would suggest the use of caching techniques to improve overall system performance. However, significant issues remain in the implementation of effective ACID behavior when distributed caches are implemented in the face of a potentially unreliable physical network link layer. This is a topic of active research as well.

Some approaches that have been suggested for maintaining consistency in the distributed data network include the use of separate "control" and "data" distribution. In one approach, the use of a broadcast mechanism, such as the MBONE[45] has been suggested for control-state management (e.g. "who has the master copy of a dataset?") while data-sets or programs themselves are shipped around via unicast point-to-point mechanisms[43].

Modern VLSI designs consist of working sets approaching 4 gigabytes of data today and designs in the future will require considerably larger working sets, unless the EDA industry finds truly effective ways of implementing partitioned design (via the exploitation of regularity or hierarchy). However, no matter what approach is used, working-set size will remain large if the performance of modern computers and computer clusters is to be exploited effectively. Network performance is increasing significantly in most organizations but there is likely to remain the need to move the tool to the data at times rather than move the data to the tool. One of the clients envisioned in the distributed network described above can be seen as a data repository, requesting service from tools as they migrate around the network. While not yet capable of handling practical EDA tasks, much of the architecture and technology needed to implement such a model has been developed(e.g. [24]).

One interesting architectural approach to these issues is the use of clusters of commodity workstations interconnected by a very high-speed local network[10]. In this approach, service-independent software architectures for managing partial failures and administering large clusters of machines remains a challenge.

A simplified view of the major components of a proxy that might be used in this architecture is shown in Figure 5.

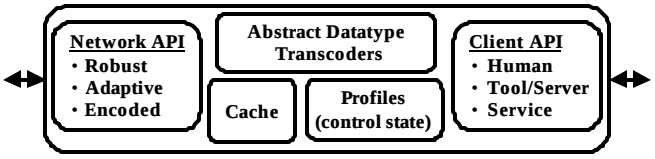

Fig. 5: Schematic Diagram of an EDA Proxy

In this approach, all proxies in the EDA network would be relatively simple and architecturally identical. On the client side, different API's are used to extend the interface for human, tool, data/processor cluster, or service use. On the network side, the API's implement the various criteria associated with network reliability, availability, and security outlined earlier. The proxy would be provided with a number of abstract datatype handlers as plug-ins, would store client-specific information locally in a profile, and would implement a cache. Some experimental systems have been implemented which register the client for interaction with other clients in the network and which implement a "heartbeat" scheme for periodically checking the reliability of communication links[25]. These schemes could be extended to provide an adaptable network fabric that adjusts automatically to problems with computer or network reliability issues.

4. Other Services

As mentioned earlier, it would be impossible to do justice here to all of the ongoing research and development in data visualization, collaboration, and work flow management that involve the network and that have a relationship to EDA. Rather, in the space available we reference a number of the key resources and highlight some important directions for the future.

Once a network infrastructure has been selected, a variety of approaches can be taken to implement client-oriented services, such as data manipulation and querying (e.g.[14-17]), visualization for design and collaboration (e.g.[31-32]), work flow management (e.g.[22,25-27,36]), and library and part management (e.g.[28-30]).

Much of the distributed tool flow and collaboration work to date has been implemented under the X-Windows architecture or, more recently, using Tcl/Tk[37]. An excellent review of these projects and extensive references can be found in [22]. More recent developments utilize the Java programming environment and JavaBeans[44] and a number of efforts are investigating the application of CORBA[46] as well.

From an EDA on the network perspective, the REUBEN system[22] features *user-defined and reconfigurable execution sequences* by creating dependency edges between program nodes (application icons) and file nodes (data icons), *data-dependent execution sequences* by dynamic scheduling of path as well as loop executions, and *host-transparency* as to the location of applications and data (both can reside on any host with a unique IP address). The system is presently written in Tcl and makes use of popular protocols such as telnet and ftp. . There are a number of commercial systems available that already tackle specific aspects of the distributed workflow management task for EDA (e.g.[27-30]).

CREW, The Collaboratory for Research on Electronic Work at the University of Michigan[31], focuses on the design of new organizations and the technologies of voice, data, and video communication that make them possible. The site contains an comprehensive set of on-line references to work on collaboration at many different levels.

