Repository-Based Software Engineering Program

Progress Report

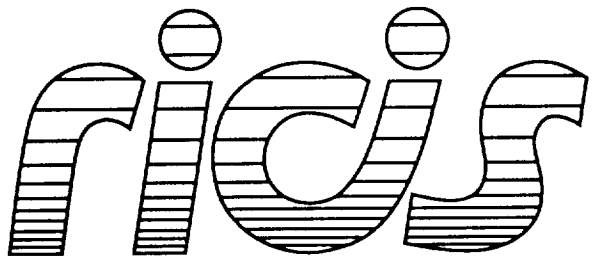
P-51

Applied Expertise, Inc.

9/92

Cooperative Agreement NCC 9-16
Research Activity No. RB.04

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division



Research Institute for Computing and Information Systems
University of Houston-Clear Lake

INTERIM REPORT

RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Applied Expertise, Inc. Mr. James Wilson served as principal investigator for Applied Expertise. Dr. E. T. Dickerson served as RICIS research coordinator.

Funding was provided by the Information Technology Division, Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA research coordinator for this activity was Ernest M. Fridge III, Deputy Chief of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

Applied Expertise, Inc.
PROGRESS REPORT
September, 1992

TASK 01

During September 1992, Applied Expertise performed the following work in support of Task 02 of the Repository-Based Software Engineering Program under University of Houston, Clear Lake subcontract 101.

ACTIVITIES AND ACCOMPLISHMENTS

- Prepared briefing charts for the following sections of September '92 Program Review
  - J. Garman - NASA Direction
  - E. T. Dickerson - Level 3 Report
    - -- Interviewed MountainNet and prepared several charts on their progress
  - J. R. Wilson - Program Management Plan Status
  - J. R. Wilson - Liaison Report
- Obtained consensus on COSMIC/AdaNET MOU
- Traveled to JSC/UHCL
  - Met with E.T. Dickerson, J. Garman, C. McKay, D. Eichmann, E. Fridge
  - Identified and documented two pilot projects (Jointly with D. Eichmann and E. Fridge)
  - Identified/obtained consensus on key program direction and performance criteria
- Documented two potential pilot programs
  - Interviewed Dr. Pitman, (MOC pilot [Pilot 1]) wrote up notes, reviewed notes with Pitman
  - Interviewed Dr. Eichmann, (STB pilot [Pilot 0]), reviewed notes and Pitman materials with Eichmann
  - Prepared process chart to describe and recommend activities for future pilot programs (including Pilot 0, 1)
- Attended RIG meeting on September 1-2, 1992 (Washington D.C.)
  - Chaired Metrics Technical Subcommittee
  - Chaired General Technical Committee Meeting (Chair was out of town.)
- Attended Reuse Acquisition Action Team (RAAT) meeting on September 9-10 (Washington D.C.) As Co-Recorder of the group (Sharon Rotter of Naval Command, Control and Ocean Surveillance Center shares the recorder position)

  - Documented the group's selection of recommendations to address, and groups to take responsibility of those actions and a diagram of implementation strategy (Attachment A)
  - Produced attendance list (Attachment B)
  - Attended Institute for Defense Analyses brief on legal issues in reuse (Attachment C)
  - Obtained Defense Software Repository System Non-Disclosure Agreement (Attachment D)
  - Obtained "Air Force Software Reuse Implementation Plan" (Copies already sent to Eichmann, McKay)
  - Discussed Fee-For-Service issues
- Prepared briefing to alert RBSE team of the upcoming RAAT/MIWG meeting and their task to provide recommendations to DoD on Fee-For-Service
- Prepared issue brief on ASV3 software/configuration management problems
- Reviewed STARS Prime Affiliates Press Release for Bill Hodges, Boeing STARS Program Manager
- Prepared initial outline for RBSE White Paper
- Participated in CCBs by phone.
- Revised "Top Five Issues List" (Attachment E)

DELIVERIES

- RBSE Program Review presentation Charts:
  - J. Garman - NASA Direction
  - J. R. Wilson - Program Management Plan Status
  - J. R.. Wilson - Liaison Report
- E.T. Dickerson RBSE Program Review Briefing Book
- Monthly Report: August, 1992d
- Top Five Issues List

PLANS FOR OCTOBER

- Prepare White Paper on RBSE's new role within NASA
- Attend and report on ASQC Software Conference
- Attend RAAT/MIWG Fee-For-Service meeting

TASK 02

During the month of September, AE performed several tasks in support of Task Number 02, representing NASA's technology transfer efforts.

ACTIVITIES AND ACCOMPLISHMENTS

- Prepared briefing charts for Frank Peñaranda's TCIS presentation to the Technology Utilization Officer's Conference at ARC.
- Continued analysis of TCIS processing capabilities. Captured analysis in a TCIS Process Model.
- Validated the TCIS Information Model with NASA Langley TUO.
- Designed a Macintosh-based Requirements Traceability Matrix (RTM) for TCIS requirements.
- Parsed the User Survey Document to extract User requirements. Populated the RTM with requirements from the User Survey.
- Continued revising the Concept Document. Major areas revised: software engineering, system capabilities and architecture sections.
- Received verbal approval from NASA to proceed with a separate TCIS development activity to support NASA's technology transfer. Generated program schedules for this activity.

Planned Activities for October

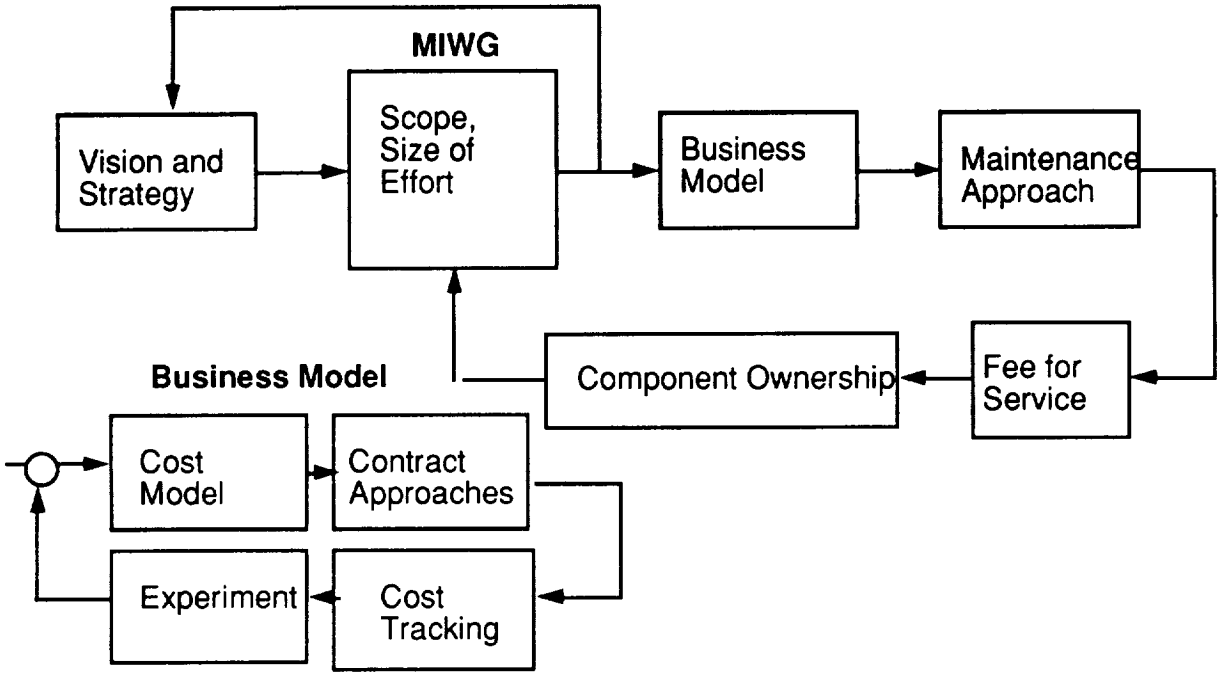
- Complete TCIS Concept Document. Distribute document for review. Baseline the Concept Document.
- Complete TCIS Process Model. Review model with TUOs.
- Travel to TUO sites to discuss models and promote new system, get User feedback on the development effort.
- Generate budgets for TCIS.
- Hire software engineer to support TCIS.
- Begin development of TCIS Software Requirements Specification.

