March 10, 1998

SRC Research Report

151

Reports 100–150: The Abstracts

Compiled by James Mason

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

Systems Research Center

The charter of SRC is to advance both the state of knowledge and the state of the art in computer systems. From our establishment in 1984, we have performed basic and applied research to support Digital's business objectives. Our current work includes exploring distributed personal computing on multiple platforms, networking, programming technology, system modelling and management techniques, and selected applications.

Our strategy is to test the technical and practical value of our ideas by building hardware and software prototypes and using them as daily tools. Interesting systems are too complex to be evaluated solely in the abstract; extended use allows us to investigate their properties in depth. This experience is useful in the short term in refining our designs, and invaluable in the long term in advancing our knowledge. Most of the major advances in information systems have come through this strategy, including personal computing, distributed systems, and the Internet.

We also perform complementary work of a more mathematical flavor. Some of it is in established fields of theoretical computer science, such as the analysis of algorithms, computational geometry, and logics of programming. Other work explores new ground motivated by problems that arise in our systems research.

We have a strong commitment to communicating our results; exposing and testing our ideas in the research and development communities leads to improved understanding. Our research report series supplements publication in professional journals and conferences. We seek users for our prototype systems among those with whom we have common interests, and we encourage collaboration with university researchers.

Reports 100–150: The Abstracts

Compiled by James Mason

March 10, 1998

Abstract

This report supplements SRC Research Report 100 by giving a list of abstracts and titles for the following fifty reports in the series. The series is accessible at:

`www.research.digital.com/SRC/publications/`

Also at this web site is the SRC Technical Notes series and a feature articles section.

SRC research reports are formal publications that are subject to peer review while the technical notes series provides a fast track for the publication of work in progress, position papers, and interim results. The feature articles provide overviews of research projects for general audiences.

Information on how to access and order research reports and subscribe to publication announcements is given in section two of this document.

Contents

1 Abstracts of SRC Research Reports 100-150

- SRC Research Report 100

  The First 99 Reports
  Compiled by James Mason
  June 5, 1995. 122 pages.

  This 100th issue in the SRC Research Report series contains indexed abstracts of the previous ninety-nine, with title-page cartoons, and book and journal source information. It also documents what software SRC makes freely available for research and educational use.

- SRC Research Reports 101a and 101b

  Report 101a
  The Second Annual Video Review of Computational Geometry
  Edited by Marc H. Brown and John Hershberger
  May 4, 1993. 34 pages.

  Computational geometry concepts are often easiest to understand visually, in terms of the geometric objects they manipulate. Indeed, most papers in the field rely on diagrams to communicate the intuition behind their results. However, static figures are not always adequate.

  The accompanying videotape showcases advances in the use of algorithm animation, visualization, and interactive computing in the study of computational geometry. This report contains brief descriptions of all the segments of the videotape. The eight segments in the video cover a wide range of geometric concepts and software systems. The segments have been prepared by researchers at eight different institutions.

  Videotape 101b
  The Second Annual Video Review of Computational Geometry
  Edited by Marc H. Brown and John Hershberger
  May 4, 1993. Time: 57:53

- SRC Research Report 102

  Safe, Efficient Garbage Collection for C++
  John R. Ellis and David L. Detlefs
  June 10, 1993. 76 pages.

  We propose adding safe, efficient garbage collection to C++, eliminating the possibility of storage-management bugs and making the design of complex, object-oriented systems much easier. This can be accomplished with almost no change to the language itself and only small changes to existing implementations, while retaining compatibility with existing class libraries.

  Our proposal is the first to take a holistic, system-level approach, integrating four technologies. The language interface specifies how programmers access garbage collection through the language. An optional safe subset of the language automatically enforces the safe-use rules of garbage collection and precludes storage bugs. A variety of collection algorithms are compatible with the language interface, but some are easier to implement and more compatible with existing C++ and C implementations. Finally, code-generator safety ensures that compilers generate correct code for use with collectors.

- SRC Research Report 103

  A Coherent Distributed File Cache With Directory Write-behind
  Timothy Mann, Andrew Birrell, Andy Hisgen, Charles Jerian, Garret Swart
  June 10, 1993. 45 pages.

  Extensive caching is a key feature of the Echo distributed file system. Echo client machines maintain coherent caches of file and directory data and properties, with write-behind (delayed write-back) of all cached information. Echo specifies ordering constraints on this write-behind, enabling applications to store and maintain consistent data structures in the file system even when crashes or network faults prevent some writes from being completed. In this paper we describe the Echo cache's coherence and ordering semantics, show how they can improve the performance and consistency of applications, and explain how they are implemented. We also discuss the general problem of reliably notifying applications and users when write-behind is lost; we addressed this problem as part of the Echo design but did not find a fully satisfactory solution.

- SRC Research Report 104

  New-Value Logging in the Echo Replicated File System
  Andy Hisgen, Andrew Birrell, Charles Jerian, Timothy Mann, Garret Swart
  June 23, 1993. 39 pages.

  The Echo replicated file system uses new-value logging. Echo's use of new-value logging provides a clean separation of the internals of the system into one module that is concerned with logging and recovery, and another module that is concerned with accessing and updating the file system's on-disk structures. The logging module provides a restricted form of transaction. The restrictions simplify its implementation but impose constraints on the file system module. The file system module easily satisfies these constraints, resulting in a good match overall.

- SRC Research Report 105

  The Vesta Approach to Precise Configuration of Large Software Systems
  Roy Levin, Paul R. McJones
  June 1993. 38 pages.

  The problems of software configuration and release management limit the size of systems that we can build efficiently. Today's large systems strain the capabilities of traditional development tools. The Vesta system provides a novel repository and system builder that emphasize complete yet manageable descriptions of software components. These facilities enable Vesta to eliminate much of the manual and error-prone drudgery of system construction without enforcing a particular methodology on its users. This paper presents an overview of Vesta, followed by a series of detailed examples that illustrate how Vesta's facilities simplify the development of large systems. These examples are drawn from a year's use of Vesta by a group of about 25 researchers developing a rapidly changing software system of over 1.4 million source lines. That experience clearly demonstrates the power and practicality of the Vesta approach and its advantages over conventional tools.