Another major project that involves the application of visualization technologies to collaboration is DARPA's Intelligent Collaboration & Visualization (IC&V) program[32] is chartered to enable teams, consisting of individuals with specific roles, and teams of teams to collaborate more effectively through distributed information systems that encode relevant knowledge, expertise, and semantics and that permit multiple, shareable views of common collaborative information spaces. Their Web site contains a number of efforts that could eventually be very useful in the EDA world.

5. Summary

Computing infrastructure will continue to play a much larger role in determining overall EDA tool integration strategies, approaches to data visualization and collaboration, and even the most effective choice of algorithms for tools, than most EDA professionals realize.

With the appropriate investments in research and development over the next few years, we can expect to see significant developments in the *service layer*, as described here. These developments will enable transparent distributed EDA systems that are reliable and secure and that enable distributed collaborative design. Associated developments in visualization technologies, both for data and to support collaboration, are likely to make such environments both efficient and effective for the design and transfer to manufacturing of electronic systems.

Eventually, the operating system as we know it will cease to exist as many of the facilities we describe here migrate closer to the underlying hardware. Recent advances in operating systems and networking both demonstrate this trend and provide the technical underpinnings for another step forward. In particular, work in distributed file systems (e.g. [39]) has succeeded in providing efficient location-transparent access to data. For a glimpse into one possible future here, see [33,38].

No matter what direction we head, it is clear that electronic system design is poised to benefit early from these revolutionary developments in the use of distributed computing infrastructure. The EDA systems in use a decade from now will bear virtually no resemblance to the systems in use today.

6. Acknowledgements

Our work in Web-based electronic design is supported in part by DARP under Contract DABT63-95-C-0074 and by the Digital Equipment Corp., Hughes Research, Hewlett Packard, and Synopsys. Their support is gratefully acknowledged. We would also like to thank Wray Buntine, Francis Chan, Andrew Mayer, Michael Shilman and the members of the WELD team for their contributions to this work.

7. References

[1] L. W. Nagel and D. O. Pederson, "Simulation Program with Integrated Circuit Emphasis," *Proc. 16th Midwest Symp. Circ. Theory*, Waterloo, Canada, April 1973.

[2] F. J. Corbato and V. A. Vyssotsky, "Introduction and Overview of the Multics System," *AFIPS Conference Proceedings*, No. 27, pp.185-195, Fall Joint Computer Conf., 1965. (At http://www.lilli.com/fjcc1.html.)

[3] The *Bell System Technical Journal, Special Issue on UNIX*, Vol.57, No.6, Jul-Aug., 1978, contains a collection of papers describing UNIX.

[4] D. Gibson and S. Nance, "SLIC - Symbolic Layout of Integrated Circuits," *Proc. 13th ACM/IEEE Design Automation Conference*, pp.434-440, June 1976.

[5] J. D. Williams, "STICKS - A Graphical Compiler for High-Level LSI Design," *AFIPS Conference Proceedings*, Vol.47, pp.289-295, June 1978.

[6] N. Weste, "Virtual Grid Symbolic Layout" *Proc. 18th ACM/IEEE Design Automation Conf.*, pp.225-233, 1981.

[7] J. Ousterhout, et al., "Magic: A VLSI Layout System," *Proc. 21st ACM/IEEE Design Automation Conf.*, pp.152-159, June 1984.

[8] T. R. Halfhill, Inside the Web PC, *Byte Magazine*, pp. 22-36, March 1996

[9] J. Gray, "The Transaction Concept: Virtues and Limitations, " *Proc. VLDB*, pp.144-154, Cannes, France, Sept. 1981

[10] A. Fox, et al., "Extensible Cluster-Based Scalable Network Services," *Proc. 16th ACM Symposium on Operating Systems Principles (SOSP-16)*, St. Malo, France, Oct. 1997.

[11] A. Demers, et al., "The Bayou Architecture: Support for Data Sharing Among Mobile Users," available at www.parc.xerox.com/csl/projects/bayou/pubs/ba-mcw-94/www/MobileWorkshop_1.html

[12] D. Clark, "Policy Routing in Internet protocols," *Internet Request for Comments No.1102*, May 1989.

[13] A. D. Birrell, et al., "Grapevine: An Exercise in Distributed Computing," *Comm. of the ACM*, 25(4), Feb. 1984

[14] See *Infoscape*, at http://www.infoscape.com

[15] See *NetDynamics*, at http://www.netdynamics.com

[16] See *ObjectStore*, at http://www.odi.com.

