EDA and the Net

Prof. A. Richard Newton

Department of Electrical Engineering and Computer Sciences

University of California at Berkeley

Presentation available at:

http://www-cad.eecs.Berkeley.edu/~newton

ICCAD Tutorial 1997
San Jose, CA
11 Nov 1997

Outline

- Semiconductors and EDA
  - How did we get here?
  - Where are we going?
- Networks and the Internet
- The Opportunity
- Distributed Computing: The Future of the OS
- The Role of Java
- Visualization as a Critical Technology
  - The design and associated algorithms
  - The process of collaboration

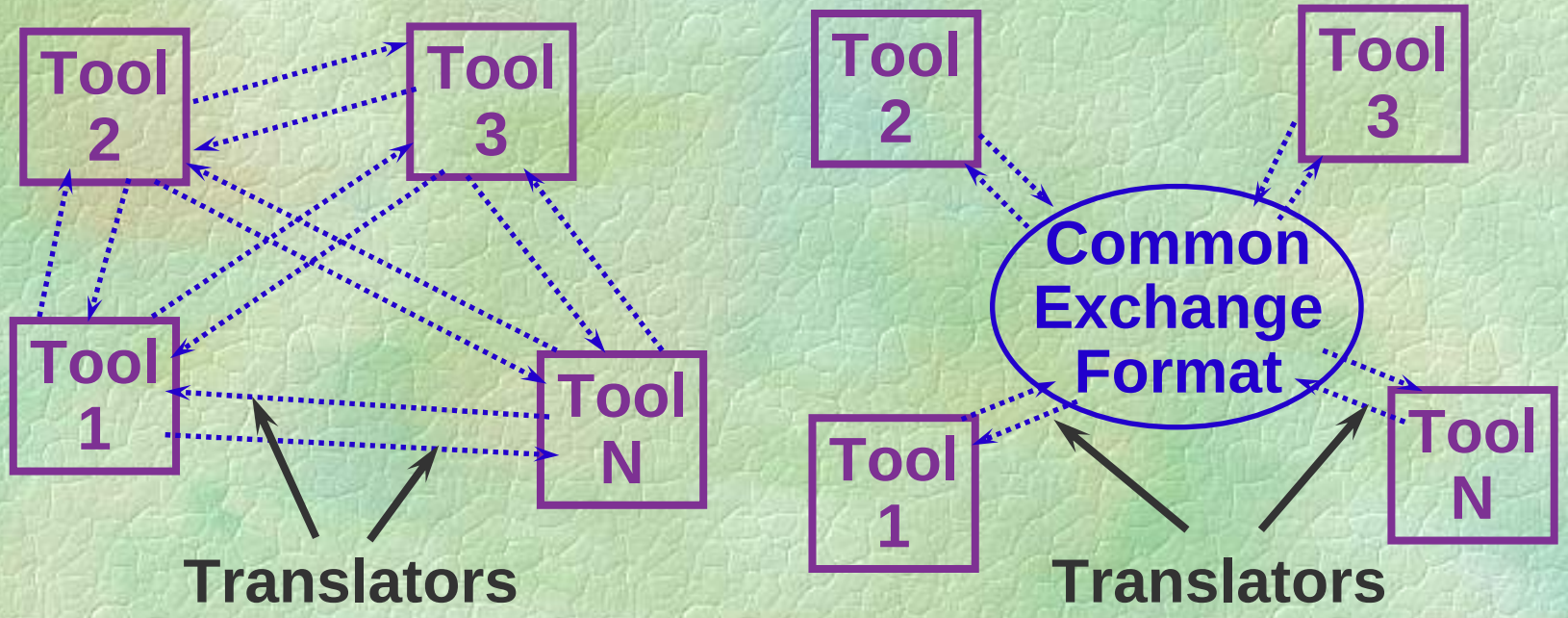
Semiconductor Industry Growth

	1995	Steady State? 2030
Semiconductor as % of Electronics	17%	35%
Electronics as % of GWP	4%	8%
Semiconductors as % of GWP	0.7%	3%
CMOS Technology	0.35μm	0.05μm
World Semiconductor Sales	$140B	$12,000B
Annual Growth Rate	16%	8%, same as GWP

Source: Prof. Chenming Hu, UC Berkeley, 1996

1970's: The First Generation
A *Tool-Centric* view of CAD

- Single tools were the focus (Spice, DRC, CVS, etc.). task-specific formats evolving to point standards (CIF, Stream, Spice, TDL) by the end of the decade.



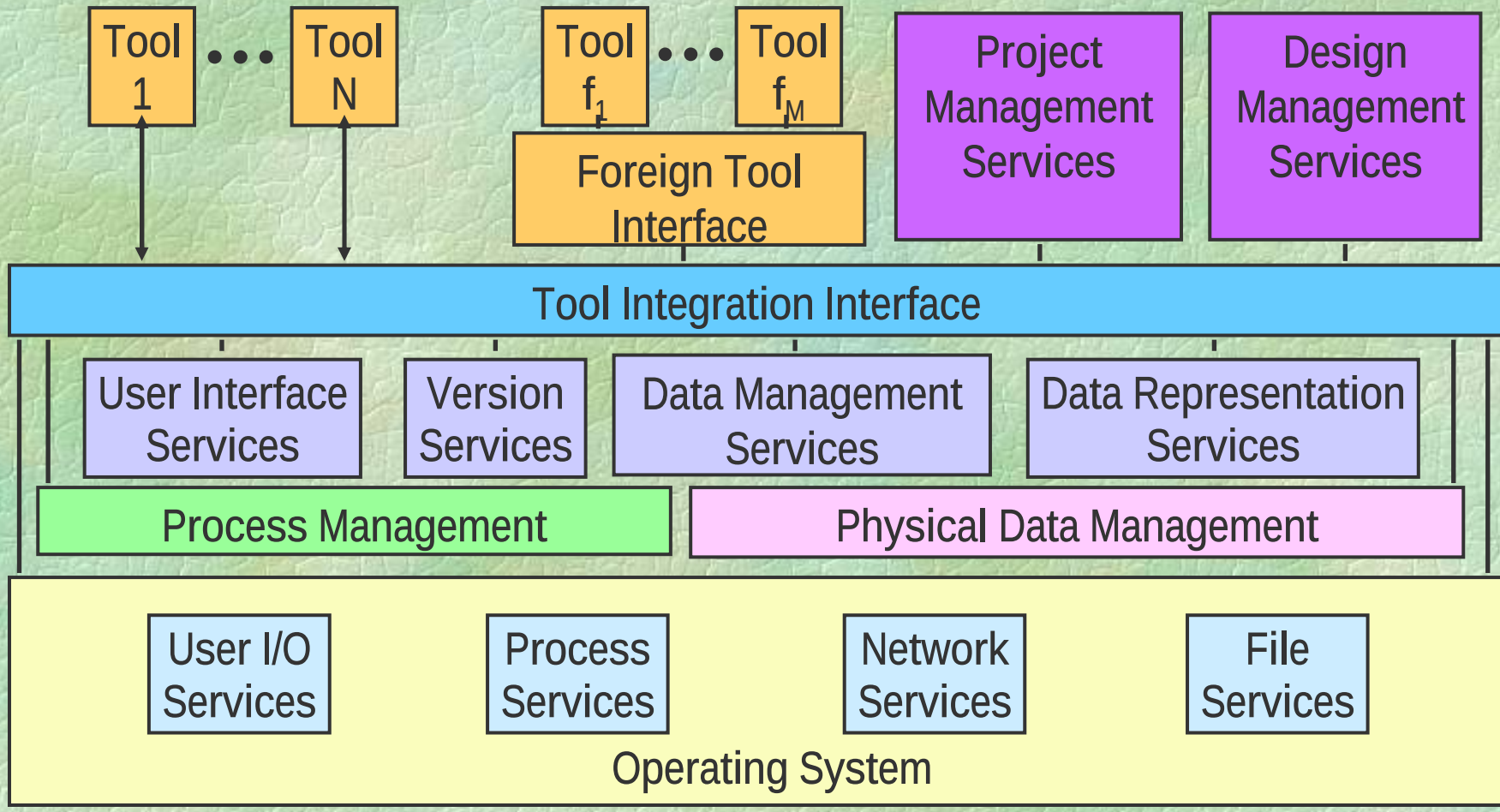
1980's: The Second Generation
An *Integration-Centric* View of CAD

- Integrated multi-tool (CAE) systems were the focus.
- Towards the end of the decade, development of framework-based integration-oriented standards began (e.g. CFI, CFL)
- Relatively large CAD vendors evolved to dominate the CAD systems business (e.g. Mentor, Cadence, Valid)
- Virtually every successful product was initiated by a person or team that used to be designers.

A State-of-the-Art CAD Framework

- The tool integration environment includes higher-level facilities for:
  - Constructing user interfaces (*User Interface Services*)
  - Managing the CAD data associated with the design and coordinating access to the data by multiple CAD tools or human users (*Data Management Services*)
  - Managing the evolution, or history, of the design (*Version Services*)
  - Facilities for defining the legal organization of the data and what particular data items, and their relationships to other data items, mean (*Data Representation Services*).

Abstractions in a Framework Model



History of the Design Technology Industry

Electronic CAE

1960's

1970's

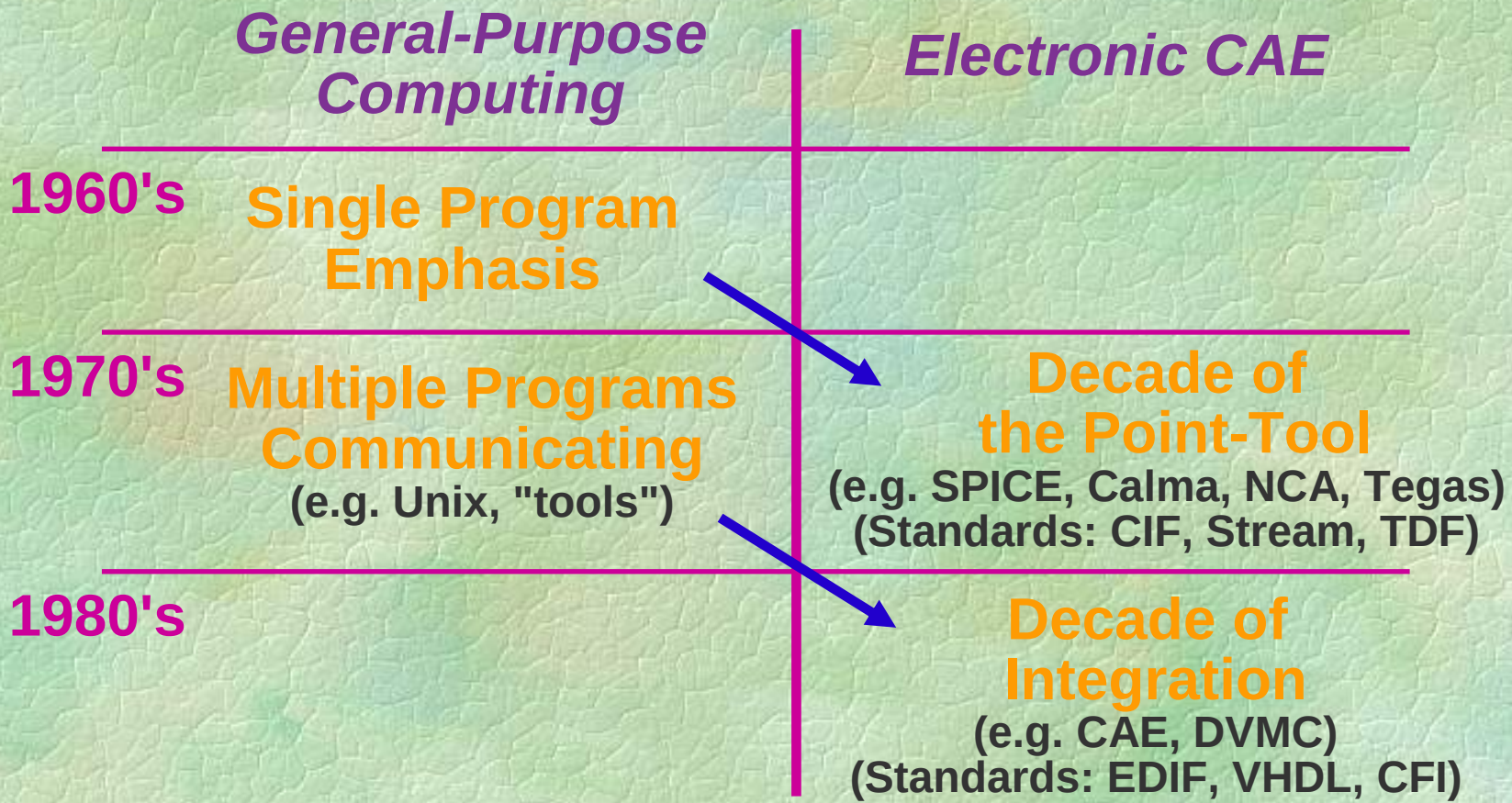
Decade of the Point-Tool
(e.g. SPICE, Calma, NCA, Tegas)
(Standards: CIF, Stream, TDF)

1980's

Decade of Integration
(e.g. CAE, DVMC)
(Standards: EDIF, VHDL, CFI)