MIWG/RAAT Joint Management Issues

(Voting MIWG members included: Jay Crawford, Elana Wright, Linda Brown, Stan Levine, C. Ronald Green and Dave Permar)

PROCESS

Diagram of Implementation Strategy



RECOMMENDATIONS - INITIAL POLL

√1. Develop Contract Clauses to Support Reuse and Evaluation Criteria for RFPs
6 Yes; need information/status

2. Use Value Engineering Change Proposals as a Way of Rewarding Reuse of Assets
1 Yes; 3 No; 2 Sideways

3. Establish Mechanism for Removing Barriers for Company B Using Company A's Software
4 No; 2 Sideways

√4. Provide Financial Incentives to Suppliers/Distributors/Maintainers Based on Performance Criteria
5 Yes; 1 Sideways

5. Remove Liability from Software Developer once Component is Certified for Library
2 Yes; 1 No; 3 Sideways

6. Develop Library Standards, etc.
Eliminated as being out of scope of Management Issue.

7. Establish Domain Specific Libraries. DoD Should Fund Cross-Service Domain Studies.
4 Yes; 2 Sideways

8. Provide to the Community a Listing of Existing Libraries and Their Contents, Including POCs.
Eliminated as being out of scope of Management Issue.

9. Work with Projects Offices (Government and Contractor) to Develop Quality Assets for Reuse.
3 Yes; 1 No; 2 Sideways

10. Change DoD Evaluation of Project Performance to Include Reuse. Use TQM Concepts to Insure Proper Perspective.
5 Yes; 1 Sideways

11. Consider IMEP, SBIR, and IRAD as Possible Funding Sources for Reuse Initiatives within Government and Industry.
5 Yes; 1 Down

12. Convene High Level WG Consisting of Government and Senior Industry; Meet to Define Appropriate Financial Incentives for Production and Use of Quality Reuse Assets.
6 Yes

13. DoD Fund Cross-Service Domain Studies.
Same as item 7.

14. DoD, Service, and PEO Awards in Recognition for Promoting Reuse.
5 Yes, 1 Sideways

15. Encourage Use of Value Engineering Change Proposals (VECP) as a Way of Rewarding Reuse of Assets.
Same as item 2.

16. Establish Business Case for Reuse Total Cost of SW Development and Benefits of Reuse.
6 Yes

√17. Establish Standard Measures for Reuse and Collect Data on Major Programs.
6 Yes

18. Give Credit to Organizations/Persons for Reuse Activities/Participation.
3 Yes; 3 No; related to item 14.

√19. Include Reuse Plans/Strategies in Acquisition Approval Process.
6 Yes

20. Provide Financial and Tech Support to PEOs and PMs for Reuse Initiative. Comes from Service Acquisition Executives or Above.
5 Yes; 1 Sideways

21. Provide Financial Incentives to Suppliers, Distributors, and Maintainers Based on Performance Criteria.
Same as item 4.

22. Update SEI Process Maturity Model (and Questionnaire) to Include Reuse.
3 No, 3 Sideways

23. Work with Government and Contractor Project Offices to Develop Quality Software Assets for Reuse.
Same as item 9.

24. Develop Library Standards, Interface Specifications, Certification and Acceptance Criteria, Library Network Interconnections, Library Interoperability, and CM.
Eliminated as not being in scope of Management Issue.

25. Establish Performance Criteria for Suppliers, Distributors, Maintainers Based on Use and Customer Satisfaction.
Eliminated as not being in scope of Management Issue.

26. Establish Standard Measures for Reuse (Amount that is Accomplished) and Collect Data on all Major DoD Programs.
Same as item 17.

27. Form Small Teams for Produce Initial Architecture and Interface Standards/Specifications for Domains. Establish a Method for Updating Standards More Quickly.
Eliminated as being a Technical Issue.

28. Government Should Sponsor an Effort to Standardize Software Requirements and Design Methodologies and Notations, Perhaps Through IEEE and ISO Standards/Specifications.
Eliminated as being a Technical Issue.

29. Integrate Software Reuse into Total Software Engineering Life Cycle Within the DoD. Incorporate it into DoD-STD-2167B and the Training for the Revision.
6 Yes

30. Set Up Some User Group to CM/Maintain Standards/Specifications for Reuse.
1 Yes; 1 No; 3 Sideways (One voting member departed)

31. Update SEI Process Maturity Model (and Questionnaire) to Include Reuse.
Same as previous item

32. CMU/SEI Produce Program Course Material in Reuse and Distribute It.
1 Yes, 3 No; 1 Sideways

33. Coordinate Sharing of All Reuse Activity Through Quarterly Information Sharing Sessions.
2 Yes; 1 Down; 2 Sideways

34. Develop DoD-Wide Reuse Newsletter w/Success, Lessons Learned and Cost Savings and Licensing Fee(s).
4 Yes, 1 Sideways

35. Integrate SW Reuse into Total Software Engineering Life Cycle; Put in 2167B and Include Training.
Same as item 29

36. Provide to the Community a Listing of Existing Libraries and Contents including POC.
Eliminated as being a Technical Issue.

37. Support Annual SW Reuse Symposium/Conference.
2 Yes; 1 No; 2 Sideways

38. Train Government Personnel in Reuse Techniques and Use of Available Resource -- Have DSMC Produce Stand Alone, PM Add-On and Correspondence Reuse Course.
5 Yes

New Recommendations

39. MIWG Undertaking Conceptual Approach to Scope and Size
ACTION ITEM: Dave Permar to expand definition.

- Recognize characteristics of DoD systems
- Categorize systems (consider underlying reasons for differences)
  safety and security
  packaging
  representation of requirements
- Develop solutions tailored to requirements of each category
- Requires active participation of Government and industry
- Clarification from Strassmann?

40. Develop Maintenance/Logistics Approach for Systems with Reusable Components
ACTION ITEM: Harry Joiner

- New version releases
- Trouble reports
- Incorporating Changes
- Who does/ who pays?
- Different environment than present
- Technical solutions to interfaces
- Some domain-specific issues

Develop an approach to the maintenance and logistical support for systems that contain significant reused components. For components that are not being maintained and supported by the Government, this will involve the issues of new version releases of the reusable components, handling of trouble reports and enhancements, integration of new capabilities, Government requested changes, etc. For components over which the Government has maintenance and support control, there remain the issues how to incorporate changes requested by one user in the systems of other users, handling problem fixes and updates to new target environments, who pays for changes, storage, RDIT, etc.

√41. Develop Fee for Service Strategy for Software

- Competitive market
- Unit cost (as determined by DoD Comptroller)
- Fee for Service (FFS) practices and reuse implementation strategy consistent
- Competition between Government and industry
- Must include liability, use agreements

√42. Establish Ownership Criteria
ACTION ITEM: Ron Green

Developer's Concerns

- Guidelines for decision making on ownership
- may be different models for types of components (design, requirements, code, etc.
- Various combinations of scenarios are possible
- Consider Government's interest
- Timing is important to contractor needs to know at RFP phase

√43. Business Model (Need Definition)
ACTION ITEM: Ron Green

- includes all factors
- validated cost models
- contract approaches include type of business base
- maturity of domain
- incremental implementation
- benefits and regrets from reuse; return on investment
- reuse as a part of software development process
- domain needs vs. product needs

44. Prescribe Policy for Addressing Software Reuse Liability

√45. Case studies on reuse

- use either present or past data to get started

5 Yes

46. Change current emphasis on building code libraries to domain development business case, proces, etc.

46a. Review current DoD priorities and produce finding (condition, cause, effect, criteria)

46b. Question merit of populating code libraries at this time.
ACTION: Jay Crawford to pursue with reuse Technical Working Group and Reuse Executive Steering Committee

Dave Permar Notes

Significant reform of Government (DoD) business is necessary to achieve more efficient and effective[STRASS] delivery of products/services (as measured against the cost of delivering the same or similar products/services by the private sector).

Such report as is necessary will only be achieved by unleashing competitive market forces.

The goal of DoD reform is greater efficiency and effectiveness in delivery of products and services consistent with the National Defense.

One avenue to achieving this reform is adoption of a Fee for Service (FFS) business program.