[17] See *Objectivity Version 5* at http://www.objectivity.com.

[18] A. Fox, et al., "Adapting to Network and Client Variability via On-Demand Dynamic Distillation," *Proc. ACM 7th International Conf. on Architectural support for Programming Languages and Operating Systems*, Cambridge, MA, Oct. 1-5, 1996.

[19] *The Distributed Clients Project*, can be found at http://www.osf.org/www/dist_client.

[20] A. D. Joseph, et al., "Rover: A Toolkit for Mobile Information Access," *Proc. 15th ACM Symposium on Operating Systems Principles*, Copper Mountain, CO., Dec. 1995

[21] M. Abrams, et al., "Caching Proxies: Limitations and Potentials," *Proc. 4th International World Wide Web Conf.*, Boston, MA, Dec. 1995

[22] H.Lavana, et al., "Executable Workflows, "A Paradigm for Collaborative Design on the Internet," *Proc. 34th ACM/IEEE Design Automation Conference*, June 1997. Also available at http://www.cbl.ncsu.edu/publications/

[23] Javaworld online magazine contains many useful Java references at http://www.javaworld.com

[24] See *Marimba, Inc*. at http://www.marimba.com

[25] M. Spiller, "Architecture and Infrastructure for a Distributed Design Environment--A Server Perspective", MS Report, University of California at Berkeley, August 1997

[26] F. Chan, "Architecture and Infrastructure for a Distributed Design Environment--A Client Perspective," MS Report, University of California at Berkeley, May, 1997

[27] See *Runtime Design Automation*, at http://www.rtda.com

[28] See *Synchronicity* at http://www.syncinc.com

[29] See *Electronic Design and Technology Network* (EDTN) at http://www.edtn.com

[30] See *Aspect Development, Inc*., at http://www.aspectonline.com

[31] See *CREW, The Collaboratory for Research on Electronic Work* at www.crew.umich.edu,

[32] See *DARPA's Intelligent Collaboration & Visualization program (IC&V)* at http://www.ito.darpa.mil/ResearchAreas96/IntellCollabVisual.html

[33] William J. Bolosky, et al., "Operating System Directions for the Next Millennium," *Position Paper*, available at http://www.research.microsoft.com/research/os/Millenniu m/mgoals.html

[34] D. S. Harrison, et al., "Electronic CAD Frameworks,". *Proceedings of the IEEE*, Vol.78, No.2, pp.1062-1081, Feb. 1990.

[35] J.Daniell and S.W. Director, "An Object Oriented Approach to CAD Tool Control Within a Design Framework," *IEEE Transactions on Computer-Aided Design*, Vol.10, No.6, pp.98-713, Jun. 1991.

[36] A. Casotto and A. L. Sangiovanni-Vincentelli, "Automated Design Management Using Traces," *IEEE Trans on Computer-Aided Design*, Vol.12, No.8, pp.1077-1095, Aug. 1993.

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

[38] Amin Vahdat, et al. "Turning the Web Into a Computer," available at http://now.cs.berkeley.edu/WebOS/webos.ps, 1996.

[39] Kistler & Satya 92] J. J. Kistler and M. Satyanarayanan. "Disconnected Operation in the Coda File System." *ACM Trans. on Computer Systems*, Vol. 10, No.1, pp.3–25, Feb. 1992.

[40] Special Issue on Bayesian Networks: *Comm. of the ACM.*, vol. 38, no. 3,.March, 1995,

[41] P. van der Wolf, *CAD Frameworks: Principles and Architecture*, Kluwer Academic Publishers, Boston, 1994

[42] *WELD: Web Based Electronic Design* can be found under www-cad.eecs.Berkeley.edu/Respep/Research/weld, March, 1996.

[43] E. Brewer, University of California at Berkeley, Private Communication.

[44] A number of excellent references for JavaBeans can be found at http://splash.javasoft.com/beans/.

[45] An excellent introduction to the MBONE, "MBONE (Multicast Backbone)" by Jean Bunn, Geneva University, is available at http://www.unige.ch/seinf/mbone.html

[46] See "The Object Management Group's Common Object Request Broker Architecture (OMG/CORBA)" at http://www.acl.lanl.gov/CORBA/ for more details and an extensive list of references.