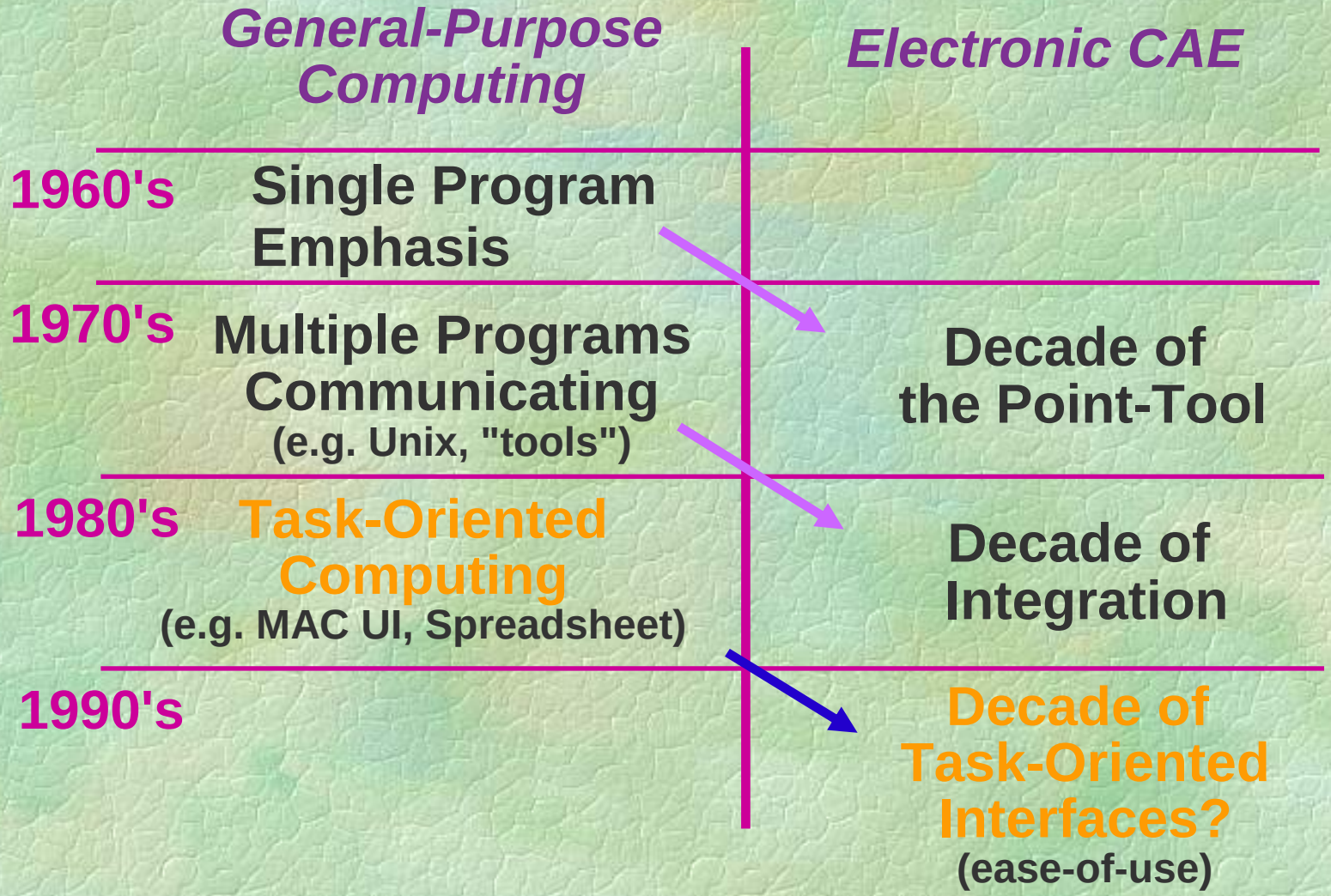
DAC Tutorial, 1992

History of the Design Technology Industry



DAC Tutorial, 1992

1990s: The Third Generation



Industry Trends

- Impact of Design Complexity Coupled with Shrinking Time-to-Market Windows
  - Broad range of skills needed to complete designs

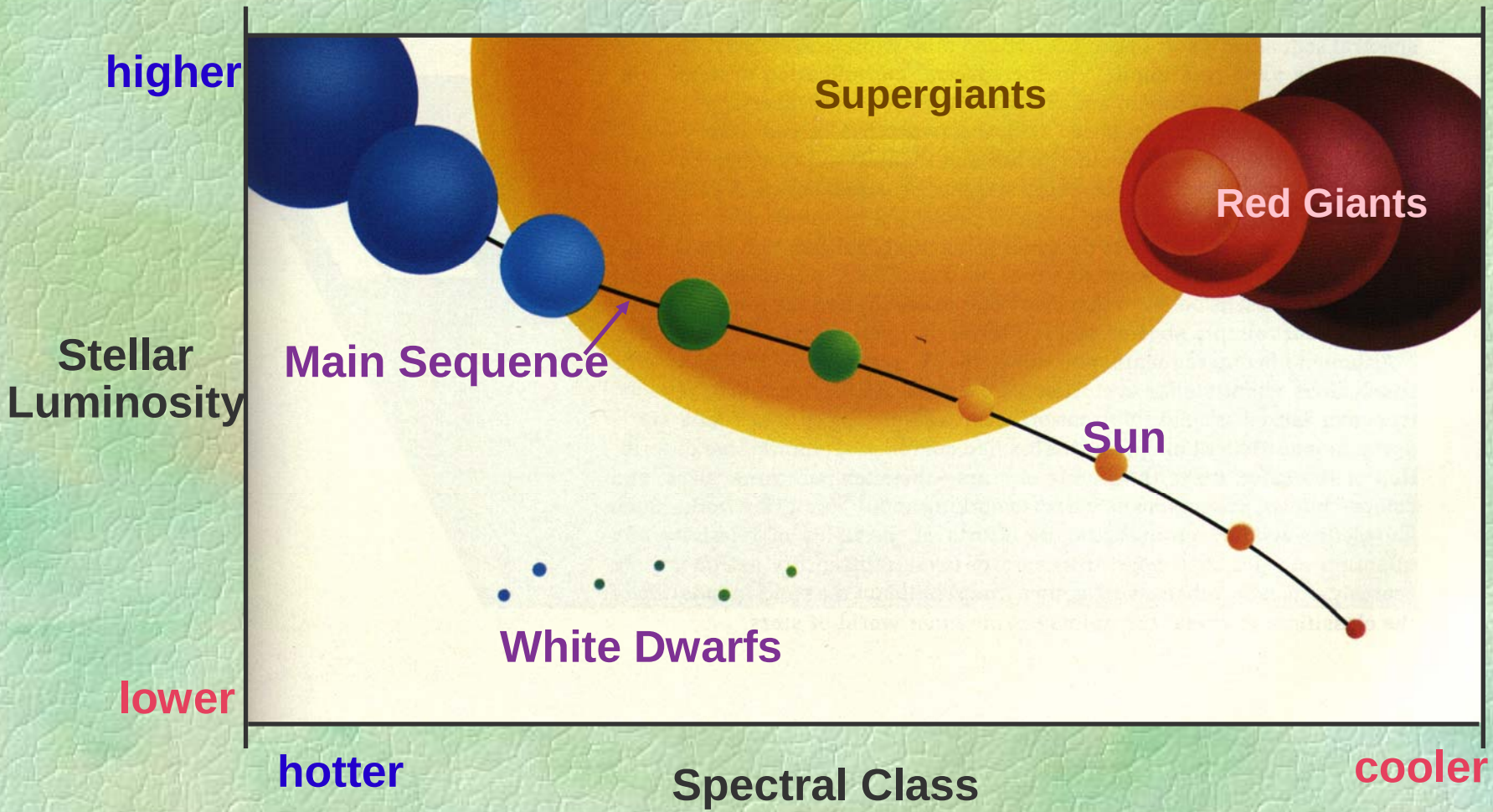
    e.g. analog design, RF, high-speed digital design, communications protocols, multi-chip, video coding, compression, 3D rendering
  - Need for specific expertise in-house short lived in many areas
  - Not *everyone* wants to live in Silicon Valley
  - Manufacturing economies are driving *outsourcing model*

➔ Major shift towards shared, remote expertise and outsourcing of many aspects of design and manufacturing will accelerate

Collaboration in Electronic Systems Design: Commercial State-of-The-Art

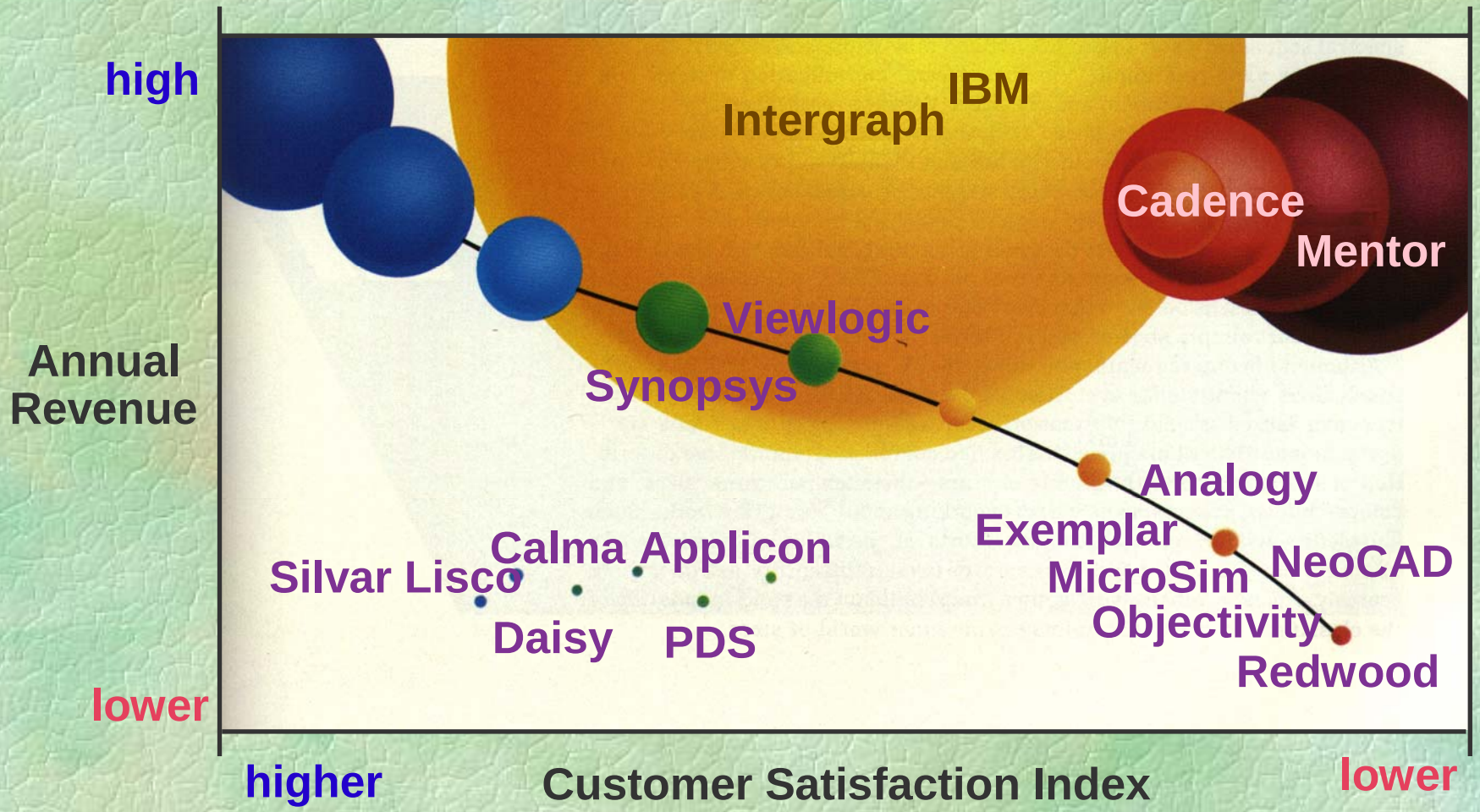
- Ad-hoc application of code-control analogy
  - Design components and manipulated at coarse-grain level
  - Shared, in-memory data structures for all tool interaction
  - Some use of IPC-based remote tool use
  - Data(file)-copying for design sharing/transfer
  - Overall (external) simple versioning mechanisms
- Design manager's nightmare!

Hertzsprung-Russell Diagram



DAC Tutorial, 1992

Evolution of Design Technology Companies

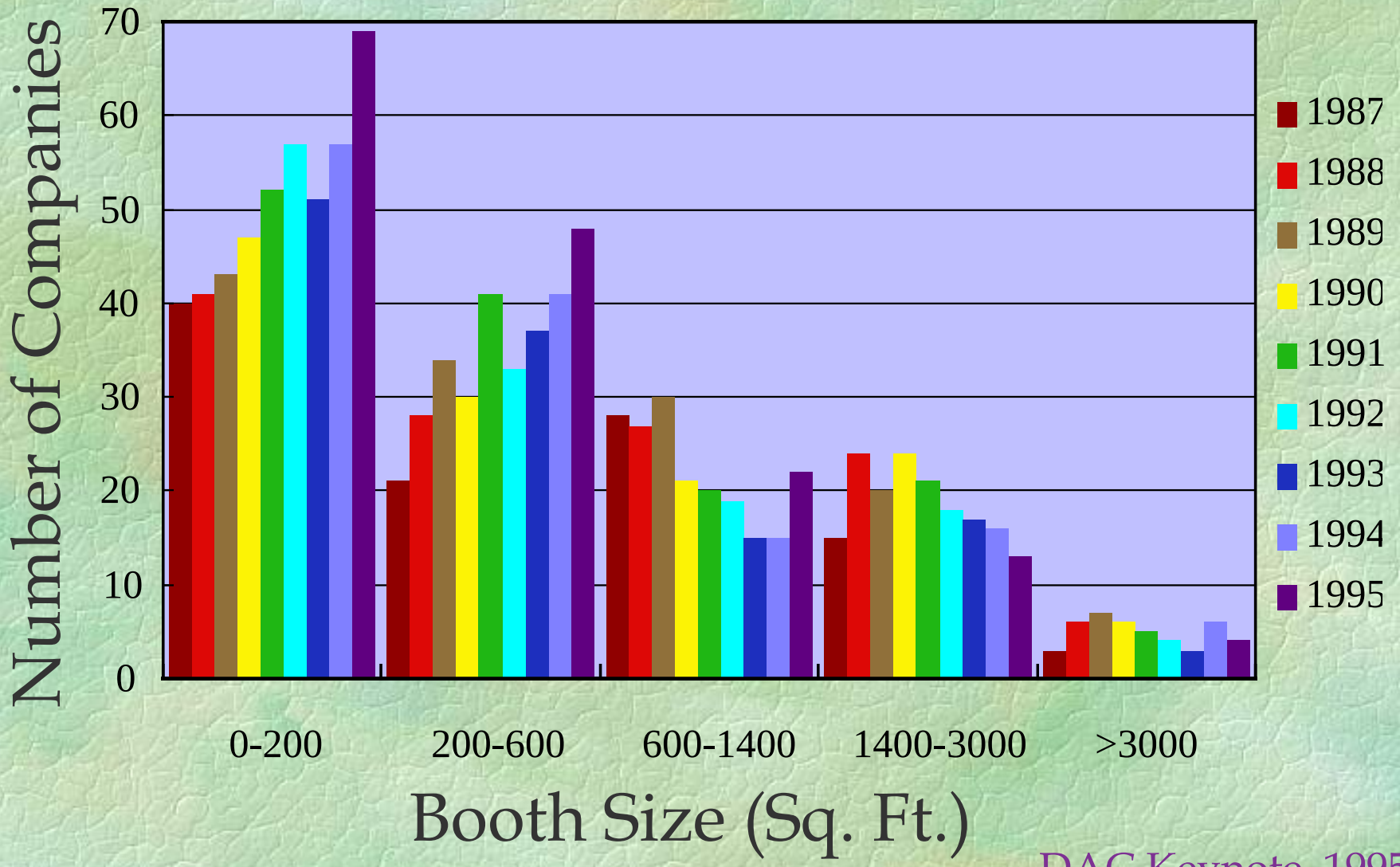


DAC Tutorial, 1992

The Dilemma

- No electronic design technology system supplier has ever made a successful transition to distribution channel
  - We are in the midst of the third generation of attempts
  - It will require real contribution from users
- The key missing ingredient is a prejudice-free integration standard with wide availability
  - The "DOS" or "finder" of engineering.
  - It's not rocket science, but it is a serious engineering and business issue!