- SRC Research Report 106

  The Vesta Repository: A File System Extension for Software Development
  Sheng-Yang Chiu, Roy Levin
  June 14, 1993. 34 pages.

Conventional file systems are increasingly recognized as an unsuitable basis for software configuration management, especially for large systems. While ordinary file systems have many useful properties, their facilities for managing coordinated changes that span many files are weak. To address this problem, the Vesta configuration management system implements a file system extension that tailors the file abstraction to the needs of large-scale software development.

This paper begins by presenting the essential properties required in the storage facility that underlies a successful configuration management system. It then defines a file-system-like abstraction derived from those properties and explains how it can be implemented on top of a conventional file system.

- SRC Research Report 107

  The Vesta Language for Configuration Management
  Christine B. Hanna, Roy Levin
  June 14, 1993. 60 pages.

  Current approaches to software configuration management and system building do not scale up to support large-scale software engineering; common practice involves numerous stopgap measures to work around this shortcoming. The Vesta system is designed to eliminate this problem, by providing (1) a language designed to support complete, concise system descriptions and (2) a novel caching mechanism that permits efficient system building.

  The Vesta system uses a functional programming language to describe configurations. This language provides the flexibility and power needed to describe large software components. The system descriptions are specific and complete, and include all of the sources that are used to build the system and all of the instructions that tell how the sources are composed. Only information written down in the description can influence construction of the system. Nevertheless, the descriptions are concise and easy to read and write.

  The language evaluator caches the results of evaluating function applications, which are the expensive operations in the Vesta language. Caching in Vesta is automatic and persistent. Because the language is functional and there are no side-effects, caching is conceptually straightforward. Vesta caches the result of all function applications–from those at the leaves (e.g., compiling one source file), to those in the middle (e.g., packaging up a library), all the way to the top. Caching function applications at all levels permits Vesta to build and rebuild large software systems efficiently.

- SRC Research Report 108

  Bridges: Tools to Extend the Vesta Configuration Management System
  Mark R. Brown, John R. Ellis
  June 14, 1993. 42 pages.

  Vesta is a highly flexible configuration management system that supports large-scale development. Vesta provides a repository of immutable objects and a functional programming language for writing concise yet complete descriptions of configurations.

  A Vesta bridge is a set of related functions and data types provided by tool builders to a Vesta environment. For instance, a C bridge might include a function for compiling C sources and a function for linking compiled C sources into executable images.

  Vesta has supported development on a significant scale. The Vesta prototype included several low-aspiration bridges that encapsulated existing tools without modification; these bridges were straightforward to write. Vesta also included one high-aspiration bridge, Vulcan, a compiler server based on abstract-syntax trees. Vulcan gained both functionality and performance from its integration with Vesta. Both types of bridge benefited from Vesta's single, uniform naming facility that replaced ad hoc name spaces of traditional environments.

  Bridges themselves are described and configured within Vesta. This allows tool builders to provide consistent collections of tools, control their evolution, and manage their installation using Vesta.

- SRC Research Report 109

  Formal Parametric Polymorphism
  Martín Abadi, Luca Cardelli, Pierre-Louis Curien
  July 15, 1993. 43 pages.

  A polymorphic function is parametric if its behavior does not depend on the type at which it is instantiated. Starting with Reynolds's work, the study of parametricity is typically semantic. In this paper, we develop a syntactic approach to parametricity, and a formal system that embodies this approach, called system R . Girard's system F deals with terms and types; R is an extension of F that deals also with relations between types.

  In R, it is possible to derive theorems about functions from their types, or "theorems for free", as Wadler calls them. An easy "theorem for free" asserts that the type $\forall(X)\, X \rightarrow \mathit{Bool}$ contains only constant functions; this is not provable in F. There are many harder and more substantial examples. Various metatheorems can also be obtained, such as a syntactic version of Reynolds's abstraction theorem.

- SRC Research Reports 110a and 110b

  Report 110a
  Algorithm Animation Using 3D Interactive Graphics
  Marc H. Brown and Marc A. Najork
  September 15, 1993. 19 pages.

  This report describes a variety of 3D interactive graphics techniques for visualizing programs. The third dimension provides an extra degree of freedom for conveying information, much as color adds to black-and-white images, animation adds to static images, and sound adds to silent animations. The examples in this report illustrate three fundamental uses of 3D: for providing additional information about objects that are intrinsically two-dimensional, for uniting multiple views, and for capturing a history of execution. The application of dynamic three-dimensional graphics to program visualization is largely unexplored.

  Videotape 110b
  Algorithm Animation Using 3D Interactive Graphics
  Edited by Marc H. Brown and Marc A. Najork
  September 15, 1993. Time: 8:20

- SRC Research Report 111

  The Echo Distributed File System
  Andrew D. Birrell, Andy Hisgen, Chuck Jerian, Timothy Mann, and Garret Swart
  September 10, 1993. 22 pages.

  Echo is an ambitious distributed file system. It was designed around a truly global name space. It uses a coherent caching algorithm. It is fault tolerant. And it is real–it was the primary file system for a large group of researchers. Its novel aspects include an extensible "junction" mechanism for global naming; extensive write-behind with ordering semantics that allow applications to maintain invariants without resorting to synchronous writes; and fault tolerance mechanisms that are highly configurable and that tolerate network partitions. It was designed with the intention that its performance could be as good as a local file system, while supporting large numbers of clients per server. Its reliability was designed to be higher than other distributed file systems, and higher than centralized systems. It was designed to work well in arbitrarily large networks.

- SRC Research Report 112

  Availability in the Echo File System
  Garret Swart, Andrew Birrell, Andy Hisgen, and Timothy Mann
  September 10, 1993. 43 pages.