FFS is a means of doing business which holds to the following tenants:

1. Competitive market forces are the proven means to the allocation of increasingly scarce resources to the programs, products and services which are the most efficiently and effectively (acquired) produced in any given domain.

2. The customer, the one with responsibility and accountability for program accomplishment, must have the money and the discretion to place it with provider(s) of his/her choice.

3. The customer must know and hear <u>all</u> the cost of the products/services he/she purchases.

4. The provider(s) must be able to compete for work on a fair -- but fully cost justified loses (consistent with the Government cost accounting standards), as determined by the customer.

5. The price charged for products/services is fair if a customer with the money accepts the products/services, pays for them, and the provider delivered on time and within budget conforming goods and services.

6. Alt: The price charged for products/services is fair if it equals the unit cost (unit cost as determined by DoD Comptroller or his delegate) for those particular items and they are delivered on time and within budget.

[STRASS] Efficient, Effective defined by Strassmann, Paul, in *Information Payoff and The Business Value of Computers.*

In order to address perceived barriers to effective software reuse, any implementation strategy must contain policies and procedures for

1. Assigning liability for software reuse components,
2. identifying and preserving intellectual property rights in software reuse components,
3. identifying and assigning rights to monetary rewards and obligations derived from software component reuse, and
4. identifying and compensating for the costs of administrating for the costs of administration, maintenance and support of software reuse components (collections), support, libraries and library systems.

41. Develop Software Reuse implementation strategy consistent with FFS practices and principles (as described above).

Implementing an effective program of software reuse in DoD is a means to achieving more efficient and effective delivery of software.

To do so will require the active participation of Government and private sector persons whose business involves software acquisition.

In order that the scope and size of software reuse efforts be manageable, it is necessary to recognize certain characteristics of DoD systems incorporating substantial software components.

It is useful to categorize all such systems in one of two ways:

1. Real time/Weapons Systems or
2. Logistics/MIS

Alternative (one of three ways):

1. Real time/ Weapons Systems
2. Logistics/MIS or
3. Command and Control

Doing so allows those working for implementation of software reuse to tailor recommendations and actions to the category of system involved.

Each category has different requirements which necessitate substantially different solutions. For example Logistics and MIS systems typically involve general purposes computers and commercially available systems and application software (i.e. DBMS). The others do not.

One of the benefits of doing so is to make manageable efforts to address the particular program needs of each category.

Other benefits include:

- Focus
- Division of labor
- Leverage

FIRST VOTE			SECOND VOTE		ACTION
Item	# Yes	Selected √ = Yes	#Yes	Selected √ = Yes	
1.	9	√	4		
4.	10	√	2		
10.	7				
11.	2				
16.	8				
17.	14	√	9	√	Harry Joiner
19.	10	√	3		
20.	7				
29.	8				
38.	6				Elena Wright
39.	7				
40.	6				
41.	9	√	8		Dave Permar
42.	9	√	7		Ron Green
43.	13	√	11	√	Bill Farrell*
44.	8				
45.	11	√	9	√	Teri Payton**

* and Dave Permar
** and John Foreman

ATTACHMENT B

Reuse Acquisition Action Team
ACM SIG Ada Reuse Working Group

Joint MIWG/RAAT Meeting
9-10 September 1992
List of Attendees

Name	Organization/Address	Telephone/Fax (F)
Dennis Ahern	Westinghouse Electronics Systems Group SD&ED Software Engineering, P.O. Box 746-MS 432 Baltimore, MD 21203-0746 ahern@eclus.bwi.wec.com	(410) 993-6234 F (410) 765-4400
James Baldo, Jr.	Institute for Defense Analyses IDA/POET baldo@ida.org	(703) 845-6624 F (703) 553-0806
Linda Brown	OASD (C3I)/DDI lbrown@ddi.c3i.osd.mil	(703) 746-7928
Sherry S. Chaples	ASSET/SAIC 1710 Goodridge Drive McLean, VA 22102 chaples@mcl.saic.com	(703) 448-6411 F (703) 821-1433
Jay Crawford	NAWC-WD Code 31C China Lake, CA 93555 crawford%gssf.decnet@nwc.navy.mil	(619) 939-9738 F (619) 939-5841
Dave Dikel	Applied Expertise, Inc. 1925 N. Lynn St. Suite 802 Arlington, VA 22204 ddikel@ajpo.sei.cmu.edu	(703) 516-0911 F (703) 516-0918
Bill Farrell	DSD Laboratories 75 Union Avenue Sudbury, MA 01776	(508) 443-9700 F (508) 443-9738
John Foreman	STARS 802 N. Randolph Street, Suite 400 Arlington, VA 22203	(703) 243-8655 F (703) 528-2627
Diane Foucher	NAWC-Weapons Div Code 253 China Lake, CA 93555 foucher%25a.decnet@nwc.navy.mil	(619) 939-8160
C. Ronald Green	US Army Space and Strategic Defense Command CSSD-CR-S (Dr. C. Ronald Green) P.O. Box 1500	(205) 955-3498 F (205) 955-1310

	Huntsville, AL 35807 crgreen@redstone-emh2.army.mil	
Harley Ham	NAWC-Aircraft Div Code 825 Indianapolis, IN 46219-2189	(317) 351-4457 F (317)353-3583
Steven Harvey	Lockheed Aeronatical Systems Company Marietta, Georgia	(404) 494-1075
Phil Hood	PRC 1500 PRC Drive McLean, VA 22102	(703) 556-2370
David Hughes	Dynamics Research Corporation 60 Frontage Road Andover, MA 01810	(508) 475-9090 ext. 1795
Harry Joiner	Telos 55 N. Gilbert St. Shewsbury, NJ 07702 joiner@tsg.com	(908) 842-8647 F (908) 530-5904
Stanley H. Levine	PM CHS SFAE-CC-CHS Ft. Monmouth, NJ 07703 sfae-cc-chs@monmouth-emh3.army.mil	(908) 544-2603
Chuck Lillie	SAIC/ASSET 1710 Goodridge Drive (304)594-9836 McLean, VA 22102	(703) 749-8732
Henry L. Marshall	Army Reuse Center Stop H-4 Fort Belvoir, VA 22060 marshallh@melpar-emh1.army.mil	(703) 285-9714 F (703) 285-6377
Kathy Miles	NIST Bldg. 225, Room A266 Gaithersburg, MD 20899 miles@ecf.ncsl.nist.gov	(301) 975-3156 (301) 590-0932
Colleen Murphy	Softech, Inc. 1600 N. Beauregard Alexandria, VA 22311	(703) 824-4536
Michael Nash	Institute for Defense Analyses	(703) 845-6697 F (703) 845-6848
Teri Payton	STARS 802 N. Randolph Street, Suite 400 Arlington, VA 22203	(703) 351-5310 F (703) 528-2627

	payton@stars.reston.unisys.com	
Dave Permar	DLA Systems Automation Center Columbus, OH	(614) 692-9399
Richard Peterson	SofTech 1600 N. Beauregard Street Alexandria, VA 22311	(703) 824-4521
Sharon D. Rotter	Naval Command, Control and Ocean Surveillance Center (NCCOSC) RDT&E Division (NRaD), Code 411 San Diego, CA 92152-5000 rotter@nosc.mil	(619) 553-4013 F (619) 553-4808
LCDR Anne Sullivan, USN	BUPERS-10T3 Washington, D.C 20370-5100	(703) 614-3578 (703) 614-1561 F (703) 693-5942
Roger B. Williams	Software Productivity Consortium 2214 Rock Hill Rd. Herndon, VA 22070 williams@software.org	(703) 742-7132
Elena Wright	DISA/CIM//XER[Reuse] 701 S. Courthouse Rd. Arlington, VA 22204-2199	(703) 536-6900

ATTACHMENT C

SOFTWARE PATENTS: THE CASE FOR COMPULSORY LICENSING

DI Sc Ke E1

Craig A. Will

Computer and Software Engineering Division
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

INTRODUCTION

In the last several years, there has been increasing concern among software developers about the potential impact of software patents on the software industry. Many observers have expressed the opinion that the increasing tendency to patent software innovations and to aggressively seek royalties under threat of infringement lawsuits will have a strongly negative effect on software innovation and cause instability in the U.S. software industry. Because software is an increasing part of today's technological systems and business, if these claims are true, software patents could have an adverse impact on the economic competitiveness of the United States.