ACM/IEEE Design Automation Conference
Exhibit Booth & Suite Space



DAC Keynote, 1995

Impact of Collaboration Technology

- The “cell-based” design methodologies applied successfully over the past thirty years *will not scale much further*.
- A new, *comprehensive* approach to System-On-a-Chip (SOC) design is required.
  - The relatively radical changes to the design process that are likely to be required *will provide new opportunities for other key players in the semiconductor industry*, especially in design technology.
- The effectiveness of the collaborative environment and the ability to “visualize” the design as well as the design process will *ultimately determine the quality of any overall solution* (methodology) to the microelectronics design problem.

Use of Networks in EDA Integration

- Floating licenses, leading to time-based accounting
- Internal, relatively explicit use of distributed networks
- Web sites for marketing, recruiting, etc.
- Limited use for software purchases and distribution (distribution of keys)
- Software support: e.g. Synopsys SOLV-IT!
- Basically, we are *way* behind other industries…
- *Very* few dollars being spent on R&D in this area!

EDA Integration Environments

- EDA has always leveraged methodological developments in other, related areas of computer science
- Huge opportunity to bring additional stability to the EDA industry and empower young, innovative companies in tools, design, and services
- Would help significantly in the ability to leverage University EDA research
- Has never received the investments in R&D it needs and deserves
- As a community, we are already significantly behind the broader industry in the adoption of new technology in this area!

Outline

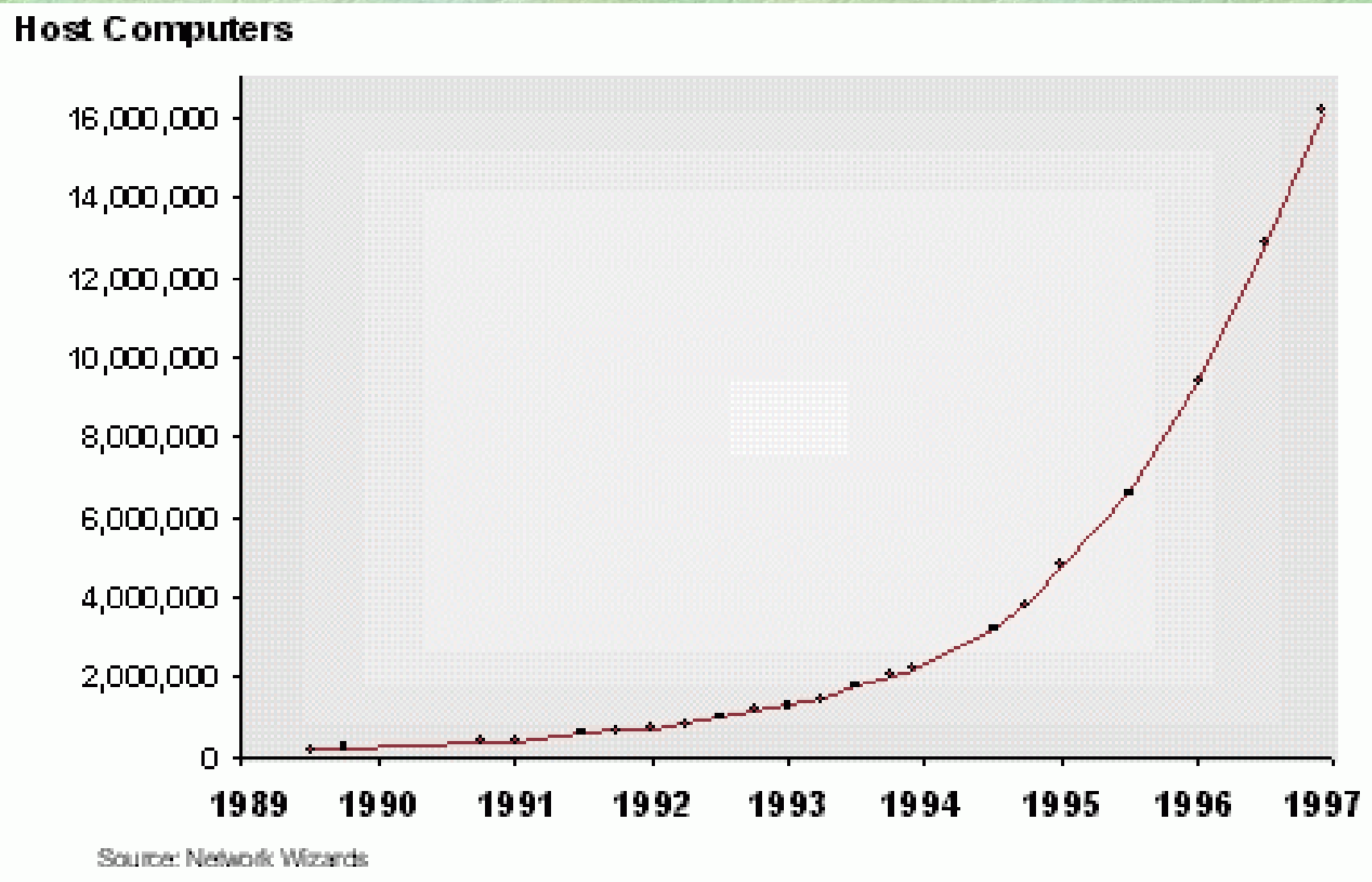
- Semiconductors and EDA
  - How did we get here?
  - Where are we going?
- Networks and the Internet
- The Opportunity
- Distributed Computing: The Future of the OS
- The Role of Java
- Visualization as a Critical Technology
  - The design and associated algorithms
  - The process of collaboration

Perspectives and Market Opportunities

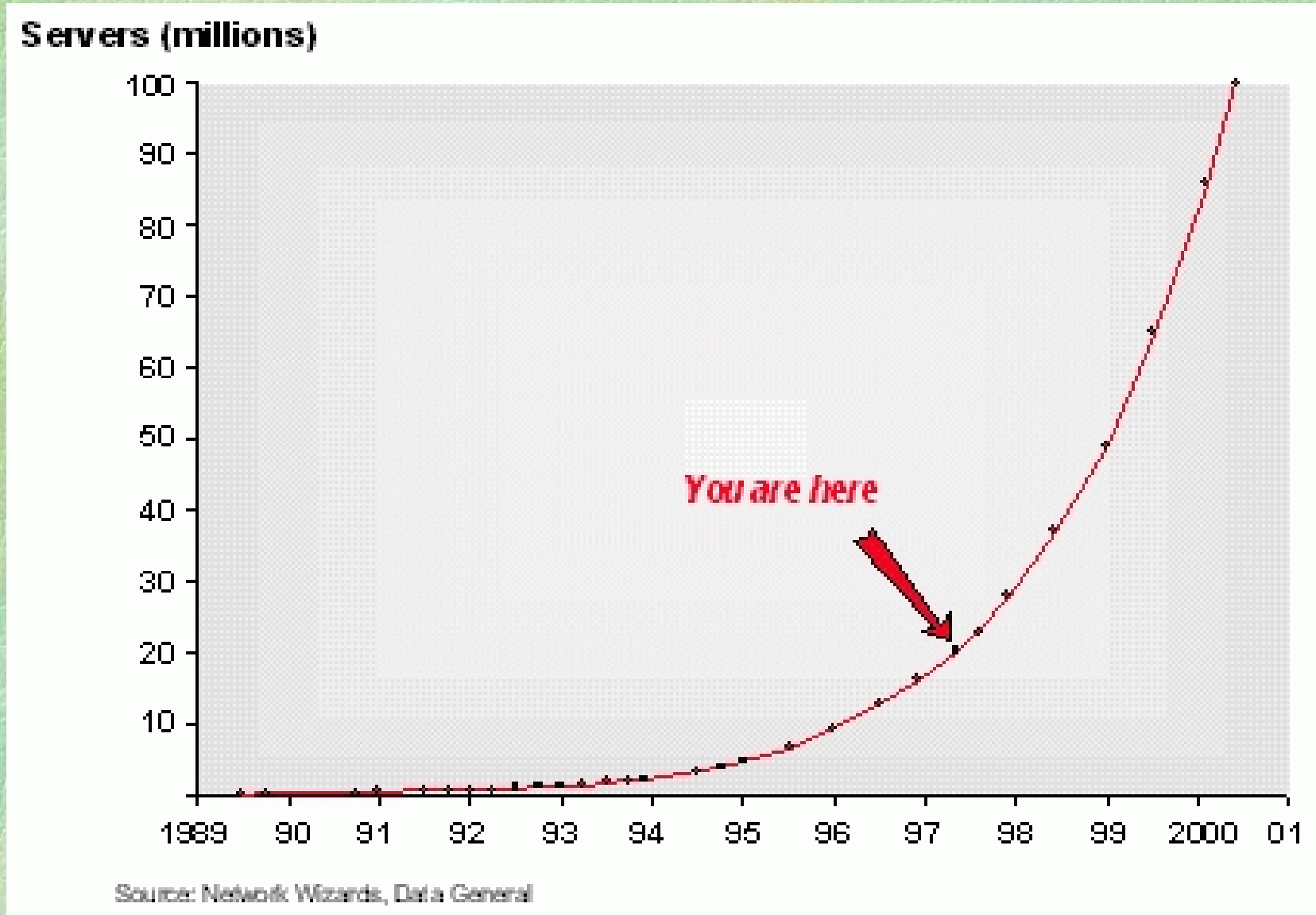
- Less Than:
  - 15% office workers have PCs
  - 10% US population has cellular phones
  - 10% worldwide PC users have electronic mail
  - 7% worldwide PC users have real-time Internet access
  - 4% US population has real-time Internet access
- For US Homes, More Than:
  - 95% have television sets
  - 95% have corded telephones
  - 85% have VCRs
  - 60% have cable TV
  - 45% have video gaming software

Source: Morgan Stanley

Growth of Internet Hosts



Growth of Internet Servers



The Connectivity Revolution

- It is a total revolution, it is happening, and it will only accelerate
- Assume the number of everything of interest is extremely large (infinite)
- For example, servers should be thought of as distributed and ubiquitous: Millions/organization, not tens or hundreds
- Managing the evolution of organizations and practices with very large numbers of very agile components is the key architectural challenge!