  The Echo file system project explored several issues in the design and implementation of distributed file systems. This paper describes the aspects of the Echo design that are related to providing high availability. These aspects include the provision of redundant components (replicated disks and backup servers), the replication of information, and recovery from failures. Further, we discuss some less obvious mechanisms needed for providing truly high availability: load control, dynamic reconfiguration of the system, and the detection and reporting of faults. Finally, we discuss some of the impact of our availability mechanisms on application software.

- SRC Research Report 113

  Some Useful Modula-3 Interfaces
  Jim Horning, Bill Kalsow, Paul McJones, Greg Nelson
  December 25, 1993. 103 pages.

  This manual describes a collection of interfaces defining abstractions that SRC's programmers have found useful over a number of years of experience with Modula-3 and its precursors. We hope the interfaces will be useful as a "starter kit" of abstractions, and as a model for designing and specifying abstractions in Modula-3.

- SRC Research Report 114

  Automated Proofs of Object Code for a Widely Used Microprocessor
  Yuan Yu
  October 5, 1993. 122 pages.

  Computing devices can be specified and studied mathematically. Formal specification of computing devices has many advantages; it provides a precise characterization of the computational model, and allows for mathematical reasoning about models of the computing devices and programs executed on them. While there has been a large body of research on program proving, work has almost exclusively focused on programs written in high-level programming languages. Here we address the important but largely ignored problem of machine-code program proving. This work formally describes a substantial subset of the MC68020, a widely used microprocessor built by Motorola, within the mathematical logic of the automated reasoning system Nqthm a.k.a. the Boyer-Moore Theorem Proving System. Based on this formal model, we mechanized a mathematical theory to automate reasoning about object code programs. We then mechanically checked the correctness of MC68020 object code programs for binary search, Hoare's Quick Sort, the Berkeley Unix C string library, and other well-known algorithms. The object code for these examples was generated using the Gnu C, the Verdix Ada, and the AKCL Common Lisp compilers.

- SRC Research Report 115

  Network Objects
  Andrew Birrell, Greg Nelson, Susan Owicki, and Edward Wobber
  February 28, 1994. Revised December 4, 1995. 48 pages.

  A network object is an object whose methods can be invoked over a network. This report describes the design and implementation of a network objects system for Modula-3. The system is novel for its overall simplicity. The report includes a thorough description of realistic marshaling algorithms for network objects, precise informal specifications of the major internal interfaces, preliminary experience, and performance results.

- SRC Research Report 116

  Distributed Garbage Collection for Network Objects
  Andrew Birrell, David Evers, Greg Nelson, Susan Owicki, Edward Wobber
  December 15, 1993. 18 pages.

  In this report we present a fault-tolerant and efficient algorithm for distributed garbage collection and prove its correctness. The algorithm is a generalization of reference counting; it maintains a set of identifiers for processes with references to an object. The set is maintained with pair-wise communication between processes, so no global synchronization is required. The primary cost for maintaining the set is one remote procedure call when an object reference is transferred to a new process for the first time. The distributed collector collaborates with the local collector in detecting garbage; any local collector may be used, so long as it can be extended to provide notification when an object is collected. In fact, the distributed collector could be used without a local collector; in that case, the programmer would insert explicit "dispose" commands to release an object. The algorithm was designed and implemented as part of the Modula-3 network objects system, but it should be suitable for a wide range of applications. It tolerates communication and process failure, and can reclaim the space for objects held by a crashed process. The algorithm balances functionality, performance, and fault-tolerance in a way that makes it highly practical to use in implementing distributed systems.

- SRC Research Report 117

  Authentication in the Taos Operating System
  Edward Wobber, Martín Abadi, Mike Burrows, and Butler Lampson
  December 10, 1993. 38 pages.

  We describe a design for security in a distributed system and its implementation. In our design, applications gain access to security services through a narrow interface. This interface provides a notion of identity that includes simple principals, groups, roles, and delegations. A new operating system component manages principals, credentials, and secure channels. It checks credentials according to the formal rules of a logic of authentication. Our implementation is efficient enough to support a substantial user community.

- SRC Research Report 118

  Conjoining Specifications
  Martín Abadi, Leslie Lamport
  December 7, 1993. 65 pages.

  We show how to specify components of concurrent systems. The specification of a system is the conjunction of its components' specifications. Properties of the system are proved by reasoning about its components. We consider both the decomposition of a given system into parts, and the composition of given parts to form a system.

- SRC Research Report 119

  How to Write a Long Formula
  Leslie Lamport
  December 25, 1993. 6 pages.

  Standard mathematical notation works well for short formulas, but not for the longer ones often written by computer scientists. Notations are proposed to make one or two-page formulas easier to read and reason about.

- SRC Research Report 120

  Dynamic Typing in Polymorphic Languages
  Martín Abadi, Luca Cardelli, Benjamin Pierce, Didier Remy
  January 26, 1994. 22 pages.

  There are situations in programming where some dynamic typing is needed, even in the presence of advanced static type systems. We investigate the interplay of dynamic types with other advanced type constructions, discussing their integration into languages with explicit polymorphism (in the style of system F), implicit polymorphism (in the style of ML), abstract data types, and subtyping.

- SRC Research Report 121

  Extensible Syntax with Lexical Scoping
  Luca Cardelli, Florian Matthes, and Martín Abadi
  February 21, 1994. 35 pages.

  A frequent dilemma in programming language design is the choice between a language with a rich set of notations and a small, simple core language. We address this dilemma by proposing extensible grammars, a syntax-definition formalism for incremental language extensions and restrictions.

  The translation of programs written in rich object languages into a small core language is defined via syntax-directed patterns. In contrast to macro-expansion and program-rewriting tools, our extensible grammars respect scoping rules. Therefore, we can introduce binding constructs while avoiding problems with unwanted name clashes.

  We develop extensible grammars and illustrate their use by extending the lambda calculus with let-bindings, conditionals, and constructs from database programming languages, such as SQL query expressions. We then give a formal description of the underlying rules for parsing, transformation, and substitution. Finally, we sketch how these rules are exploited in an implementation of a generic, extensible parser package.