The concern in the software industry about patents has been expressed in a number of press reports. The *New York Times*, for example, reported that "fear is mounting among software companies and programmers that [software] patents are about to start a flood of lawsuits" resulting in possible "chaos in the software industry."[1] The article noted that "although there has been little litigation of software patents yet, lawyers expect an onslaught of suits that could dwarf the copyright disputes now being heard between software companies." "Critics say," reported the *Times* article, that "the patent office has frequently issued patents on software that was really standard procedure in use for years by programmers. Software developers then discover after the fact that some portion of their program ... infringes a patent they never knew existed." According to the *Times* report, one major software developer, Wordperfect, gets a letter per month from patent holders claiming infringement.

According to Brian Kahin, an attorney and research fellow at Harvard University's Kennedy School of Government, "in the long run, the costs of doing business in a patent environment will radically restructure the industry. Many small companies will fold under the costs of licensing, avoiding patent infringement, and pursuing patents defensively. The individual software entrepreneur and inventor may all but disappear. There will be fewer publishers and fewer products, and the price of software will rise to reflect the costs."[2]

The special concern about software patents—as opposed to patents for any other kind of technology—results from several factors, including the nature of software and the software industry and the history of software patents.

The nature of software, and the software industry, seems to result in special problems. Software products can be extremely complex, with thousands of mechanisms that could conceivably infringe a patent. Moreover, software typically makes use of very general processes that can apply to a very broad range of applications. The industry itself is composed of many small companies, with many competitors working on the same problems, who are likely to come up with very similar solutions nearly simultaneously.

Whether software could be patented was the subject of much controversy and litigation in the late 1960s and throughout the 70s, with software ruled patentable in some circumstances and not in others (as will be discussed in more detail later in this paper). The uncertainty of obtaining a patent and the widespread perception that software was unpatentable resulted in discouraging all but a relative trickle of patent applications. This began to change in 1981, when the Supreme Court made a

1. Fisher, Software Industry in Uproar Over Recent Rush of Patents. *New York Times*, May 12, 1989, at A1.

2. Kahin, *The Software Patent Crisis*. TECHNOLOGY REVIEW, April, 1990, p. 53.

significant ruling in *Diamond v. Diehr*[3] that favored software patents, and the Patent and Trademark Office dropped its opposition to the practice and adopted an apparently increasingly liberal policy toward the patenting of software.

Although there continued to be only a relatively small number of patents issued for software thoughout most of the 1980s, by 1988 many of the patents that had been in the pipeline for some time were issued, including patents that seemed to cover practices that were common throughout the industry. For example, one patent covering the ability to display several documents at once in "windows" on a computer screen[4] is reported to apply to large numbers of application programs that make use of such multiple windows.[5]

The rapid increase in the number of software patent applications, after a long period when software patents were rare, itself caused additional problems, because of the lack of effective ways of searching to see if an invention is really new. This and the relative inexperience of patent examiners with software has resulted in delays in issuing software patents and questions about the validity of issued patents. Delays are particularly troublesome in that pending applications, which are not public, are threats to developers who invest resources into developing software that may turn out to infringe a patent without the developer knowing. Invalid patents can result in what some in the industry have termed "extortion," as software developers pay license fees to avoid the cost and uncertainty of lawsuits, which can cost each side from hundreds of thousands to millions of dollars.

This paper discusses software patents, their economic implications for the computer industry, and possible ways of improving the system so as to improve the economic competiveness of the United States.

The paper first describes the basic patent system in the United States, and then presents a history of the patentability of software from the 1960s to the 1980s.

The paper then looks at evidence suggesting what the economic impact has been of the patent system, and discusses variations among patent systems worldwide and the effect of a particularly important parameter in patent systems, that of compulsory licensing.

Benefits and costs of the patent system, with particular respect to software and the computer industry, are then discussed.

The results of an initial analysis of a very small sample of issued patents are then presented, together with a calculation that attempts to estimate the number of patents for software that could be issued if patents were taken advantage of in the software industry in the same way as in other industries.

Some assertions about the nature of the software patent problem and possible solutions are then presented, including specific proposed revisions to the present patent system.

A final section presents general conclusions, and an appendix presents text of proposed revisions to the patent statute.

CHARACTERISTICS OF PATENTS

Patents are a powerful form of protection for technology. Obtaining a patent is, however, far more difficult, costly, and uncertain than is making use of other mechanisms such as copyrights and trade secrets. The standard of innovativeness required to obtain a patent is high—the invention must be novel (not previously discovered), and it's creation must not have been obvious to someone of ordinary skill.[6] Obtaining a patent generally takes 18 to 24 months (sometimes longer for software patents), and can cost from about $5000 to $15,000 or more for attorney's fees, plus from about $3500 to nearly $7000 or more for fees paid to the Patent and Trademark Office.[7]

Applications for a patent, which must include a complete description of the invention along with "claims" that define the boundaries of the protected invention,[8] are examined to determine whether they are novel, unobvious, and meet other requirements. Because these claims describe an invention at a conceptual level, rather than a mere

3. 450 U.S. 173 (1981).

4. U.S. Patent 4,823,108, issued April 18, 1989, and assigned to Quarterdeck Office Systems, Inc.

5. According to one analyst quoted in the *New York Times* article cited above the Quarterdeck patent has the potential to "shatter the industry."

6. 35 U.S.C. § 103 (1988).

7. PTO fees include an application fee, issue fee, and "maintenance fees," the last due at intervals throughout the life of the patent. Patent fees were only a few hundred dollars before 1980, when application and issue fees were increased and maintenance fees introduced, as part of a desire by Congress for the PTO to be self-supporting. They were increased again in 1990 and are now set at a total of $3340 for an individual, small business, or nonprofit institution or $6680 for a large corporation, plus slightly more for especially complex patent applications.

8. 35 U.S.C. § 112 (1988).

implementation, patent protection for software is much broader than is copyright. Patents protect an invention in a manner that is relatively independent of its particular implementation, whereas copyright law is designed to protect an expression of an idea rather than the underlying idea itself. While extremely broad interpretations of copyright law might see copyright as protecting software at a level as abstract as that of a patent, most observers see little or no overlap between what aspects of software are protected by a patent and what are protected by copyright.[9]

A successful patent holder gains a powerful right—the right to exclude others from using the invention for 17 years after the patent is issued. A patent, once issued, is presumed to be valid.[10] The first inventor obtains all rights to an invention—independent discovers of the invention have no rights and may be sued for infringement. Although the actual first inventor—not the first to apply for a patent—generally gains these rights, an inventor who is issued a patent is presumed to be the true inventor, and attempts to change this presumption usually require long and costly litigation.

Patent holders can sue those who infringe their patents to recover damages that are at a minimum a "reasonable royalty," plus costs,[11] and, in addition, can bring an injunction against use of the invention by another, although such injunctions cannot be brought against the U.S. government or, given proper authorization, its contractors.[12]

The right to exclude others from use or sale of an invention is a particularly powerful one. In general, a patent holder can choose to license or not to license an invention, and can ask any license fee he or she chooses.[13] In cases of deliberate or "willful infringement," the patentee can recover up to treble damages.[14]

9. Samuelson, *Survey on the Patent/Copyright Interface for Computer Programs*. 17 AIPLA Q.J. 256 (1989).

10. 35 U.S.C. § 282 (1988).

11. 35 U.S.C. §§ 284, 285 (1988).

13. Chisum, cite section. There are some exceptions in that a patent cannot be used to further what would otherwise be an unfair trade practice. For example, a patent holder can choose to license or not a patented machine that uses certain supplies as part of its operation, but cannot require that these supplies be purchased from the patent holder as a condition of obtaining a license.

13. 28 U.S.C. § 1498 (1988). Remedies for infringement by the government or authorized contractors are limited to claims for royalty payments.

14. 35 U.S.C. § 284 (1988). Leinoff v. Luisin Milona & Sons, Inc., 726 F.2d 734 (CAFC, 1984).