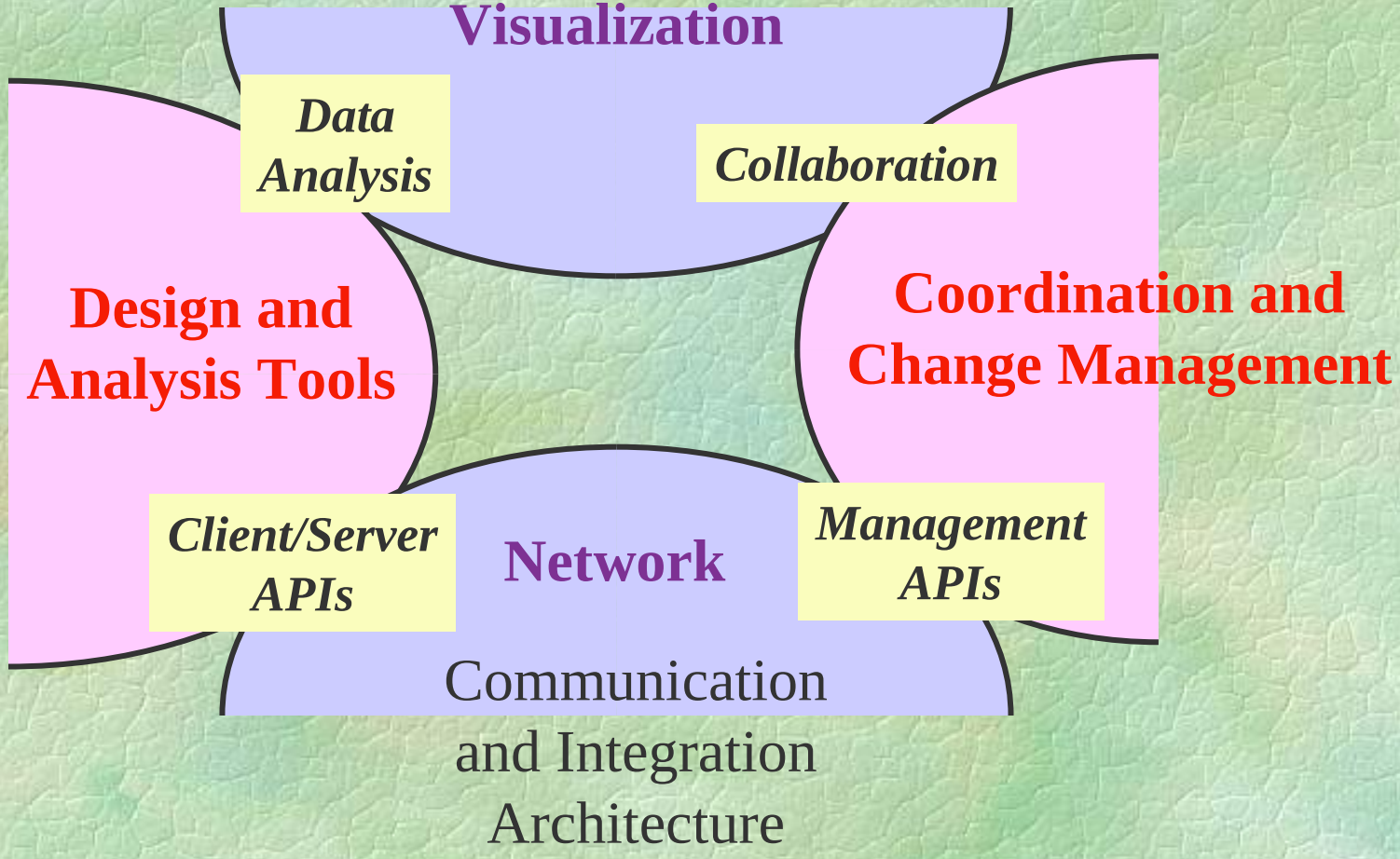
Outline

- Semiconductors and EDA
  - How did we get here?
  - Where are we going?
- Networks and the Internet
- The Opportunity
- Distributed Computing: The Future of the OS
- The Role of Java
- Visualization as a Critical Technology
  - The design and associated algorithms
  - The process of collaboration

The Opportunity

- A New Generation of Infrastructure
  - Powerful, low-cost computation
  - Networks, 3D graphics support, haptic interfaces
- The Need for a New Approach to Design
  - Deep-sub-micron silicon
  - Strong economic driver: time-to-money
- (Relatively) Well-Defined Problem
  - Well-defined inputs
  - Well-defined (high-level) required output
  - Clear measures of success and failure!

Major Aspects of the Integration Environment



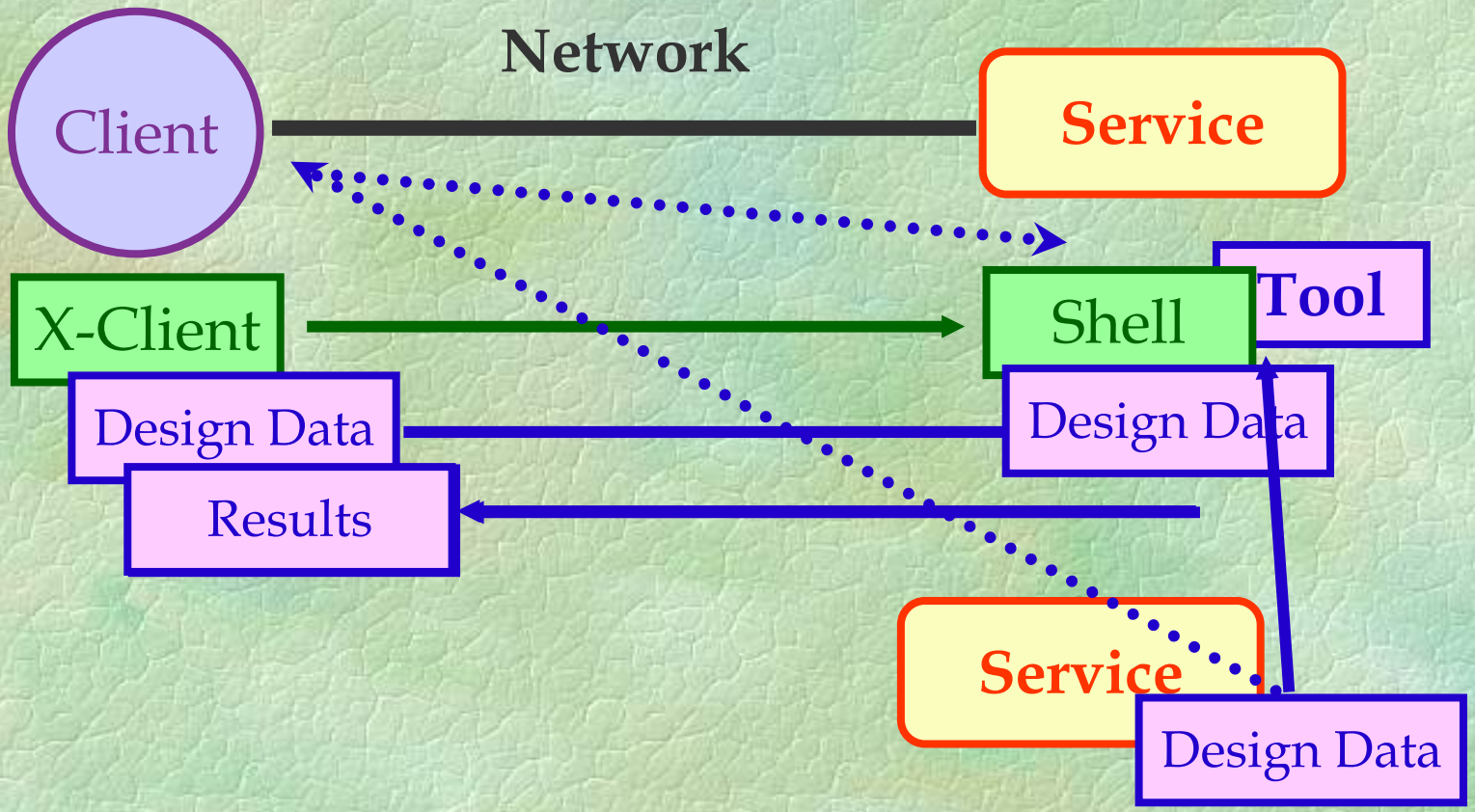
Single, Integrated EDA Community

- Scaleability: Must support exponential growth in the number of users and servers
- Availability: Access uninterrupted and service degradation due to catastrophic failure should appear “graceful.”
- Adaptability: Users, services, and servers added and removed incrementally. Must adapt automatically.
- Robustness: Any failure which may result in loss of data must be recoverable efficiently and incrementally.
- Cost Effectiveness: All of the above must be achieved cost effectively.

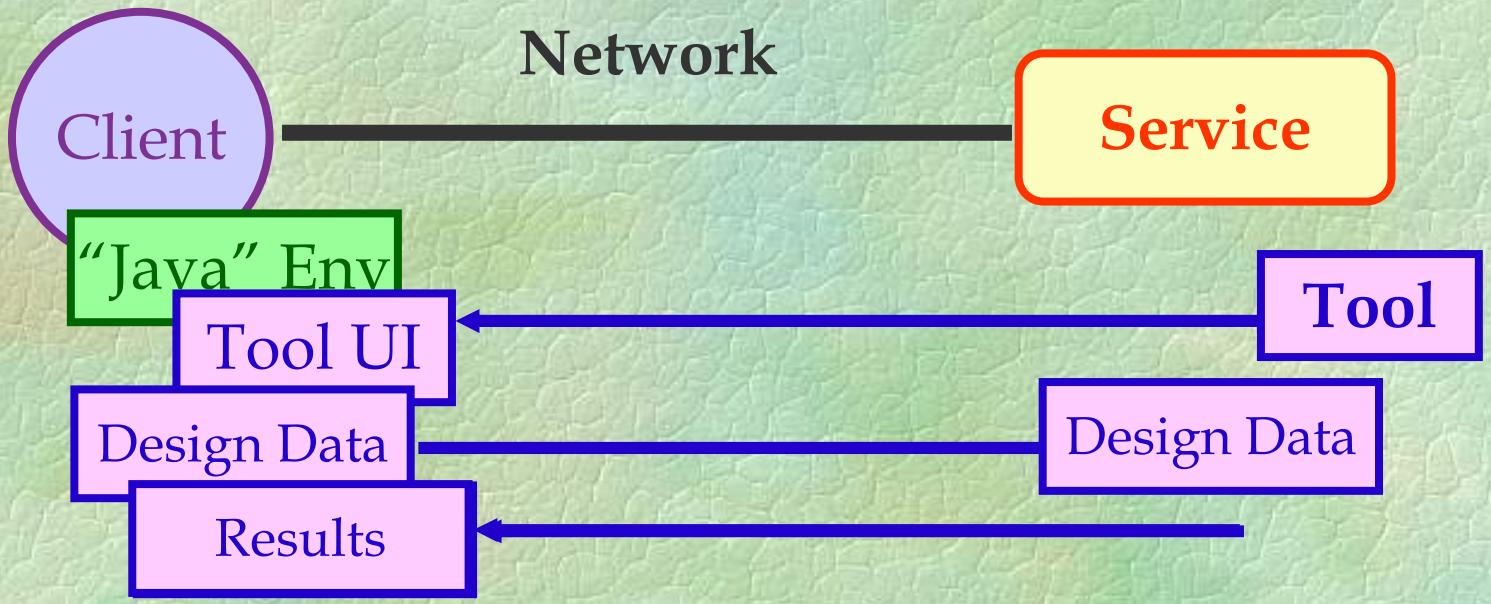
Design Transaction Management

- ACID: Traditional Transactional Database
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- BASE: Requirements for Engineering Design
  - Basically Available
  - Soft State
  - Eventual Consistency
- Like implementing high-performance software on a "bad" multiprocessor.