- SRC Research Report 122

  Obliq: A Language with Distributed Scope
  Luca Cardelli
  June 3, 1994. 64 pages.

  Obliq is a lexically-scoped untyped interpreted language that supports distributed object-oriented computation. An Obliq computation may involve multiple threads of control within an address space, multiple address spaces on a machine, heterogeneous machines over a local network, and multiple networks over the Internet. Obliq objects have state and are local to a site. Obliq computations can roam over the network, while maintaining network connections.

- SRC Research Report 123

  Inside Hector: The Systems View
  Loretta Guarino Reid and James R. Meehan
  April 28, 1994. 43 pages.

  Over a period of two and a half years, a team from the Systems Research Center designed, built, and revised a set of software tools for the dictionary division of Oxford University Press. Many aspects of this project were novel, including the approach to lexicography, the software environment, the problems of scale, and the demands of high performance and high reliability. In this paper, two members of the team describe some of the systems problems they faced in building these tools, and the solutions they devised to solve them.

- SRC Research Report 124

  A Block-sorting Lossless Data Compression Algorithm
  M. Burrows and D. J. Wheeler
  May 10, 1994. 18 pages.

  We describe a block-sorting, lossless data compression algorithm, and our implementation of that algorithm. We compare the performance of our implementation with widely available data compressors running on the same hardware.

The algorithm works by applying a reversible transformation to a block of input text. The transformation does not itself compress the data, but reorders it to make it easy to compress with simple algorithms such as move-to-front coding.

Our algorithm achieves speed comparable to algorithms based on the techniques of Lempel and Ziv, but obtains compression close to the best statistical modelling techniques. The size of the input block must be large (a few kilobytes) to achieve good compression.

- SRC Research Report 125

  Prudent Engineering Practice for Cryptographic Protocols
  Martín Abadi and Roger Needham
  June 1, 1994. 25 pages.

  We present principles for designing cryptographic protocols. The principles are neither necessary nor sufficient for correctness. They are however helpful, in that adherence to them would have prevented a number of published errors.

  Our principles are informal guidelines; they complement formal methods, but do not assume them. In order to demonstrate the actual applicability of these guidelines, we discuss some instructive examples from the literature.

- SRC Research Report 126

  The 1993 SRC Algorithm Animation Festival
  Marc H. Brown
  July 29, 1994. 31 pages.

  This report describes the 1993 SRC Algorithm Animation Festival. The festival continues an experiment in developing algorithm animations by non-experts, started the previous year, and described in SRC Research Report 98. This year nineteen researchers at Digital Equipment Corporation's Systems Research Center worked for two weeks on animating algorithms. Most of the participants had little (if any) experience writing programs that involved graphics. This report explains why we organized the festival, and describes the logistics of the festival and the advances in our algorithm animation system. This report presents the complete code for a simple, but non-trivial, animation of first-fit binpacking. Finally, this report contains snapshots from the animations produced during the festival.

- SRC Research Report 127

  TLA in Pictures
  Leslie Lamport
  September 1, 1994. 20 pages.

  Predicate-action diagrams, which are similar to standard state-transition diagrams, are interpreted as formulas of TLA (the Temporal Logic of Actions). We explain how these diagrams can be used to describe aspects of a specification, even when the complete specification cannot be written as a diagram, and to illustrate proofs.

- SRC Research Reports 128a and 128b

  Report 128a
  A Library for Visualizing Combinatorial Structures
  Marc A. Najork and Marc H. Brown
  September 1, 1994. 20 pages.

  This paper describes ANIM3D, a 3D animation library targeted at visualizing combinatorial structures. In particular, we are interested in algorithm animation. Constructing a new view for an algorithm typically takes dozens of design iterations, and can be very time-consuming. Our library eases the programmer's burden by providing high-level constructs for performing animations, and by offering an interpretive environment that eliminates the need for recompilations. This paper also illustrates ANIM3D's expressiveness by developing a 3D animation of Dijkstra's shortest-path algorithm in just 70 lines of code.

  Videotape 128b
  A Library for Visualizing Combinatorial Structures
  Marc A. Najork and Marc H. Brown
  September 1, 1994. Time: 6:22

- SRC Research Report 129

  Obliq-3D Tutorial and Reference Manual
  Marc A. Najork
  December 1, 1994. 110 pages.

  Obliq-3D is an interpreted language that is embedded into the 3D animation system Anim3D. Anim3D is based on a few simple, yet powerful constructs that allow a programmer to describe three-dimensional scenes and animations of such scenes. Obliq-3D, by virtue of its interpretive nature, provides the programmer with a fast turnaround environment. The combination of simplicity and fast turnaround allows application programmers to construct non-trivial animations quickly and easily.

The first half of this report contains a tutorial to Obliq-3D, which develops the various concepts of the animation system. The second part contains a reference manual, which describes the functionality of Obliq-3D module by module.

- SRC Research Reports 130a and 130b

  Report 130a
  Visual Obliq: A System for Building Distributed, Multi-User Applications by Direct Manipulation
  Krishna Bharat and Marc H. Brown
  October 31, 1995. 29 pages.

  This report describes Visual Obliq, a user interface development environment for constructing distributed, multi-user applications. Applications are created by designing the interface with a GUI-builder and embedding callback code in an interpreted language, in much the same way as one would build a traditional (non-distributed, single-user) application with a modern user interface development environment. The resulting application can be run from within the GUI-builder for rapid turnaround or as a stand-alone executable. The Visual Obliq runtime provides abstractions and support for issues specific to distributed computing, such as replication, sharing, communication, and session management. We believe that the abstractions provided, the simplicity of the programming model, the rapid turnaround time, and the applicability to heterogeneous environments, make Visual Obliq a viable tool for authoring distributed applications and groupware.