THE HISTORY OF SOFTWARE PATENTABILITY

Whether and what aspects of software might be patentable has been the subject of controversy and litigation for nearly 25 years. That there would be any question about whether an entire area of technology is patentable is rather unusual, with the only other comparable example being the development of artificial life forms using biotechnology, which have been ruled patentable.[15]

There are a number of reasons why software patentability became an issue. The very nature of software, which seems to have similarities to both writing and to machinery, without being clearly either, caused difficulties early on, since it was not clear whether software might be better suited for protection by copyrights or by patents. In addition, there has never been agreement within the computer industry on whether patent protection was desirable for software, and thus neither the courts nor Congress could act on the basis of industry consensus.

The specific grounds upon which the legal controversy has been fought primarily involved three specific characteristics of patent law. The patent statute has long required that an invention be either a "process, machine, manufacture, or composition of matter,"[16] and case law in the courts interpreted the meaning of "process" and "machine" in ways that seemed to exclude software long before software patentability ever became an issue. One problem was that a "process" initially referred to a series of steps for manufacturing a chemical, and an early court decision appeared to define it very restrictively, as "a mode of treatment of certain materials" that are "transformed and reduced to a different state or thing."[17] A second problem was a legal doctrine that became established in the 1940s and 50s, known as the "mental steps doctrine," that appeared to prevent patentability for processes that were composed of steps that could be carried out in the human mind, on the grounds, as one court put it, that "thought is not patentable."[18] Third, the notion that scientific principles were not patentable was extended to inventions that primarily consisted of "mathematical formulas."

15. Diamond v. Chakrabarty, 447 U.S. 303 (1980). Printed matter, business systems, and scientific principles are also considered unpatentable.

16. 35 U.S.C. § 101.

17. Cochrane v. Deener, 94 U.S. 780 (1876).

18. In re Abrams, 188 F.2d at 168 (CCPA, 1951).

Software Patents in the 1960s and Early 70s

The first guidelines adopted by the Patent Office in the 1960s for the patentability of software[19] provided that computer programs, whether claimed as a machine or as a process, were not patentable. These resulted in part from a Presidential Commission report[20] that recommended that computer programs be expressly excluded from coverage by the patent laws, based primarily on the inability of the Patent Office to properly examine applications for software patents, and also as a result of opposition from IBM and other major computer manufacturers.[21] The Patent Office strongly opposed software patents until the early 1980s, with software patentability the subject of a continuing legal battle involving the Patent Office and the Court of Customs and Patent Appeals (CCPA).

Unlike the Patent Office, the CCPA consistently took a favorable view of software patentability, and in a series of court decisions from 1968 through 1970, it briefly eliminated the legal arguments against patentability. In the first such case, *In re Prater*,[22] the court ruled that the "mental steps" doctrine did not apply in cases in which all steps were carried out completely mechanically, such as in a computer program, and thus such programs were patentable. The *Prater* decision also concluded that the notion "that all processes, to be patentable, must physically operate upon substances" was an incorrect interpretation of the 19th century decision taken out of context.[23] A 1970 ruling, *In re Musgrave*,[24] appeared to throw out the mental steps doctrine completely, even in cases where the invention consisted of purely mental processes, not their mechanical equivalent.

Software Patentability After the Benson Decision

A landmark 1972 decision by the Supreme Court, however, in the case of *Gottschalk v. Benson*,[25] reversed these trends by ruling that an invention involving a method for converting binary-coded-decimal (BCD) numbers to binary numbers was unpatentable. The court said that "phenomena of nature ... mental processes, and abstract intellectual concepts are not patentable," and the patent, if granted, "would wholly preempt the mathematical formula and [the] practical effect would be a patent on the algorithm itself." The *Benson* ruling caused much confusion because of the lack of clear rationale and reasoning for its decision, and was widely criticized by legal scholars.[26] Its principal effects were to reverse the trend toward patentability that had been led by the CCPA, vindicate the Patent Office's long-standing opposition to software patents, and to create the perception in the computer industry that software was not patentable, resulting in a strong "chilling effect" discouraging patent applications.

Even so, many patents for software were issued after the *Benson* decision, with the CCPA interpreting the Supreme Court decision in increasingly liberal terms as time passed. Inventions that were claimed as a machine, or "apparatus," but were actually constructed as software, were ruled patentable.[27] Software inventions claimed as processes that did not involve mathematical calculations were also held patentable, including a technique for dynamic assignment of priorities to tasks in an operating system[28] and a program for natural language translation (e.g., Russian to English).[29] Toward the end of the 1970s, the issue of patentability focused primarily on determining exactly what fit the criteria of the prohibited "algorithm" referred to in *Benson*. A considerable number of cases were litigated that together developed an increasingly complex set of tests for patentability, and which included a second Supreme Court ruling against patentability, *Parker v. Flook*.[30]

Software Patents in the 1980s

In a 1981 decision, *Diamond v. Diehr*,[31] the Supreme Court ruled again on software patents. Although the invention in question was little different than that in the *Flook* case that held the

19. Guidelines were first proposed in 1966. Official Gazette. August 16, 1966. Formal guidelines were adopted in 1968. 33 FEDERAL REGISTER 15609.

20. Report of the President's Commission on the Patent System, *To Promote the Progress of ... Useful Arts in an Age of Exploding Technology* (1966).

21. See Hauptman, *How Computer Software Has Come to be Protected Under the U.S. Patent Law*. 6 COMPUTER LAWYER 11.

22. 415 F.2d 1378 (CCPA) and 415 F.2d 1393 (CCPA).

23. 415 F 2d at 1387 (CCPA).

24. 431 F.2d 882 (CCPA).

25. 409 U.S. 63 (1972).

26. See, e.g., Chisum, *The Patentability of Algorithms*. 47 UNIVERSITY PITTSBURGH LAW REVIEW 959 (1986).

27. See, e.g., In re Johnson, 502 F.2d 765 (CCPA, 1974).

28. In re Chatfield, 545 F.2d 152 (CCPA, 1976).

29. In re Toma, 575 F.2d 872 (CCPA, 1978).

30. 437 U.S. 584 (1978). Other decisions that were significant in refining the criteria for patentability included In re Freeman, 573 F.2d 1237 (CCPA, 1978), In re Walter, 618 F.2d 758 (CCPA, 1980), and In re Abele, 684 F.2d 902 (CCPA, 1982).

31. 450 U.S. 173 (1981).

invention unpatentable, this time the court, in a 5-4 decision, ruled in favor of patentability. The invention itself was a process for improving the curing of rubber in a mold, using a sensor that continuously measured the temperature in the mold, and an equation that was used by a computer to recalculate the time until the curing was complete so that the mold could be opened. Though the equation itself was not new—and presumably unpatentable as a scientific principle—the process using the computer and equation was held patentable.

Although the decision itself was little different than rulings made about the same time by the appeals court, the ruling had a remarkable effect on software patentability. The Patent and Trademark Office switched from a policy of opposing software patentability to one that generally favored patentability. After a few additional rulings that interpreted *Diehr* further, the last in 1983, the federal appeals court concerned with patents made no further rulings on the subject until 1989, presumably because the PTO had liberalized its policy and it was no longer necessary for patent applicants to appeal its rulings.

Although the number of patents issued for software continued to be relatively small through 1987, the Supreme Court ruling slowly changed the perception of those in the software industry toward the view that software was patentable, and in recent years, increasing numbers of software patents have been issued as applications that were filed earlier made it through the backlog. By one count, there were 200 software patents issued in the first 4 months of 1989.[32]

In addition, the belief is growing that patent office practices are taking an increasingly liberal approach to software patents, with patents being granted involving mathematical equations in contexts that are seen by many as going considerably beyond the court's ruling.[33] A 1989 decision by the Court of Appeals for the Federal Circuit in which an algorithm using a mathematical formula in conjunction with a read-only-memory was ruled patentable, appeared to further broaden the patentability of computer software.[34]

32. Kahin. *The Software Patent Crisis.* TECHNOLOGY REVIEW. April, 1990. p. 53.

33. A particular example is the patent issued to Narenda Karmakar at Bell Laboratories for an optimization method that has been widely regarded as revolutionary. See Parfomak. *U.S. Patent 4,744,028—"Baiting" the Supreme Court?* __ IDEA 5. and Andrews. *Patents on Equations: Some See a Danger. New York Times.* February 15, 1989, p. D1.

34. In re Iwahashi. 888 F.2d 1370 (CAFC). The Court of Appeals for the Federal Circuit is the successor to the CCPA after a reorganization.