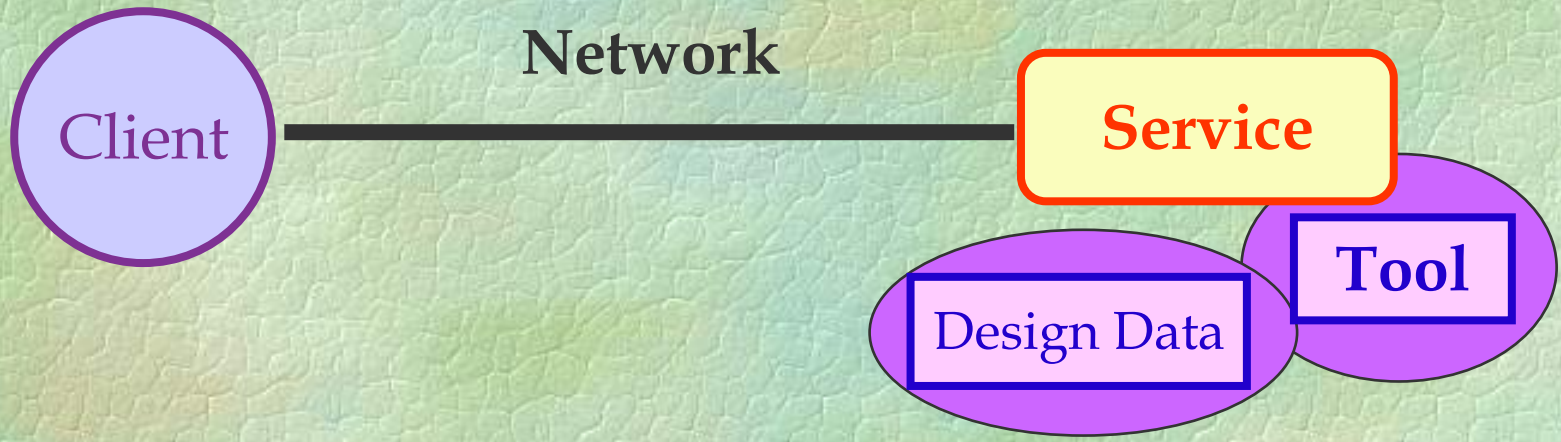
Distributed Network Architecture



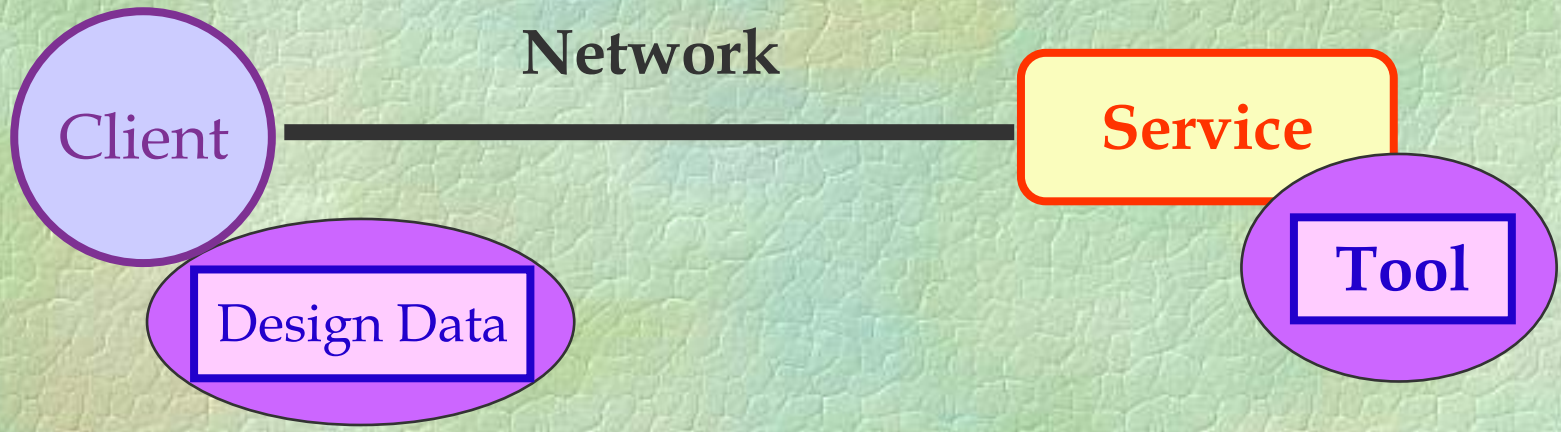
Distributed Network Architecture



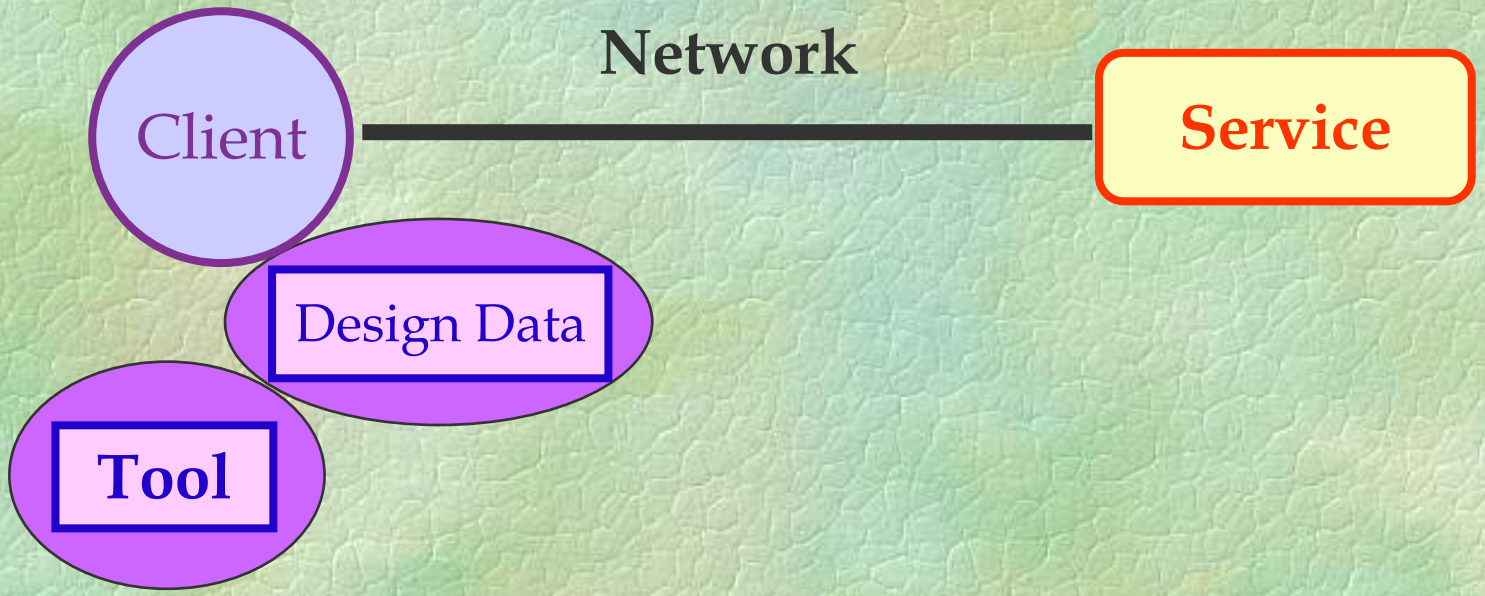
Distributed Network Architecture



Distributed Network Architecture

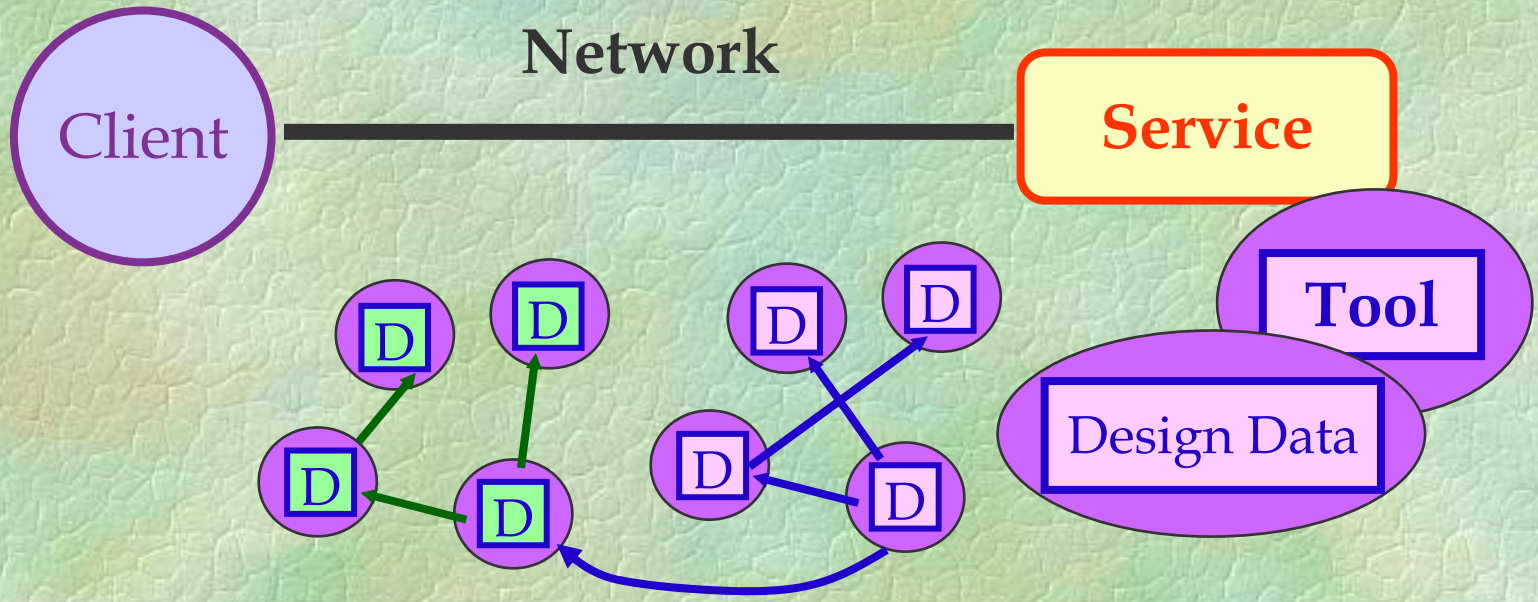


Distributed Network Architecture



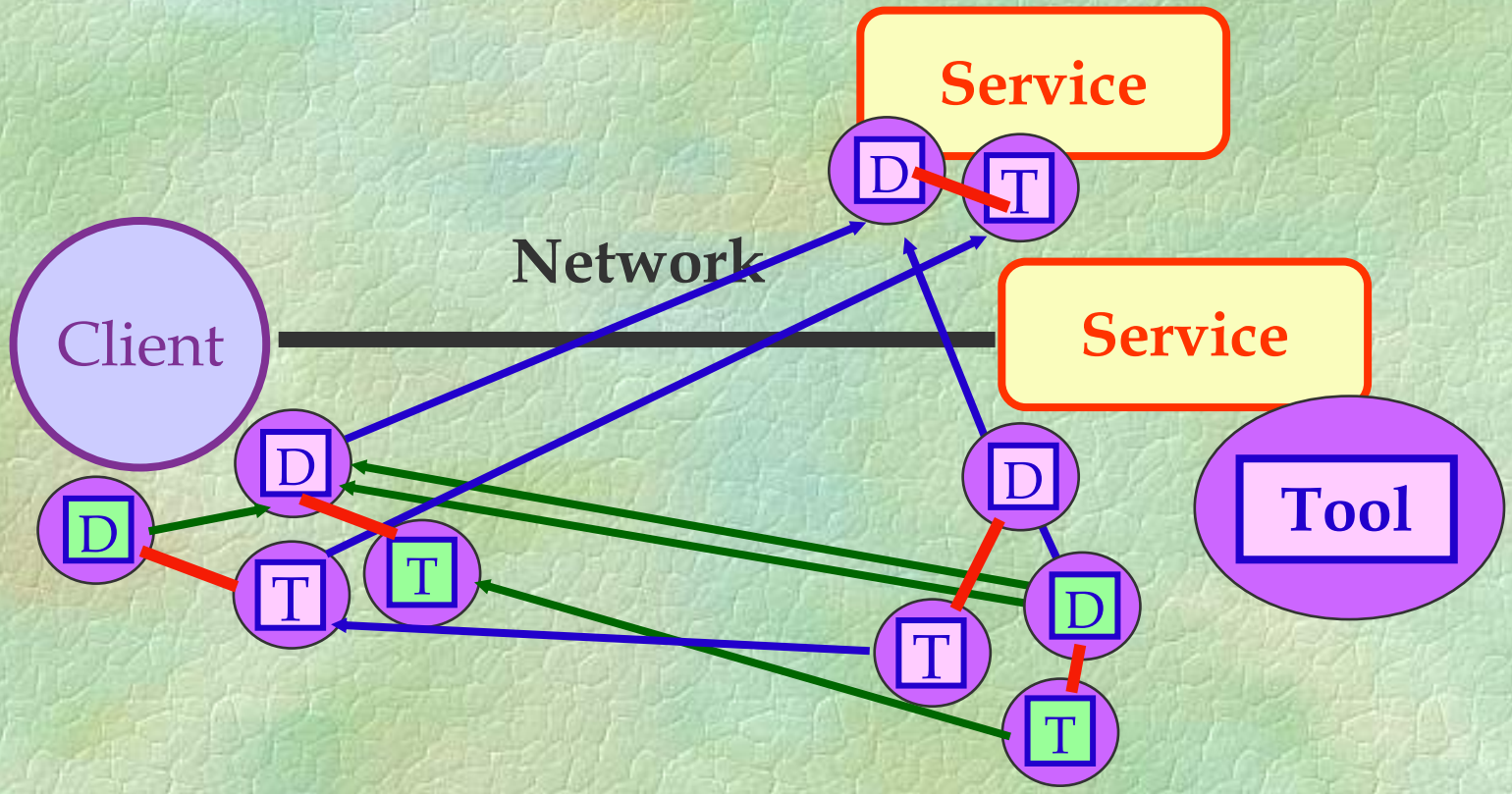
- The user/developer does not want to know!
- Peer-oriented approach

Distributed Network Architecture



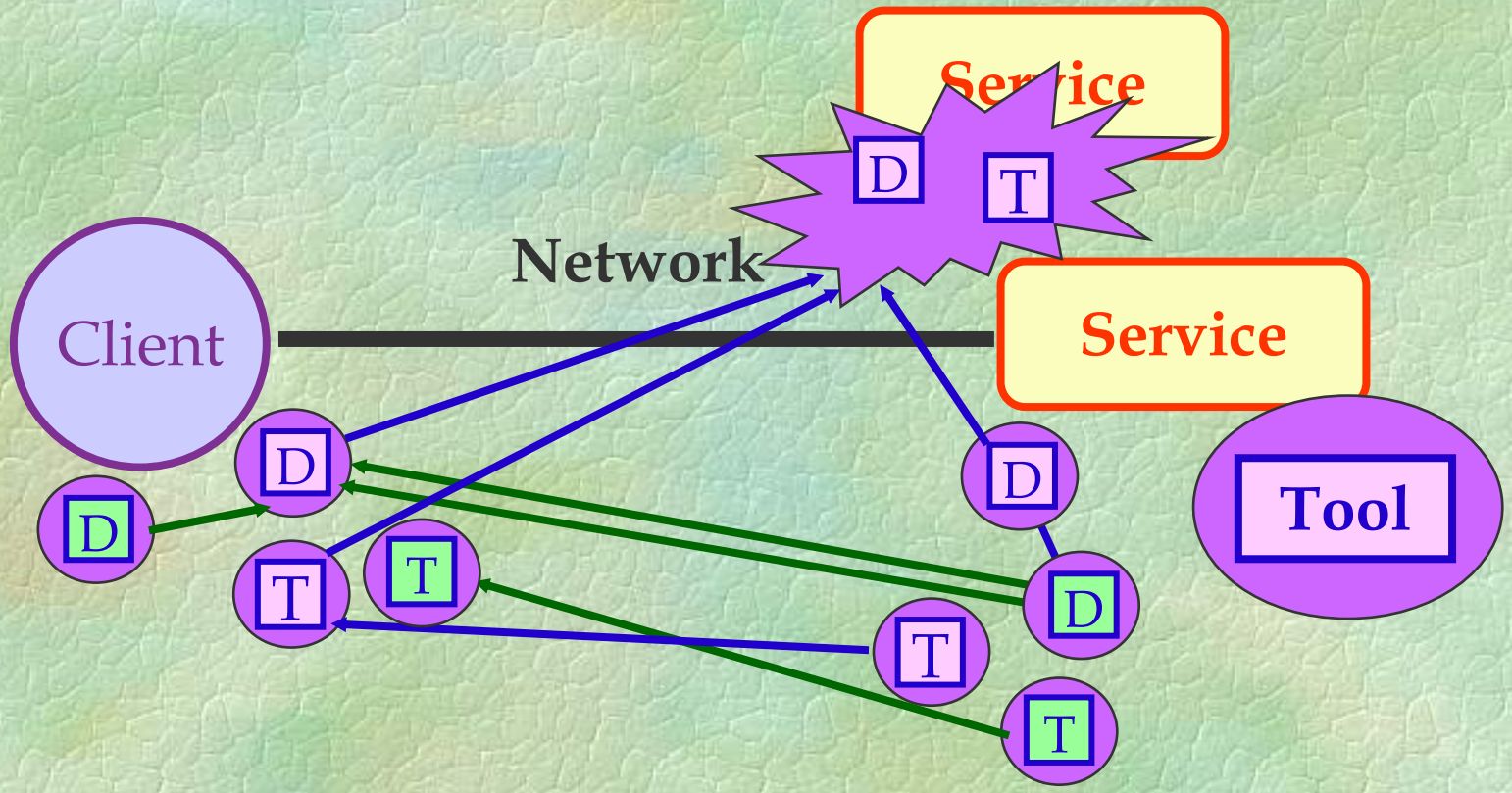
- But what about the UI?
- Must implement “EDA Code ‘Design Reuse’”

Distributed Network Architecture



- Distributed Data, Distributed “tool-lets” composed to provide overall capability
- What’s the architecture?? … Don’t forget transactions!