  Videotape 130b
  Building a Distributed Application Using Visual Obliq
  Krishna Bharat and Marc H. Brown
  November 1, 1994. Time: 8:50

- SRC Research Reports 131a and 131b

  Report 131a
  The Juno-2 Constraint-Based Drawing Editor
  Allan Heydon and Greg Nelson
  December, 1994. 19 pages.

  Constraints are an important enabling technology for interactive graphics applications. However, today's constraint-based systems are plagued by several limitations, and constraints have yet to live up to their potential.

  Juno-2 is a constraint-based double-view drawing editor that addresses some of these limitations. Constraints in Juno-2 are declarative, and they can include non-linear functions and ordered pairs. Moreover, the Juno-2 solver is not limited to acyclic constraint systems. Juno-2 also includes a powerful extension language that allows users to define new constraints. The system demonstrates that fast constraint solving is possible with a highly extensible, fully declarative constraint language.

  The report describes what it is like to use Juno-2, outlines the methods that Juno-2 uses to solve constraints, and discusses its performance.

  Videotape 131b
  The Juno-2 Constraint-Based Drawing Editor
  Allan Heydon and Greg Nelson
  February 1, 1995. Time: 13:32

- SRC Research Report 132

  Processes are in the Eye of the Beholder
  Leslie Lamport
  December 25, 1994. 23 pages.

  A two-process algorithm is shown to be equivalent to an N-process one, illustrating the insubstantiality of processes. A completely formal equivalence proof in TLA (the Temporal Logic of Actions) is sketched.

- SRC Research Reports 133a and 133b

  Report 133a
  The Third Annual Video Review of Computational Geometry
  Marc H. Brown and John Hershberger December 30, 1994. 34 pages.

Computational geometry concepts are often easiest to understand visually, in terms of the geometric objects they manipulate. Indeed, most papers in the field rely on diagrams to communicate the intuition behind their results. However, static figures are not always adequate.

Videotape 133b
The Third Annual Video Review of Computational Geometry
Marc H. Brown and John Hershberger
December 30, 1994. Time: 54:10

- SRC Research Reports 134a and 134b

Report 134a
From Quadrangular Sets to the Budget Matroids
Lyle Ramshaw and Jim Saxe
May, 1995. 162 pages.

The complete quadrilateral is a configuration with six points and four lines in the projective plane. It is a classical result that, given the six roots of three quadratic polynomials, we can use the complete quadrilateral (or its dual, the complete quadrangle) to test geometrically whether or not those three quadratics are linearly dependent. But does there exist some configuration that provides an analogous geometric test for the linear dependence of four cubic polynomials?

Yes, there is such a cubic analog of the complete quadrilateral; it has twelve points, two lines, and thirteen planes in 3-space. In fact, the complete quadrilateral and its cubic analog are just two members of a large family of intriguing configurations that are defined in this monograph, using the notion of a "budget". The study of these "budget configurations" combines old-fashioned geometry with modern combinatorics—from null systems to matroids.

Why didn't the classical geometers discover the budget configurations long ago? Perhaps because they required their configurations to have a certain numeric symmetry that a typical budget configuration doesn't have.

The videotape animates the cubic analog of the complete quadrilateral, as well as some other budget configurations that live in the plane or in 3-space. By doing so, it tries both to give non-mathematicians some hint of these discoveries and to lure mathematicians into reading the monograph.

Videotape 134b
Introducing the Budget Configurations
Lyle Ramshaw
May 1995. Time: 45:57

- SRC Research Reports 135a and 135b

Report 135a
DeckScape: An Experimental Web Browser
Marc H. Brown and Robert A. Shillner
March 1, 1995. 13 pages.

This report describes DeckScape, an experimental World-Wide Web browser based on a "deck" metaphor. A deck consists of a collection of Web pages, and multiple decks can exist on the screen at once. As the user traverses links, new pages appear on top of the current deck. Retrievals are done using a background thread, so all visible pages in any deck are active at all times. Users can move and copy pages between decks, and decks can be used as a general-purpose way to organize material, such as hotlists, query results, and breadth-first expansions.

Videotape 135b
The Deckscape Web Browser
Marc H. Brown and Robert A. Shillner
January 8, 1996. Time: 6:15

- SRC Research Report 136

A Functional Specification of the Alpha AXP Shared Memory Model
Manfred Broy
April 3, 1995. 34 pages.

We give a functional specification of the Alpha AXP architecture with special emphasis on the Alpha Shared Memory Model. We keep the specification as abstract as possible and modular in the sense that we provide an independent description of the processors and the memory. We show how to handle a number of critical aspects of the Alpha architecture within the functional model, such as the specification of basic assumptions about the behavior of the processors and the exclusion of causal loops. We use the model for specifying the notion of lookahead and shortcut optimization for the behaviors of the processors. This allows us to define the concept of correct processor behavior by using the conservative sequential behavior as a reference. Finally, we extend the model to the constructs for synchronization in the Alpha architecture and include the instructions “read locked” as well as “store conditional”.

- SRC Research Report 137

  Proving Possibility Properties
  Leslie Lamport July 4, 1995, Revised December 1, 1997. 13 pages.

  A method is described for proving “always possibly” properties of specifications in formalisms with linear-time trace semantics. It is shown to be relatively complete for TLA (Temporal Logic of Actions) specifications.

- SRC Research Report 138

  Migratory Applications
  Krishna A. Bharat and Luca Cardelli
  February 15, 1996. 24 pages.

  We introduce a new genre of user interface applications that can migrate from one machine to another, taking their user interface and application contexts with them, and continue from where they left off. Such applications are not tied to one user or one machine, and can roam freely over the network, rendering service to a community of users, gathering human input and interacting with people. We envisage that this will support many new agent-based collaboration metaphors. The ability to migrate executing programs has applicability to mobile computing as well. Users can have their applications travel with them, as they move from one computing environment to another. We present an elegant programming model for creating migratory applications and describe an implementation. The biggest strength of our implementation is that the details of migration are completely hidden from the application programmer; arbitrary user interface applications can be migrated by a single “migration” command. We address system issues such as robustness, persistence and memory usage, and also human factors relating to application design, the interaction metaphor and safety.