THE ECONOMIC IMPACT OF PATENTS

The issue of whether software is patentable has—rather oddly—been decided almost solely on rather narrow questions of law, rather than concerns of public policy. Whether patenting software would be, on the whole, good or bad for the software industry has never been directly considered by the courts. This is as it should be, since it is Congress, not the courts or the Patent and Trademark Office, who are responsible for creating by legislation a patent system that serves the needs of economic development and protects the rights of inventors.

While critics of software patents may see the software industry as unique, most of the specific problems that they see resulting from software patents also occur with other areas of technology. It is thus worth first looking at the broad issue of what is known about the extent to which the patent system actually does successfully encourage innovation.

Evidence that Patents Encourage Innovation

The patent system, though now firmly entrenched in nearly every industrialized country in the world, has been among economists one of the more controversial economic institutions. Its existence results more from history than any real evidence that its benefits necessarily outweight its costs, particularly in every area of technology.

For example, one social scientist, in presenting evidence to a Canadian Royal Commission studying the economic effects of the patent system, stated that "no economist, on the basis of present knowledge, could possibly state with certainty that the patent system [in the United States], as it now operates, confers a net benefit or a net loss upon society." "If we did not have a patent system, it would be irresponsible on the basis of our present knowledge, to recommend instituting one. But since we have had a patent system for a long time, it would be irresponsible, on the basis of our present knowledge, to recommend abolishing it."[35]

Debate over the patent system has thus been based primarily on what one commentator has called "hero-inventor tales, horror stories, and other tools of the advocate's art."[36]

Although there have been a number of economic studies of patents, they have been primarily based on

35. Quoted in Firestone. *Economic Implications of Patents.* Ottawa, Canada: University of Ottawa Press. 1971. at 231.

36. Scherer. *The Economic Efforts of Compulsory Patent Licensing.* Monograph 1977-2. New York University. 1977. at 5.

questionnaires and interviews of business executives asking for their opinions—and thus highly subjective—or studies of special historical situations, such as that of Switzerland or the Netherlands, both of which had no patent system for a period of time, although neighboring countries did. The problem with the latter studies is that the finding that the lack of a patent system did not retard innovation—as Erich Schiff reported[37]— was hardly decisive, since patents could be obtained by Dutch inventors in neighboring countries. Schiff also found that when the Netherlands reintroduced a patent system (in 1912), patents to Dutch citizens issued by other countries significantly increased. Although Schiff interprets this as the result of the patent system encouraging more innovation, Scherer suggests that this increase could have been due to other factors, such as the establishment of patent laws making Dutch citizens more aware of patents and resulting in an increase in the number of patent attorneys.[38] In the case of Switzerland, Schiff found strong evidence of innovation despite the lack of a patent system, and no evidence of increased patenting in other countries when the Swiss patent laws were strengthened (in 1907).

Questionnaire and interview data has generally found that the tendency of managers to view patents as important varied very substantially depending on the industry.[39] Patents were seen as critically important for investment in research in some industries, such as pharmaceuticals—this results from the large investments needed for research and development and clinical testing of drugs, the ease of duplicating such drugs once known (as generic drugs), and the effectiveness of composition-of-matter patents in protecting such drugs. In other industries, however, patents were viewed as far less important. Having a technological lead on one's competitors and having superior sales or service were in most industries viewed as more important to successfully introducing new technology as were patents.

How patents are viewed also tends to vary considerably depending upon whether a company is small or large, with small companies tending to rely more on were patents.

Somewhat more substantive and reliable data do exist for some specific aspects of the economics of patents, particularly the effects of compulsory licensing, as will be discussed later.

An author of one of the interview studies concluded that "[i]t is inconceivable, whether or not there is a patent system, that the leading firms in such industries would abandon the attempt to make inventions." He suggested, however, that "defensive" research— that "directed to making an advance before competitors do," as well as research "devoted to designing around other people's patents," would be reduced, avoiding much duplication of research. "On the other hand, the absence of a patent system would undoubtedly lead to greater secrecy than at present." He concluded that "[o]n balance, the absence of patent protection would be likely to prevent some of the present duplication in research but, because of its encouragement to secrecy, it might well hinder the rapid spread of technical knowledge. The quality of invention might also be adversely affected because of this."[40]

Variations Among Patent Systems Worldwide

Patent systems are very similar throughout the world, and, although there are many variations on details, even these differences are starting to disappear as attempts are made to standardize and allow broad patent rights in many countries through reciprocal treaties and centralized organizations such as the European Patent Office.

Patent systems typically provide that an application be examined before a patent is issued and that all rights to a particular invention go to specifically named inventors (or to corporations or others to whom they have assigned rights), and provide for a similar period of protection, usually 16 to 20 years.

There have been significant differences among the systems, however, particularly over the years. While all patent systems appear to protect mechanical and electrical inventions, some also protect chemical products ("compositions of matter") but not chemical processes; others protect chemical processes but not products. Some systems start the time clock for protection upon application,

37. Schiff. *Industrialization without Patents*. Princeton: Princeton University Press. 1971.

38. Scherer. op. cit.. at 36.

39. The principal studies have been done by the following: Scherer (Ed.), *Patents and the Corporation: A Report on Industrial Technology under Changing Public Policy*. Boston: J. J. Calvin. 1958. Taylor and Silberston. *The Economic Impact of the Patent System: A Study of the British Experience*. Cambridge: Cambridge University Press. 1973. Levin. Klevorick. Nelson. and Winter. *Appropriating the Returns from Industrial R&D*. Brookings Papers on Economic Activity. 1988.

40. Quoted in Firestone. op. cit.. at 233-234.

40. Scherer. op. cit.. at 63. citing an estimate by Hollabaugh and Wright.

others only when the patent is issued.

While most systems give patent holders an exclusive monopoly on the invention, some systems provide that compulsory licensing of patents be required under certain conditions. Thus, for example, Canada—in response to complaints about price-gouging in the pharmaceutical industry—passed a law in 1969 allowing the Canadian Commission of Patents to grant compulsory licenses to importers of drugs patented in Canada.[41]

Compulsory licensing provisions have existed in a large number of countries, including the United Kingdom, Canada, France, West Germany, the Netherlands, Japan, Switzerland, and others. However, the typical situation is that companies who desire to license a patent but are refused a license by the patent holder must request a compulsory license from the government. The number of such requests made is quite small—in the hundreds over decades worldwide—and there is a wide variation in the willingness of authorities in different countries to grant such licenses. Thus, from 1960-1974 Canada received 183 requests for compulsory licenses and granted 117, while West Germany, from 1950-1979, received 37 requests and granted none.[42] The small number of requests is variously interpreted as resulting from the increased willingness of patent holders to grant requests if they know that the government can require it, excessively strict conditions for compulsory licensing, or a small demand for licenses. The United States has required licensing of patents only as a result of antitrust decrees, but the scale has been huge, with an estimated of 40,000 to 50,000 patents involved as of 1958.[43]

In most countries, the first to file a patent application receives the patent, whether or not he or she is the first inventor. United States law is unusual in that it grants the patent to the first person to invent it, with a complex system that considers who first conceived, reduced to practice, and filed an application in making this decision.[44]

Another difference between patent systems is the extent to which they use renewal fees to raise revenue and to weed out unutilized or marginal patents. West Germany, Austria, France, Great Britain, Japan, the United States, and others require such fees to maintain patent rights. Germany has particularly sharply increasing fees as time passes. In Great Britain, about half of issued patents remained in force after nine years, and only 18% throughout their full lifetime.[45]

The Effect of Compulsory Licensing

The question of compulsory licensing of patents is a particularly significant one, because requiring licensing is one way of weakening a possibly too-powerful patent system and eliminating some of the associated economic efficiencies. It is also significant because most schemes for compulsory licensing involve some discretion on the part of a regulatory system that approves applications for licenses on a case-by-base basis—and thus can, in a way that the rules of a general patent system cannot, decide based on the circumstances of each particular case.

Relatively good evidence is available on the effects of compulsory licensing, including both questionnaire data and, significantly, analysis of research and development investment by companies forced to license patents by antitrust decrees.