Distributed Network Architecture



- "Data as Tools" or "Tools as Data" or "ComponentWare"
- "Self-X-ing" Components: X= simulate, layout, verify, test, abstract,...

What is ComponentWare in this Context?

- Not like traditional hard libraries or module-generators but include all necessary software (literally or as a reference) to build, verify, test, and transfer the component (μP or DSP core, random logic, memory, I/O, etc.) to manufacturing.
- Probably family-oriented components, but not clear...

Outline

- Semiconductors and EDA
  - How did we get here?
  - Where are we going?
- Networks and the Internet
- The Opportunity
- Distributed Computing: The Future of the OS
- The Role of Java
- Visualization as a Critical Technology
  - The design and associated algorithms
  - The process of collaboration

Next-Generation Operating Environments

- Advances in hardware and networking will enable an entirely new kind of operating system, which will raise the level of abstraction significantly for users and developers.
- Such systems will enforce extreme location transparency
  - Any code fragment runs anywhere
  - Any data object might live anywhere
  - System manages locality, replication, and migration of computation and data
- Self-configuring, self-monitoring, self-tuning, scaleable and secure

Next-Generation Operating Environments

- Seamless Distribution: System decides where computation should execute or data should reside, moving them there dynamically
- Worldwide Scalability:Logically there should only be one system, although at any one time it might be partitioned into many pieces.
- Fault-Tolerance: Transparently handle failures or removal of machines, network links, etc.

Next-Generation Operating Environments

- Self-Tuning: System should be able to reason about its computations and resources, allocating, replicating, and migrating computation and data to optimize performance, resource usage, and fault tolerance.
- Self-Configuring: New machines, network links, and resources should be automatically assimilated.
- Security: Allow non-hierarchical trust domains.
- Resource Controls: Both providers and consumers may explicitly manage the use of resources belonging to different trust domains.

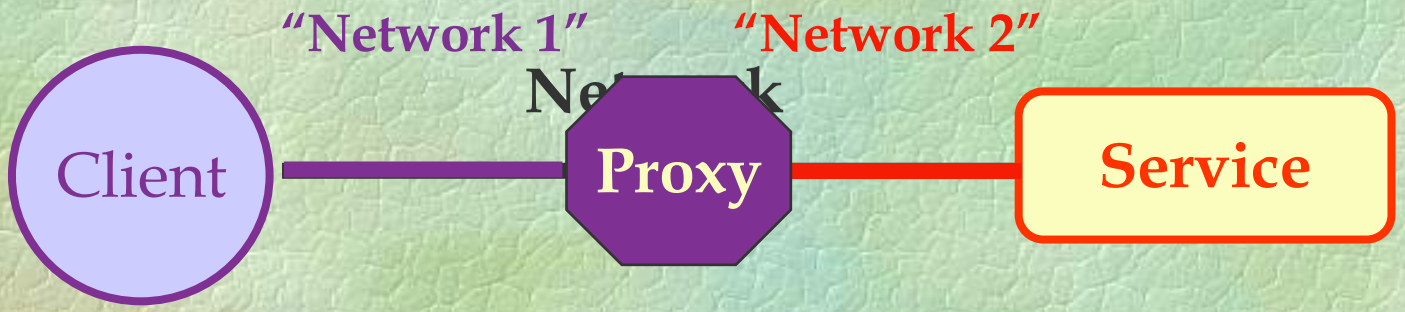
Next-Generation Operating Environments

- No Storage Hierarchy: Once information is created, it should be accessible until it is no longer needed or referenced.
- Introspection: The system should posses some aspects of introspection and reflection.
  - Pervasively self-monitoring
  - Reason about its own configuration and performance
  - Suggest improvements
- Just-in-Time Binding: Sort of like the Internet today, but extended to all object interactions. “Binding-by-Search”
- Tools Emphasis Shifting: From code-efficiency to rapid application development with wizards automatically generating scaffolding or framework code.

“WebOS”

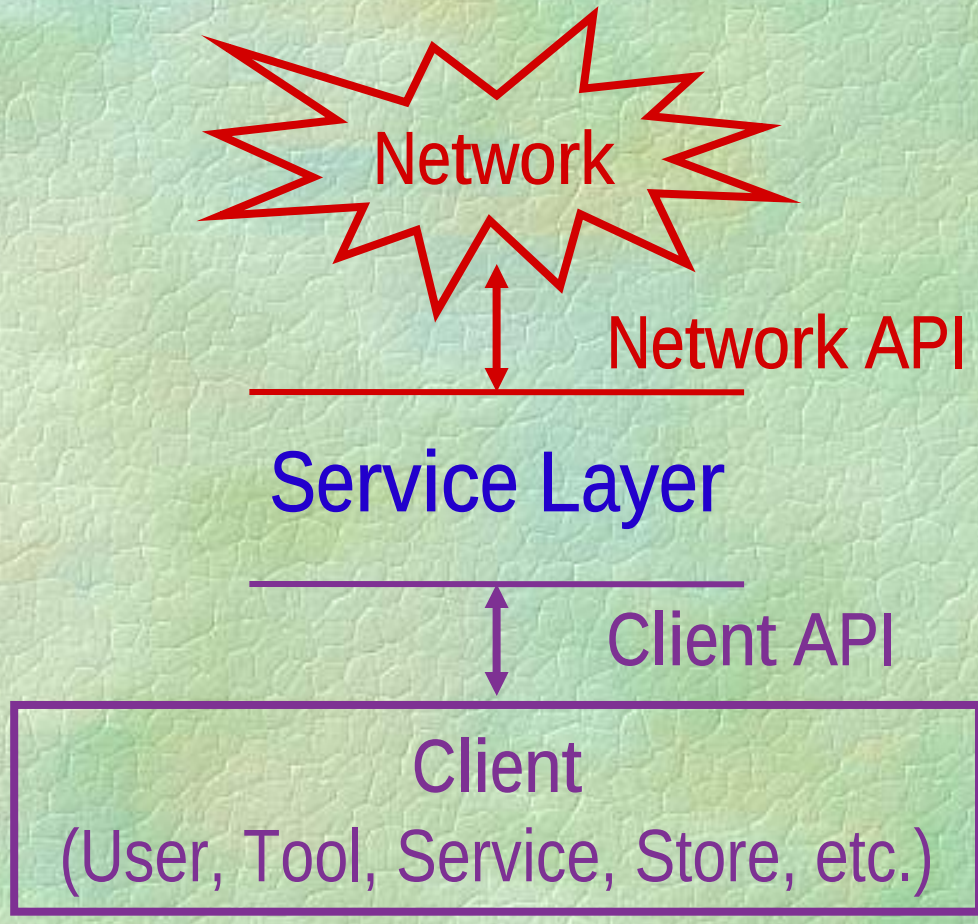
- The goal is to provide a common set of OS services to wide area applications, including mechanisms for:
  - Resource discovery
  - A global namespace
  - Remote process execution
  - Resource management
  - Authentication
  - Security
- Provide services needed to build applications that are:
  - Geographically distributed
  - Highly available
  - Incrementally scalable
  - Dynamically reconfiguring

Distributed Network Architecture

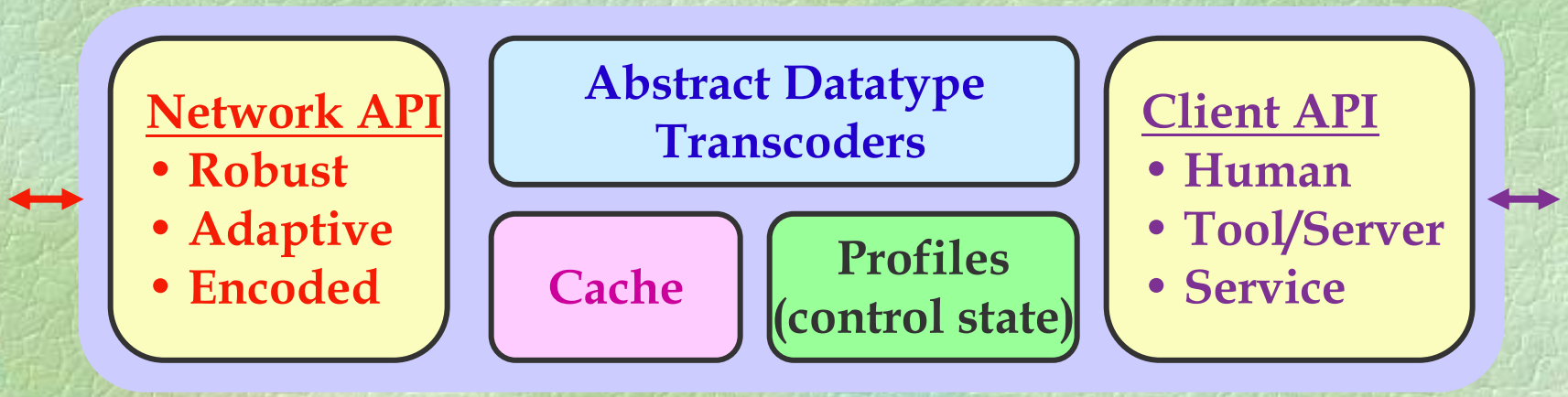


- We lost something along the way...
- Proxy: Or "agent", or "service layer", the trusted intermediary.
- "My EDA representative"