- SRC Research Reports 139a and 139b

  Report 139a
  WebCard: Integrated and Uniform Access to Mail, News, and the Web
  Marc H. Brown
  July 15, 1996. 10 pages.

  This report describes WebCard, an integrated mail/news reader and Web browser. As a mail/news reader, WebCard is fairly conventional; the innovation is that Web pages are fully integrated into the mail/news reader. The user interface is based on folders, which can contain mail messages, news articles or Web pages. Folders can be used to organize material and to present the pages returned by commands such as "search" and "auto surf".

  Videotape 139b
  The WebCard Web Browser
  Marc H. Brown
  July 15, 1996. Time: 7:54

- SRC Research Report 140

  Zippers: A Focus+Context Display of Web Pages
  Marc H. Brown and William E. Weihl
  May 22, 1996. 8 pages.

  This report describes zippers, an application of outline-processor technology to the display of Web pages. Zippers allow users to expand and contract selected sections of a document, thereby displaying simultaneously the contents of individual sections of a document as well as its overall structure. Zippers can be implemented either directly in a Web browser or by a proxy (and consequently used by any off-the-shelf Web browser); in either case, no changes to HTML source files are required.

- SRC Research Reports 141a and 141b

  Report 141a
  Distributed Active Objects
  Marc H. Brown and Marc N.Najork
  April 15, 1996. 21 pages.

  Many Web browsers now offer some form of active objects, written in a variety of languages, and the number of types of active objects are growing daily in interesting and innovative ways. This report describes our work on Oblets, active objects that are distributed over multiple machines. Oblets are written in Obliq, an object-oriented scripting language for distributed computation. The high-level support provided by Oblets makes it easy to write collaborative and distributed applications.

Videotape 141b
Distributed Active Objects
Marc H. Brown and Marc A. Najork
May 1, 1996. Time: 9:08

- SRC Research Report 142

  Collaborative Active Textbooks: A Web-Based Algorithm Animation System for an Electronic Classroom
  Marc H. Brown and Marc A. Najork
  May 31, 1996. 26 pages.

  This report describes CAT, a Web-based algorithm animation system. CAT augments the expressive power of Web pages for publishing passive multimedia information with a full-fledged interactive algorithm animation system. It improves on previous Web-based algorithm animations by providing a framework that makes it easy to construct new animations, including those that involve multiple views. Because views of the same running algorithm may reside on different machines, CAT is particularly well-suited for electronic classrooms. This strategy is an improvement over the electronic classroom systems we are aware of, which simply display the same X window on multiple machines. We believe our framework generalizes to electronic textbooks in arbitrary domains.

- SRC Research Report 143

  To Provide or To Bound: Sampling in Fully Dynamic Graph Algorithms
  Monika R. Henzinger and Mikkel Thorup
  October 8, 1996. 11 pages.

  In dynamic graph algorithms the following provide-or-bound problem has to be solved quickly: Given a set S containing a subset R and a way of generating random elements from S testing for membership in R, either (i) provide an element of R or (ii) give a (small) upper bound on the size of

R that holds with high probability. We give an optimal algorithm for this problem.

This algorithm improves the time per operation for various dyamic graph algorithms by a factor of $O(\log n)$. For example, it improves the time per update for fully dynamic connectivity from $O(\log^3 n)$ to $O(\log^2 n)$.

- SRC Research Report 144

  Program Fragments, Linking, and Modularization
  Luca Cardelli
  February 15, 1997. 25 pages.

  Module mechanisms have received considerable theoretical attention, but the associated concepts of separate compilation and linking have not been emphasized. Anomalous module systems have emerged in functional and object-oriented programming where software components are not separately typecheckable and compilable. In this paper we provide a context where linking can be studied, and separate compilability can be formally stated and checked. We propose a framework where each module is separately compiled to a self-contained entity called a linkset; we show that separately compiled, compatible modules can be safely linked together.

- SRC Research Report 145

  Modularity in the Presence of Subclassing
  Raymie Stata
  April 28, 1997. 98 pages.

  Classes are harder to subclass than they need be. This report addresses this problem, showing how to design classes that are more modular and easier to subclass without sacrificing the extensibility that makes subclassing useful to begin with.

  We argue that a class should have two interfaces, an "instance interface" used by programmers manipulating instances of the class, and a "specialization interface" used by programmers building subclasses of the class. Instance interfaces are relatively well understood, but design principles for specialization interfaces are not.

  In the context of single inheritance, we argue that specialization interfaces should be partitioned into "class components". A class component groups part of a class's state together with methods to maintain that state. Class components establish abstraction boundaries within classes, allowing modular replacement of components by subclasses. Achieving this replaceability requires reasoning about each component as an independent unit that depends only on the specifications of other components and not on their implementations.

We introduce the concept of “abstract representation” to denote the view of a class’s state given in its specialization interface. This view is more detailed than the view used to describe instances of the class, revealing details that describe the interfaces between class components. It is less detailed than the actual implementation, hiding implementation details that should not be exposed even to specializers.

We also consider multiple inheritance, specifically, Snyder’s model of encapsulated multiple inheritance. We advocate separating class components into individual classes called “mixins”. Instantiable classes are built by combining multiple mixins. With the mixin style of design, class hierarchies have more classes than in equivalent single-inheritance designs. These classes have smaller, simpler interfaces and can be reused more flexibly.

To explore the impact our ideas might have on program design, we consider classes from existing libraries in light of the proposed single- and multiple-inheritance methodologies. To explore the impact our ideas might have on language design, we present two different extensions to Java, one that provides a level of static checking for single-inheritance designs, and another that adds the encapsulated model of multiple inheritance.

- SRC Research Report 146

  Studies of Windows NT Performance using Dynamic Execution Traces
  Sharon E. Perl and Richard L. Sites
  April 4, 1997. 31 pages.