One study suggests that the impact of compulsory licensing of patents is very different for different industries. The Cambridge Study, a survey of British companies in the chemical, pharmaceutical, mechanical engineering, and electrical engineering industries, asked corporate managers to estimate the amount of research and development in their industry that would take place the year of the survey if a new system were put in place worldwide, that required compulsory licensing of patents to any competitor, on reasonable terms. In the pharmaceutical industry, respondents estimated a 64% drop in the R&D investment dependent on patent protection, while only 5% in basic chemicals, 5% in mechanical engineering, and a negligible drop in electronics engineering.[46]

The responses of managers in the pharmaceutical industry may well be exaggerated, given the powerful barriers in addition to patents to the introduction of generic drugs. For example, a compulsory license to manufacture and sell a generic equivalent of the tranquilizer Librium was granted in the U.K in 1968, and the generic product introduced in the market in 1969. However, by 1971, the generic equivalent had achieved sales of only one three-hundredth of that of Librium itself, despite a retail price 20-25% lower.[47]

41. Firestone, op. cit., at 208.

42. From Kaufer, op. cit., at 52. Original data from S. Greif, from a 1981 paper in German.

44. Cite first to file paper.

45. Taylor and Silberston, op. cit., at 97.

46. Taylor and Silberston, op. cit., at 199.

47. Report of the British Monopolies Commission, cited in Scherer, op. cit., at 43.

The negligible impact of compulsory licensing in electronics was attributed by the interviewees as resulting from patents being vulnerable to invalidity as not novel, because product life cycles are short, and because it is relatively easy to design around patents. These factors are probably as true for the software industry (or more so) as for electronics.

Studies of compulsory licensing resulting from antitrust decrees include interviews with managers of companies forced to license their patents, and statistical analysis of research and development expenditures of such companies.

Interviews conducted by Scherer and his colleagues in 1958 resulted in a conclusion that there was "no significant" discouragement of research and development resulting from the antitrust decrees, but they did find "distinct evidence that companies subjected to antitrust mandatory licensing decrees were patenting fewer of their inventions and keeping relatively more of their new technology secret."[48] This tended to be particularly true of process patents, which are more amenable to protecting by secrecy.

A study of 678 U.S. corporations with significant research and expenditures in 1975, in which 42 companies were subject to an antitrust decree involving patents, was done to see if whether they were subjected to such a decree, the impact of the decree, and other factors affected the amount of research expenditures. The researchers concluded that the analysis "provides no significant indication" that the companies "subjected to compulsory patent licensing under antitrust decrees sustained less intense R&D efforts than other firms of comparable size and industry origin. If anything, the opposite tendency is revealed," with a significant indication that expenditures increased slightly.[49]

BENEFITS AND COSTS OF PATENTS

Whether a patent system ought to exist—and, if so, what particular system is best—depends upon an assessment of whether the benefits of a given system exceed its costs.

Historically, the development of the patent system has been based on the individual inventor. It was assumed that providing a monopoly on a invention was necessary to provide incentives to inventors, who may have invested considerable effort in conceiving and perfecting an invention. Without such a monopoly, others might quickly imitate the invention, preventing the inventor from recovering much of the economic value contributed by the invention.

This simple logic has increasingly been questioned as individual inventors have been largely replaced by corporate research and development and as technology has become more complex. The magnitude of the replacement of individual inventors is seen by the change in the proportion of patents issued to individuals by the U.S. Patent and Trademark Office: at the beginning of the 20th Century, 82% of U.S. patents were issued to individual inventors. By the early 1980s, however, this had dropped to a mere 18%.[50]

In the modern corporate environment, invention is only a small part of what is necessary to bring a product to market and to maintain a corporation's market share. Investment in manufacturing facilities, control of distribution channels, investments in marketing, brand-name recognition, technological leadership and know-how, and other factors may all dwarf the effects of patents. Certainly, the software industry has thrived for decades with no apparent need for patent protection (although copyright protection seemed necessary once the widespread availability of personal computers and floppy disks provided the motivation and means for software piracy).

Determining the optimum patent system, given this environment, can only be done by carefully assessing the relative benefits and costs of patents for a particular technology.

Benefits of Patents

There are four principal benefits of patents: encouragement of investment, increased flow of information, use as a mechanism to allow small companies to enter a market, and standardization imposed by patents.

1. Encouragement of Investment. This has already been covered in some detail above, and while for some technologies patents do seem to clearly encourage investment, it is far from clear whether patents play a significant role in technologies such as electronics, computer hardware, and software, although in many cases, such as, for example, the "386" microchip manufactured by Intel, or the thus far successful ability of Apple to prevent cloning of the MacIntosh hardware, presumably affected their investment decisions.

48. Scherer, op. cit. at 64.

49. Scherer, op. cit., at 75.

50. Kaufer, op. cit. at 16.

2. *Increased Flow of Information.* The requirement that patent applicants disclose how an invention works, which is publically printed when the patent is issued, presumably helps advance the state of technology. This is probably particularly valuable with software, since once a technique is known it can usually be duplicated easily by almost anyone in the industry with no special equipment or know-how required. However, the computer industry has been generally characterized by an unprecedented level of the free flow of information, and it is not clear the extent to which patent disclosures increase this flow.

3. *Mechanisms to Allow Small Companies to Enter a Market.* Patents provide unique mechanisms that can allow small companies to enter a market that may well be critical. Such a small company may be able to enter a market with a new product even without the kind of knowhow, capital, sales, service, and distribution organization and other advantages that a large corporation may have. Here it is probably necessary for small companies to have a monopoly (and not just royalty income); otherwise large companies might quickly introduce competing products and drive the small company out of business. To maintain these incentives for small companies, any compulsory license system should have a special provision for small companies, perhaps giving them monopoly rights for a period of time (or until they had a reasonable chance to establish a market) before requiring them to license their patent.

4. *Promotion of Standardization and Software Reuse.* Patents can also promote standardization, albeit sometimes in a heavy-handed way, by monopolizing the market and offering no realistic alternative. Patents may also provide a means for encouraging a reusable components industry for software, by both providing economic incentive to develop reusable components (which cost more than ordinary software), and by potentially providing a mechanism that may make certain techniques simply unavailable for use unless they are obtained in the form of reusable components.

Costs of Patents

There are many costs of patents that must be weighted against the above benefits. These costs include those due to monopoly power, the dominance of large corporations, invalid patents, unanticipated infringement, allocative research costs, duplicate research, and administrative costs.

1. *Monopoly Power.* A well-recognized cost of patents is that the monopoly power given to a patent holder can prevent the usual effects of competition in reducing prices and encouraging the best business practices, especially if the holder refuses to license the patent. Some companies may be able to monopolize an entire technology by holding a broad range of key patents. Not only can such holders charge monopoly prices, but may also be able to get away with offering inferior products. For example, a company who has very innovative technology, but poor manufacturing practices and a poor service organization might still retain dominance in an area of technology, producing technically advanced but marginally reliable products. At the same time, competitors may be forced to use inferior techniques and produce technologically inferior products. Monopoly power can result in significant excess profits which can be considered costs of the patent system. For example, Intel Corporation in 1986 introduced a microprocessor, the 386, that became an industry standard, but refused to license it to other manufacturers. Intel's profit margins on the chip, for which there is no competition, have been estimated at 80%.[51]

2. *Dominance of Large Corporations.* Another cost of patents is the tendency for large corporations to use patents to dominate a market and to intimidate small companies, who may be more innovative. It is common for large corporations to enter into cross-licensing agreement with each other that allow use of each other's technologies, but to deny licensing to small companies, who are far less likely to have patents to trade. Large corporations, such as IBM and AT&T, are also very sophisticated about patents and aggressively seek patents and seem particularly successful in obtaining broad patents. Such corporations also have far more resources to engage in litigation, and may be able to win infringement lawsuits on staying power alone.