The Role of the Service Layer

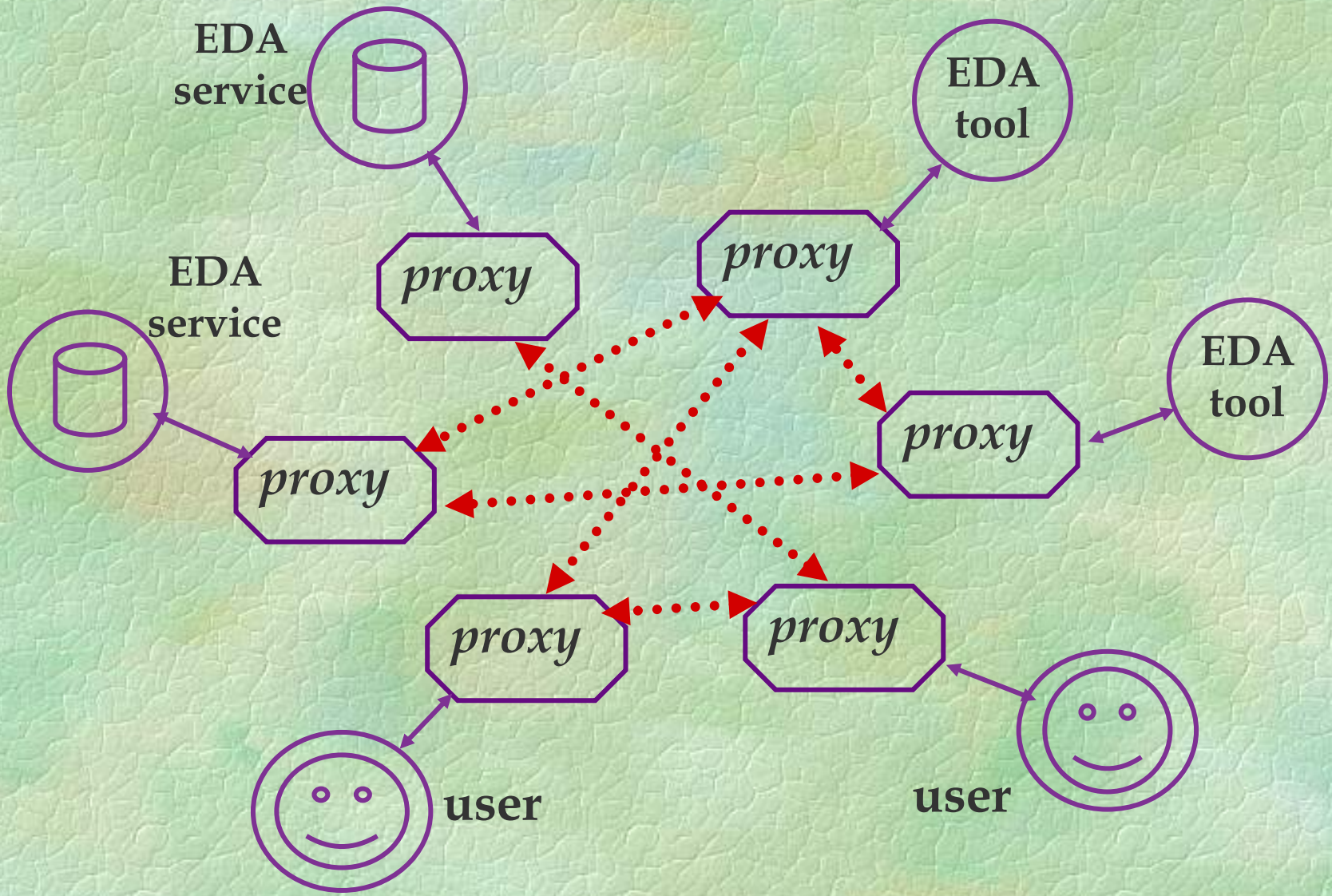


One Organization of a Proxy

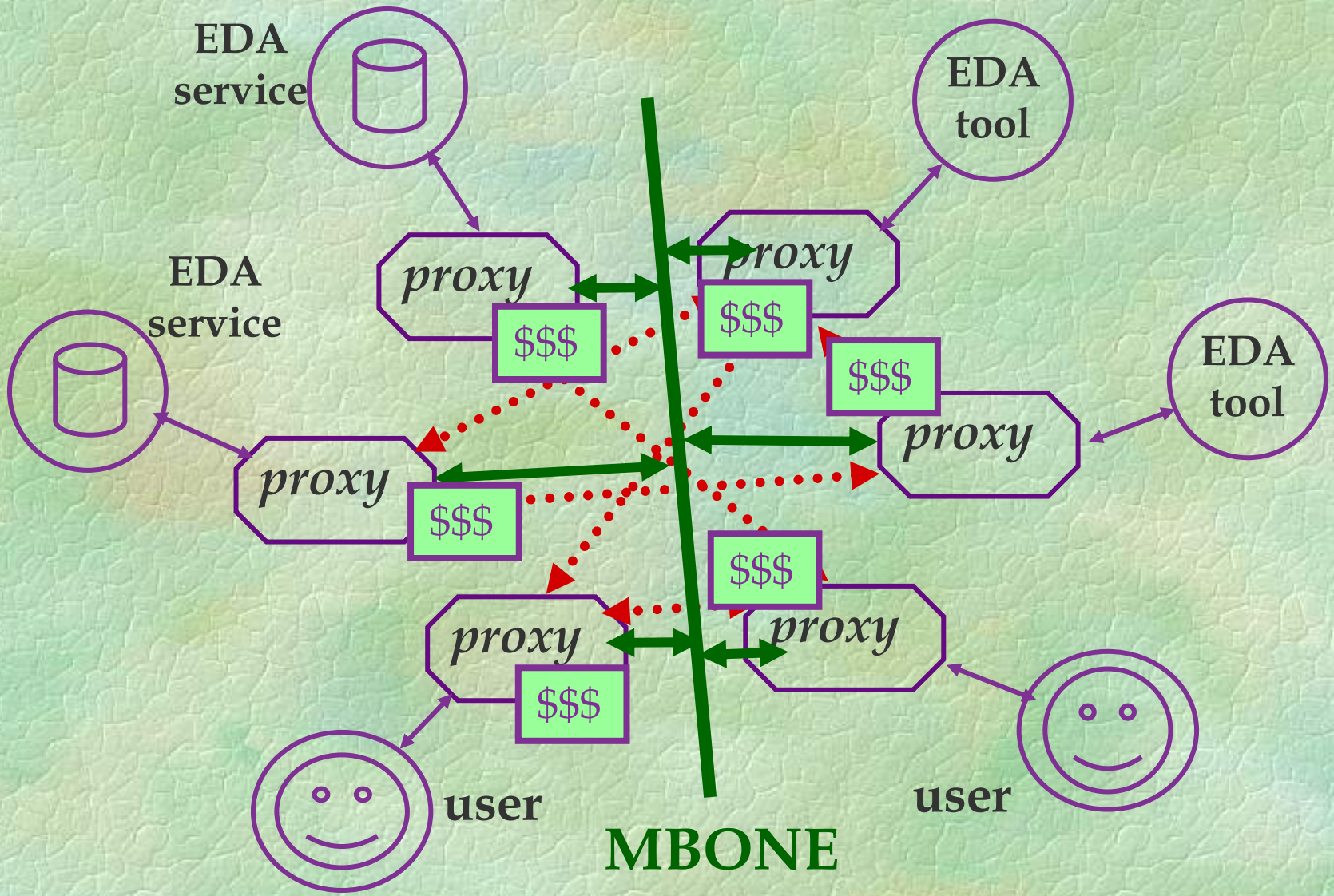


- Just one possible approach...
- What is the role of the cache? Is it a good thing?
- How do we implement namespace management?
- How do we implement data migration efficiently?
- It must also be a revolution!

The WELD Perspective



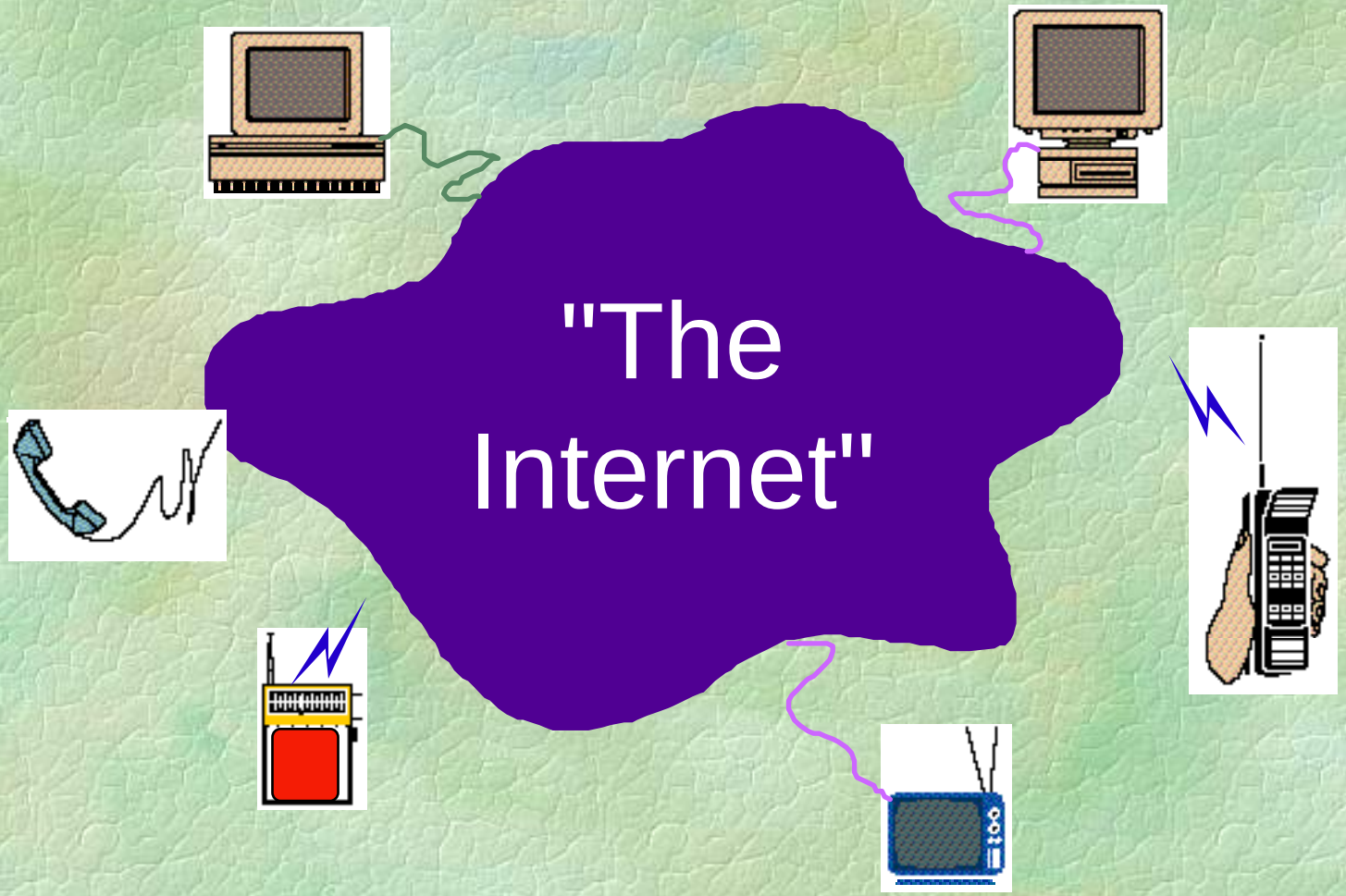
The WELD Perspective



Outline

- Semiconductors and EDA
  - How did we get here?
  - Where are we going?
- Networks and the Internet
- The Opportunity
- Distributed Computing: The Future of the OS
- The Role of Java
- Visualization as a Critical Technology
  - The design and associated algorithms
  - The process of collaboration

The "Java Terminal"



The "Network Computer"
(or "Hollow Client")

- Simple and Uniform Architecture

LCD Display

ROM

SRAM

Flash

"Thin Client"

Boot Code

Cache

Hi-Res Display

VRAM

ROM

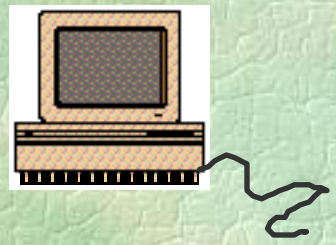
SRAM

DRAM

Disc

CD-RO

"Fat Client"



The “Network Computer”

- All Permanent State in the 'Net
- View the System as a "Bad Multiprocesor"
  - How to exploit locality
  - How to implement “transactions”
  - Light-weight, agent-based vs. database
- Back Doors for Non-‘Java-Bytecode’
  - Prediction: Eventually, almost *all* code will be written in Java
  - Tools will be developed to support Java for applets, applications, and real-time embedded systems
- Authentication/Identification Issues

What Percentage of New EDA Licenses do You Think will be Issued for Windows NT Platforms in CY2000?

	< 20%	20-40%	40-60%	60-80%	> 80%
Gerry Hsu, Avant!		◆			
Joe Costello, Cadence			◆		
Wally Rhines, Mentor			◆		
Keith Lobo, Quickturn		◆			
Aart De Geus, Synopsys			◆		
Will Herman, Viewlogic				◆	

What Percentage of New EDA Licenses do You Think will be Issued for Network Computers with Server-Farms (true client-server) Platforms in CY2000?

	< 5%	5-10%	10-20%	20-30%	> 30%
Gerry Hsu, Avant!		◆			
Joe Costello, Cadence		◆			
Wally Rhines, Mentor			◆		
Keith Lobo, Quickturn					◆
Aart De Geus, Synopsys			◆		
Will Herman, Viewlogic					◆

"Networks of Workstations"

The Role of Java in Distributed Applications

- In use today as a portable, client-side GUI development environment
- Over time, is likely to become the language-of-choice for developers: “Just another language”
- A cornerstone for a number of new operating environment developments

What is *JavaBeans*?

- *JavaBeans* is a portable, platform-independent component model written in Java.
- It enables developers to write reusable components once and run them anywhere -- benefiting from the platform-independent power of Java.
- *JavaBeans* acts as a bridge between proprietary component models and provides a seamless means for developers to build components that run in ActiveX container applications.

Attributes of *JavaBeans*

- Introspection: enables a builder tool to analyze how a Bean works
- Customization: enables a developer to use an app builder tool to customize the appearance and behavior of a Bean
- Events: enables Beans to communicate and connect together
- Properties: enable developers to customize and program with Beans
- Persistence: enables developers to customize Beans in an app builder, and then retrieve those Beans, with customized features intact, for future use

Java-as-a-Syntax for Embedded Systems

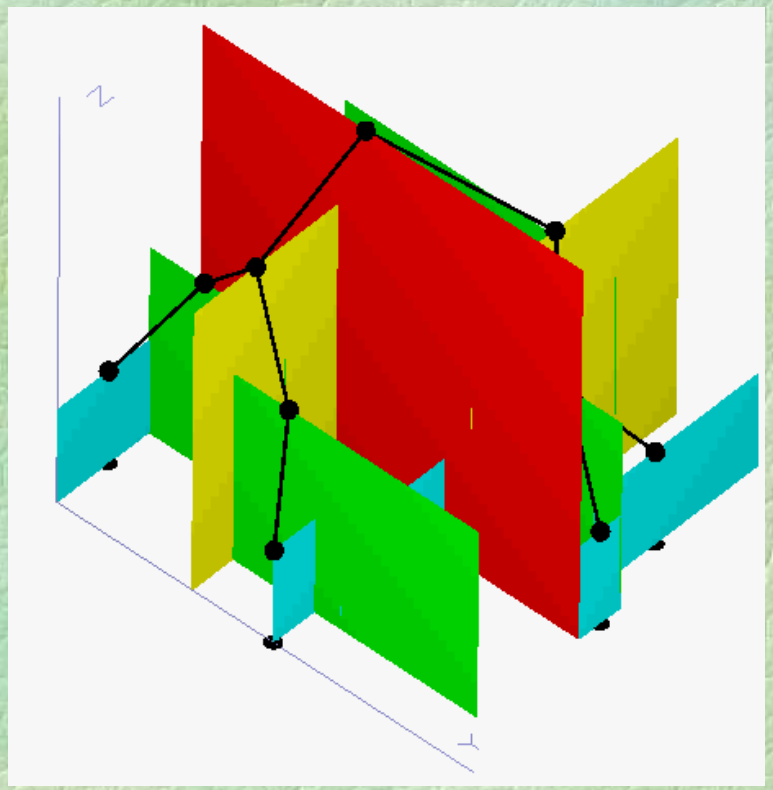
- Two very distinct reasons for exploring this path: *pragmatic* and *research-directed*
- Pragmatic:
  - Is becoming the standard “introduction to programming” language at many schools
  - Huge software development investment infrastructure to leverage
  - Similarity to C and C++
  - Ability to “embed” other languages
- Research Aspects:
  - Built in multithreading and synchronization
  - Lack of pointer arithmetic (…still reference-based)
  - Automatic memory management
  - Extensible to include temporal constraints
- … Small steps, but can be highly leveraged

Outline

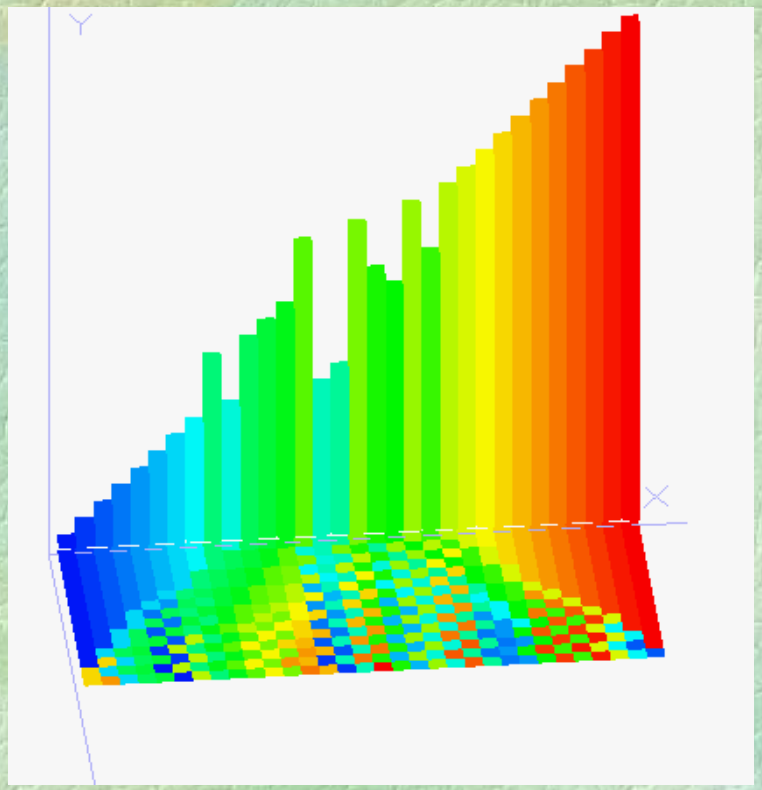
- Semiconductors and EDA
  - How did we get here?
  - Where are we going?
- Networks and the Internet
- The Opportunity
- Distributed Computing: The Future of the OS
- The Role of Java
- Visualization as a Critical Technology
  - The design and associated algorithms
  - The process of collaboration