  We studied two aspects of the performance of Windows NT processor bandwidth requirements for memory accesses in a uniprocessor system running benchmark and commercial applications, and locking behavior of a commercial database on a small-scale multiprocessor. Our studies are based on full dynamic execution traces of the systems, which include all instructions executed by the operating system and applications over periods of a few seconds (enough time to allow for significant computation). The traces were obtained on Alpha PCs, using a new software tool called PatchWrx that takes advantage of the Alpha architecture’s PAL-code layer to implement efficient, comprehensive system tracing. Because the Alpha version of Windows NT uses substantially the same code base as other versions, and therefore executes nearly the same sequence of calls, basic blocks, and data structure accesses, we believe our conclusions are relevant for non-Alpha systems as well. This paper describes our performance studies and interesting aspects of PatchWrx.

We conclude from our studies that processor bandwidth can be a first-order bottleneck to achieving good performance. This is particularly apparent when studying commercial benchmarks. Operating system code and data structures contribute disproportionately to the memory access load. We also found that operating system software lock contention was a factor preventing the database benchmark from scaling up on the small multiprocessor, and that the cache coherence protocol employed by the machine introduced more cache interference than necessary.

- SRC Research Report 147

  Should Your Specification Language Be Typed?
  Leslie Lamport and Lawrence C. Paulson
  May 1, 1997. 30 pages.

  Most specification languages have a type system. The languages used in some popular textbooks have half-baked type systems that are described informally and never spelled out in detail. Such type systems tend to have unexpected consequences, if not outright inconsistencies. Set theory can serve as the basis for a specification language without types. This possibility, which has been widely overlooked, offers many advantages. Set theory is simpler and more flexible than most typed formalisms. Polymorphism, overloading, and subtyping can make a type system more powerful, but at the cost of increased complexity, and such refinements can never attain the flexibility of having no types at all. Typed formalisms have advantages too, stemming from the power of mechanical type checking. While types serve little purpose in hand proofs, they do help with mechanized proofs. In the absence of verification, type checking can catch errors in specifications. It may be possible to have the best of both worlds by adding typing annotations to an untyped specification language.

  We consider only specification languages, not programming languages.

- SRC Research Report 148

  Service Combinators for Web Computing
  Luca Cardelli and Rowan Davies
  June 1, 1997. 15 pages.

  The World-Wide Web is rich in content and services, but access to these resources must be obtained mostly through manual browsers. We would like to be able to write programs that reproduce human browsing behavior, including reactions to slow transmission-rates and failures on many simultaneous links. We thus introduce a concurrent model that directly incorporates the notions of failure and rate of communication, and then describe programming constructs based on this model.

- SRC Research Report 149

  A Calculus for Cryptographic Protocols: The Spi Calculus
  Martín Abadi and Andrew D. Gordon
  January 25, 1998. 110 pages.

  We introduce the spi calculus, an extension of the pi calculus designed for describing and analyzing cryptographic protocols. We show how to use the spi calculus, particularly for studying authentication protocols. The pi calculus (without extension) suffices for some abstract protocols; the spi calculus enables us to consider cryptographic issues in more detail. We represent protocols as processes in the spi calculus and state their security properties in terms of coarse-grained notions of protocol equivalence.

- SRC Research Report 150

  Smooth Scheduling in a Cell-Based Switching Network
  Thomas L. Rodeheffer and James B. Saxe
  February 19, 1998. 36 pages.

  This paper describes a method for scheduling data cell traffic through a crossbar switch in such a way as to guarantee that the time slots for each flow, and optionally for certain aggregates of flows, are approximately uniformly distributed throughout the duration of the scheduling frame. Such a "smooth" schedule achieves reductions in both the latency of data flows and the need for associated buffering memory. The method can be used to compute smooth schedules even under conditions of maximum load, where the bandwidth re-quirements of the flows to be scheduled are sufficient to consume the entire capacity of the switch. This method has been implemented for a working high-speed ATM switching network, AN2. We describe the implementation, its performance, and possible future improvements.

2 Ordering Information

2.1 Reports

SRC Research Reports are available from SRC's external publications web site:

`http://www.research.digital.com/SRC/publications/`

They are also available via anonymous ftp from internet node:

`gatekeeper.dec.com` (16.1.0.2)

The ftp path to them is:

`/pub/DEC/SRC/research-reports/`

For hardcopy orders of SRC research reports, please send electronic mail to the address below or submit your order via the web site. Be sure to include your full postal address and the number of the report you wish to receive.

`src-report@pa.dec.com`

Orders may also be placed by sending requests to:

Report Distribution
Digital Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301

If you'd like to be notified by electronic mail whenever a new publication of any kind is made available on SRC's web, please send email to:

`src-report@pa.dec.com`

with the words: "add to pub email" in the subject line of your message.

2.2 Videotapes

Videotapes accompanying several of these fifty reports are identified in the abstracts section. Their reference numbers are 101b, 110b, 128b, 130b, 131b, 133b, 134b, 135b, 139b, and 141b. All are available in NTSC, PAL, and SECAM formats. Please specify which format you require.

3 List of SRC Research Reports 100–150

- 100. *The First 99 Reports*
  Compiled by James Mason

- 101a. *The Second Annual Video Review of Computational Geometry*
  Edited by Marc H. Brown and John Hershberger

- 101b. (Video)
  The Second Annual Video Review of Computational Geometry
  Edited by Marc H. Brown and John Hershberger

- 102. *Safe, Efficient Garbage Collection for C++*
  John R. Ellis and David L. Detlefs

- 103. *A Coherent Distributed File Cache With Directory Write-behind*
  Timothy Mann, Andrew Birrell, Andy Hisgen, Charles Jerian, Garret Swart

- 104. *New-Value Logging in the Echo Replicated File System*
  Andy Hisgen, Andrew Birrell, Charles Jerian, Timothy Mann, Garret Swart

- 105. *The Vesta Approach to Precise Configuration of Large Software Systems*
  Roy Levin, Paul R. McJones

- 106. *The Vesta Repository: A File System Extension for Software Development*
  Sheng-Yang Chiu, Roy Levin

- 107. *The Vesta Language for Configuration Management*
  Christine B. Hanna, Roy Levin

- 108. *Bridges: Tools to Extend the Vesta Configuration Management System*
  Mark R. Brown, John R. Ellis

- 109. *Formal Parametric Polymorphism*
  Martín Abadi, Luca Cardelli, Pierre-Louis Curien

- 110a. *Algorithm Animation Using 3D Interactive Graphics*
  Marc H. Brown and Marc A. Najork

- 110b.(Video)
  Algorithm Animation Using 3D Interactive Graphics
  Edited by Marc H. Brown and Marc A. Najork

- 111. *The Echo Distributed File System*
  Andrew D. Birrell, Andy Hisgen, Chuck Jerian, Timothy Mann,
  Garret Swart

- 112. *Availability in the Echo File System*
  Garret Swart, Andrew Birrell, Andy Hisgen, and Timothy Mann

- 113. *Some Useful Modula-3 Interfaces*
  Jim Horning, Bill Kalsow, Paul McJones, Greg Nelson

- 114. *Automated Proofs of Object Code for a Widely Used Microprocessor*
  Yuan Yu

- 115. *Network Objects*
  Andrew Birrell, Greg Nelson, Susan Owicki, and Edward Wobber

- 116. *Distributed Garbage Collection for Network Objects*
  Andrew Birrell, David Evers, Greg Nelson, Susan Owicki,
  Edward Wobber

- 117. *Authentication in the Taos Operating System*
  Edward Wobber, Martín Abadi, Mike Burrows, and Butler Lampson

- 118. *Conjoining Specifications*
  Martín Abadi, Leslie Lamport

- 119. *How to Write a Long Formula*
  Leslie Lamport

- 120. *Dynamic Typing in Polymorphic Languages*
  Martín Abadi, Luca Cardelli, Benjamin Pierce, Didier Remy

- 121. *Extensible Syntax with Lexical Scoping*
  Luca Cardelli, Florian Matthes, and Martín Abadi

- 122. *Obliq: A Language with Distributed Scope*
  Luca Cardelli

- 123. *Inside Hector: The Systems View*
  Loretta Guarino Reid and James R. Meehan

- 124. *A Block-sorting Lossless Data Compression Algorithm*
  M. Burrows and D. J. Wheeler

- 125. *Prudent Engineering Practice for Cryptographic Protocols*
  Martín Abadi and Roger Needham

- 126. *The 1993 SRC Algorithm Animation Festival*
  Marc H. Brown

- 127. *TLA in Pictures*
  Leslie Lamport

- 128a. *A Library for Visualizing Combinatorial Structures*
  Marc A. Najork and Marc H. Brown

- 128b.(Video)
  A Library for Visualizing Combinatorial Structures
  Marc A. Najork and Marc H. Brown

- 129. *Obliq-3D Tutorial and Reference Manual*
  Marc A. Najork

- 130a. *Visual Obliq: A System for Building Distributed, Multi-User Applications by Direct Manipulation*
  Krishna Bharat and Marc H. Brown

- 130b.(Video)
  Building a Distributed Application Using Visual Obliq
  Krishna Bharat and Marc H. Brown

- 131a. *The Juno-2 Constraint-Based Drawing Editor*
  Allan Heydon and Greg Nelson

- 131b.(Video)
  The Juno-2 Constraint-Based Drawing Editor
  Allan Heydon and Greg Nelson

- 132. *Processes are in the Eye of the Beholder*
  Leslie Lamport

- 133a. *The Third Annual Video Review of Computational Geometry*
  Marc H. Brown and John Hershberger

- 133b. (Video)
  The Third Annual Video Review of Computational Geometry
  Marc H. Brown and John Hershberger

- 134a. *From Quadrangular Sets to the Budget Matroids*
  Lyle Ramshaw and Jim Saxe

- 134b.(Video)
  Introducing the Budget Configurations
  Lyle Ramshaw

- 135a. *DeckScape: An Experimental Web Browser*
  Marc H. Brown and Robert A. Shillner

- 135b.(Video)
  The Deckscape Web Browser
  Marc H. Brown and Robert A. Shillner

- 136. *A Functional Specification of the Alpha AXP Shared Memory Model*
  Manfred Broy

- 137. *Proving Possibility Properties*
  Leslie Lamport

- 138. *Migratory Applications*
  Krishna A. Bharat and Luca Cardelli

- 139a. *WebCard: Integrated and Uniform Access to Mail, News, and the Web*
  Marc H. Brown

- 139b.(Video)
  The WebCard Web Browser
  Marc H. Brown

- 140. *Zippers: A Focus+Context Display of Web Pages*
  Marc H. Brown and William E. Weihl

- 141a. *Distributed Active Objects*
  Marc H. Brown and Marc N.Najork

- 141b.(Video)
  Distributed Active Objects
  Marc H. Brown and Marc A. Najork

- 142. *Collaborative Active Textbooks: A Web-Based Algorithm Animation System for an Electronic Classroom*
  Marc H. Brown and Marc A. Najork

- 143. *To Provide or To Bound: Sampling in Fully Dynamic Graph Algorithms*
  Monika R. Henzinger and Mikkel Thorup

- 144. *Program Fragments, Linking, and Modularization*
  Luca Cardelli

- 145. *Modularity in the Presence of Subclassing*
  Raymie Stata

- 146. *Studies of Windows NT Performance using Dynamic Execution Traces*
  Sharon E. Perl and Richard L. Sites

- 147. *Should Your Specification Language Be Typed?*
  Leslie Lamport and Lawrence C. Paulson

- 148. *Service Combinators for Web Computing*
  Luca Cardelli and Rowan Davies

- 149. *A Calculus for Cryptographic Protocols: The Spi Calculus*
  Martín Abadi and Andrew D. Gordon

- 150. *Smooth Scheduling in a Cell-Based Switching Network*
  Thomas L. Rodeheffer and James B. Saxe