Representation as an Aid in Decision Support

- e.g. M. Brown, et. al., Digital; Xerox



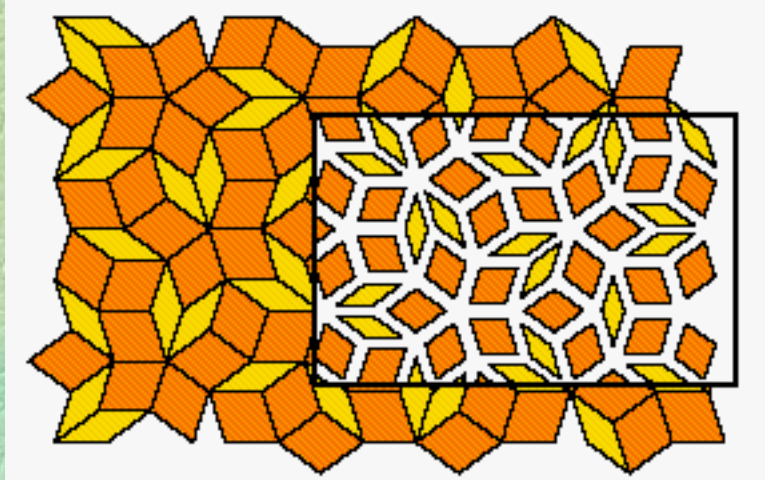
k-D Tree



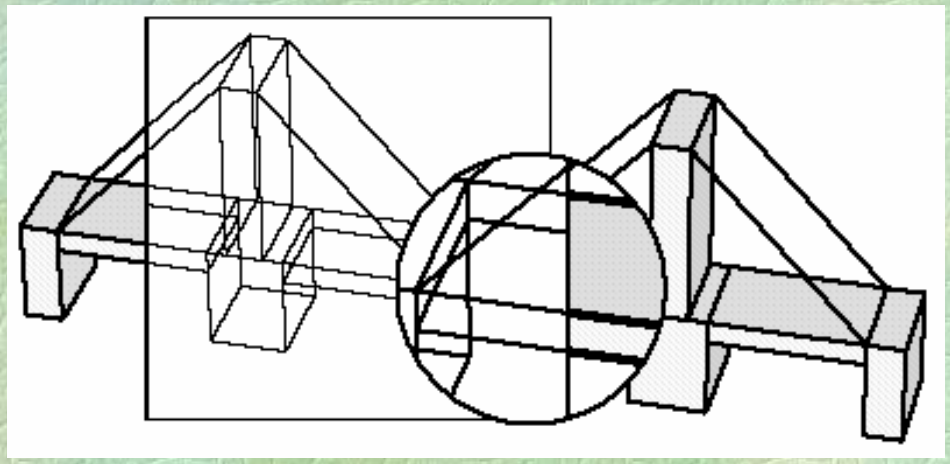
ShakeSort

Beyond Hierarchy

- e.g. Level-of-Detail (VRML 2), Magic Lens (Xerox)



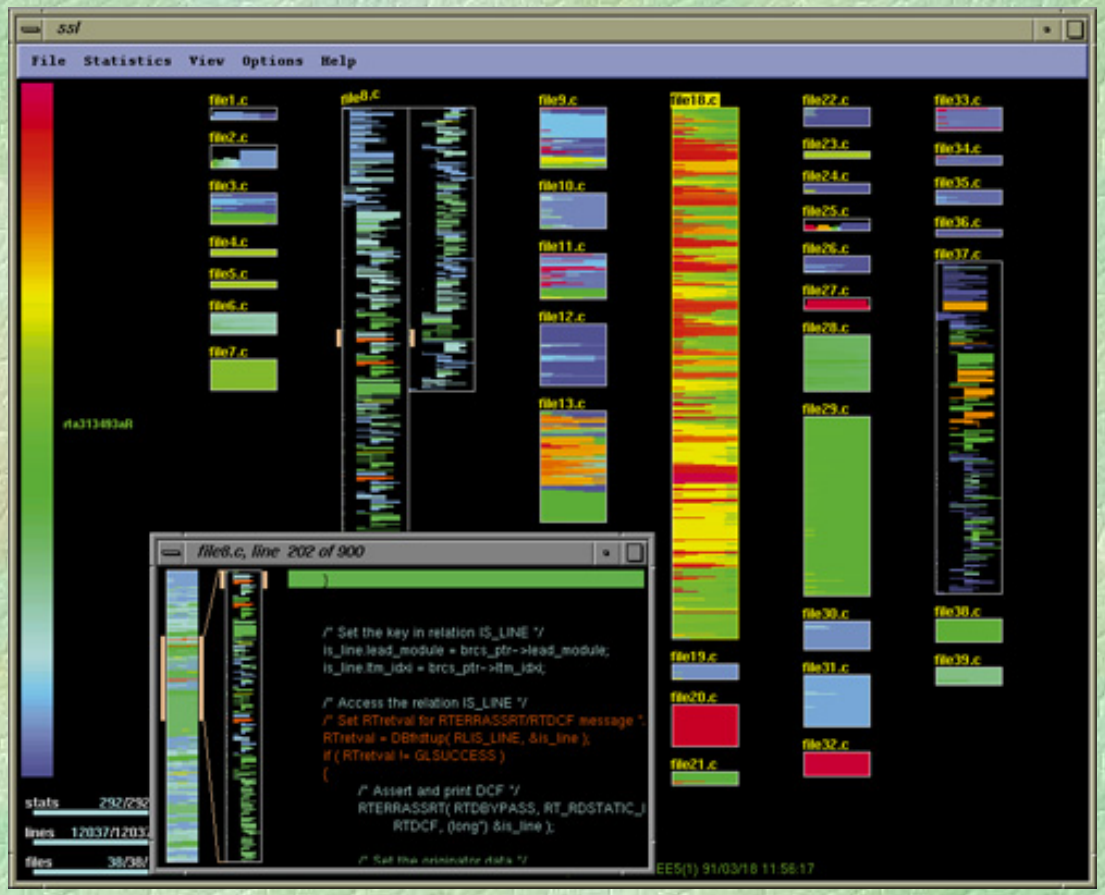
Transformation Lens



See-through lens, magnifier

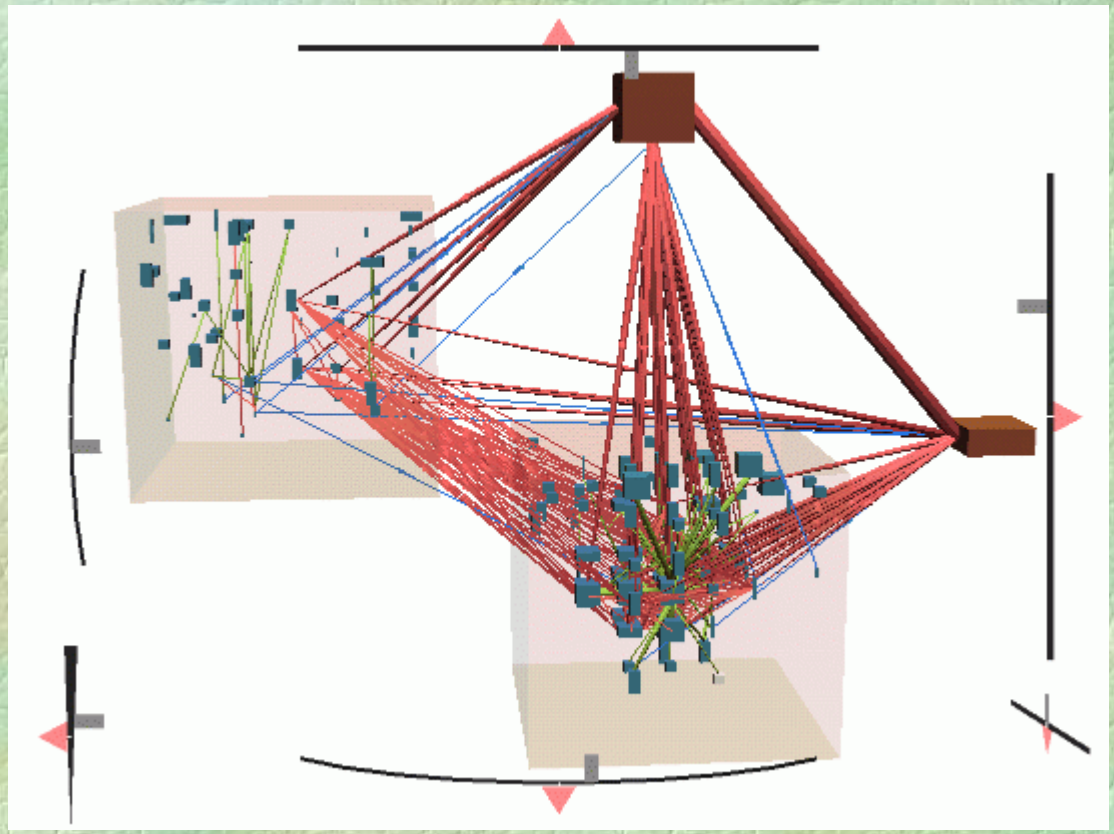
Code Properties

(age, nested loops, bug fix locations, etc.)



SeeSoft, Eick, AT&T Bell Labs

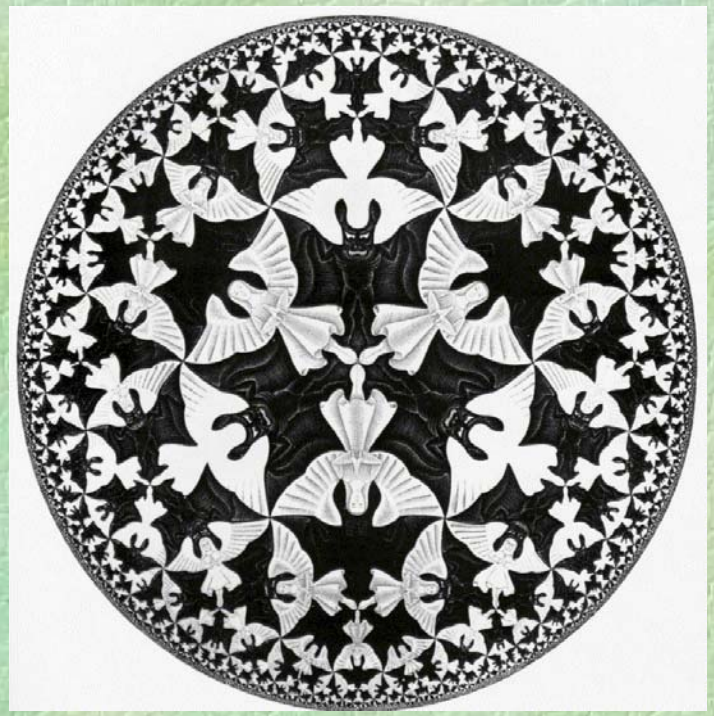
Representing Complex Structure



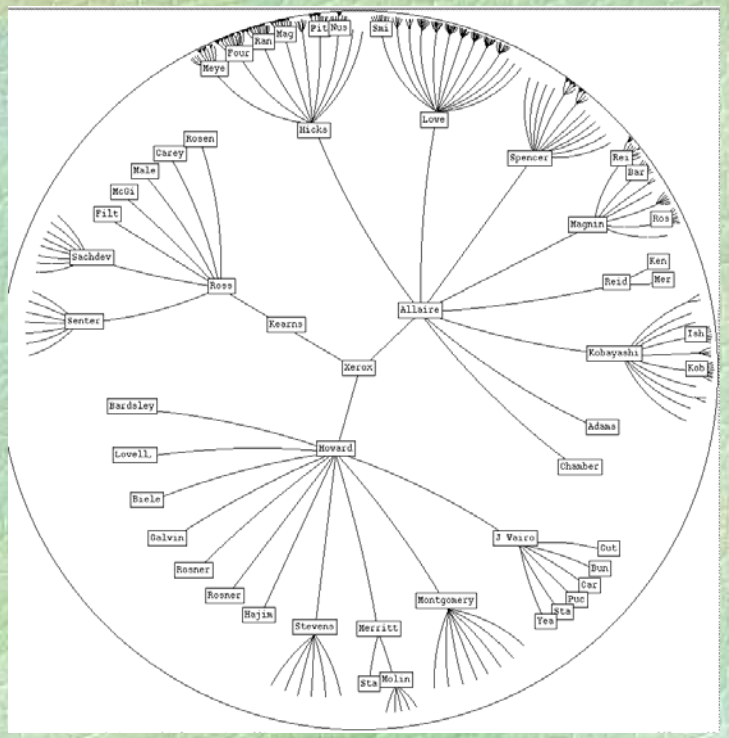
Gv3d, Franck, University of New Brunswick
(33k nodes, 34k relations)

Abstractions for Collaboration

- e.g. "Fisheye Lens" (Hyperbolic Browser; DEC, Xerox)



Escher



File System