3. *Validity of Patents.* Costs associated with invalid patents are another inevitable aspect of patents. Any patent system is plagued by errors resulting from lack of time to carefully examine patents and inexperience of examiners, and there are also cases where decisions are close and reasonable efforts by examiners are just not upheld later by the courts. Searches for prior art can also often be imperfect. Patent validity is at the present time a particular problem with software, because of examiner inexperience, the increase in the number of patent applications, lack of much prior art in the form of patents, particular difficulties in searching for software, and uncertainties about the patentability criteria for software. Most of these

51. Yoder. Intel Faces Challenge to its Dominance in Microprocessors. *Wall Street Journal*, April 8, 1991, A1.

problems are likely to be reduced in magnitude as the Patent and Trademark Office gains experience with software patents and develops more sophisticated search systems. Uncertainties resulting from software patentability issues, however, will only be resolved by further court cases or legislative action. The economic costs of invalid patents result from incorrect decisions made, such as the decision by a competitor to avoid using a particular technology that has an invalidly issued patent, and costs of litigation over infringement. Related costs, in addition, include the costs of applicants who misinterpret patentability criteria and who apply for a patent that is denied, or costs of inventors who fail to apply for a legitimate patent for the same reason.

Unanticipated Infringement. This cost involves competitors who reinvent a technology but later find that it was patented. Here the patent system did not in fact encourage the invention, and the rewards gained by the patent holders are spurious, although necessary to maintain the integrity of the system.

Allocative Costs. These are costs of patents that would have been made without patent protection, or with less patent protection.

Costs of Duplicative Research. The patent system motivates much research for the purpose of designing around patents. While most of this is wasteful and a clear cost, at least some duplicative research will result in unanticipated inventions that may be more significant than mere uninfringing duplicates of existing inventions.

Administrative Costs. The patent system includes the costs of the Patent and Trademark Office bureaucracy, the cost of search systems, and the cost of patent attorneys.

ASSESSING THE IMPACT OF SOFTWARE PATENTS

Assessing the impact of patents on the software industry is particularly difficult, because to do so we need to know the answer to the following questions not only today, but in the future. The questions are: (1) how many software patents are being issued? (2) what kind of software patents are being issued — for specific applications, or for general software methods? (3) how easily are software patents distinguished from hardware patents?

An Initial Analysis

In a crude initial study aimed at answering these questions, I examined the 348 patents in the "electrical" category issued on April 2, 1991, and selected out those patents that were either for software, for inventions that included software as part of a system, or in which software *could* be needed to realize the invention. (Patents are issued weekly, and are in 3 categories — "general and mechanical," "chemical," and "electrical.") Each patent was placed in one of six categories: (1) a software technique; (2) probably software; (3) might be either software or hardware; (4) probably hardware but could be software; (5) systems that included both hardware and software; and (6) interface or data storage formats. Decisions were made only on the basis of the drawing and the wording of the most significant claim for the patent as shown in the *Official Gazette*. The results are shown in the table below:

Category	No. of Pat.	Percent
Software	5	12%
Probably software	4	10%
Either software or hardware	15	36%
Probably hardware	8	19%
Software & hardware combinations	8	19%
Interface standard or data storage format	2	5%
TOTAL	42	100%

The five patents in the "software" category included a technique for providing interactive control over running computer programs, a technique for natural language translation (e.g., Japanese to English), a vehicle diagnostic system, a "silicon compiler" method, and an interactive statistical system. The "probably software" category included a speech recognition technique and two different vehicle brake control systems. The "either software or hardware" category included a data communications system and a method of arranging data on a random-access-memory for display. The "probably hardware" category included several video signal processing patents. The "software and hardware combination" category included two patents for autofocusing systems in cameras and a camera flash, all of which included microprocessors. The "interface standard or data storage format" category included a telecommunications transmission format and a magnetic storage media format.

While the sample here is very small, it does provide evidence that software patents are being issued, albeit in modest numbers.

What is particularly striking is the difficulty in distinguishing between patents that are clearly

SwRD

"hardware" or "software." More than a third were in the category "either hardware or software," and almost as many seemed to be one or the other but could have been otherwise. These decisions were made only on the basis of a single claim and drawing and, indeed, without a great deal of thought. Detailed analysis of the patents would probably allow most of the patents in the "either software or hardware" category to be placed in the "probably hardware" or "probably software" categories. But while the detailed specification of the patent would clearly indicate how the "best embodiment" of the invention was implemented, this does not mean that the invention cannot be implemented differently. Indeed, some patent specifications have been described as software because they were first reduced to practice in that form, but with the intention that they would eventually be manufactured as hardware. Deciding that a patent was definitely "hardware" or "software" would, for many patents, be quite difficult.

The construction of increasingly complex systems and the continuing trend to use multiple microprocessors will increasingly blur the distinction between software and hardware. This makes it very difficult to determine what a "software patent" is.

Estimates of the Number of Software Patents

The scale upon which software patents are now being issued—and, more importantly, that which they may be issued in the future—is not clear. If the sample of one week's patents analyzed above is representative, then it appears that several hundred patents are issued per year for software methods *per se* and more than 1000 patents are issued per year for inventions that in some way include software. This is consistent with an estimate reported earlier reported that there were about 200 patents issued in the first three months of 1989 for "software."[52]

What is particularly important, however, is the extent to which software patents may be issued in the future. One way of estimating this is by making use of statistical findings that relate patents issued in various areas of technology with research and development spending in those areas. In particular, Scherer has found that 85% of the variance in patenting for particular corporations can be "explained" by use of two factors: the amount of research and development spending for that company, and a measure of the average number of patents received per million dollars of R&D spending for a particular industry group. The number of patents received for each industry group varied from 0.45 per million dollars of R&D (for motor vehicles) to 3.98 (for industrial equipment). The mean for all industry groups was 1.70. (R&D spending was in 1974 dollars.)[53]

Using this data, we can estimate the number of patents for software that might result if indeed the software industry filed patent applications with a frequency similar to that of other technologies. The revenue for the U.S. software industry has been estimated at $62.7 billion.[54] Using an estimate of the proportion of revenue devoted to R&D in the software industry of 19.4%, this would suggest that the industry is now spending about $12. 2 billion per year in R&D.[55] This amount of spending would result in from about 2400 to 21,000 patents issued per year, or a most likely figure of about 9,000, if software patents were applied for with the same frequency as those in other industries.[56]

In addition, we can expect that foreign inventors would also obtain perhaps another 10 or 20% of these patents, despite the dominance of the U.S. software industry. U.S. software patents are frequently issued to inventors in Japan and Europe, and one of the software method patents described earlier, for machine translation, is from Japan.

We cannot predict the extent to which the software industry will file patent applications, nor is it clear why some industries apply for patents with a greater frequency than others. However, it does appear that the potential for patenting software is far higher, perhaps by an order of magnitude or more, than such patents are presently being applied for.

TOWARD A SOLUTION

What should be done about software patents? Are software patents a major threat to innovation in

52. Kanin, cite

53. Scherer. *The Propensity to Patent. International Journal of Industrial Organization. 1*, (1983), at 107.

54. Brandt, Schwartz, and Gross. *Can the U.S. Stay Ahead in Software?* BUSINESS WEEK. March 11, 1991, at 198.

55. This estimate is the mean of the proportion of R&D spending to revenue reported by three major software companies in the U.S., as reported in their most recent annual report. The figures are 15.3% for Microsoft, 17.0% for Lotus Development Corp., and 25.9% for Ashton-Tate, or a mean of 19.4%.

56. Data reported by Scherer was expressed in 1991 dollars using Bureau of Labor Statistics data for rises in the producer price index. This results in a "propensity to patent" of 0.20 per million dollars for the lowest industry group, 1.73 per million dollars for the highest, and 9.74 per million dollars for the industry average.

the software industry? Will expensive litigation drive small, innovative companies out of business, resulting in a less competitive and less innovative industry? More to the point, do the overall benefits of software patents exceed their costs, and, if so, what should be done about it?

The short answer to these questions is: we don't know. As the previous sections have suggested, while we know the mechanisms underlying the various benefits and costs of patents, it is very hard to assess their real impact. And while it appears that the tendency to patent software is already significant and continues to grow, we don't yet know whether this tendency will approach the levels seen in other technologies and thus be a truly important factor in the software industry.

Some of the uncertainty referred to above can be reduced by appropriate study. In particular, we need reliable data on the extent of software patenting, on the types of software-related technology that is being patented, and on the breadth and effects of specific software patents.

The patentability of software did not arise from any specific demand of the software industry, or legislative action by Congress, but evolved from court decisions that essentially recognized that, in terms of function, software was little different from hardware and that the same considerations for hardware ought to apply for software. This was not a policy decision that considered the economics of innovation in the software industry or the broader aspects of software technology. It may well be desirable to make such a policy decision that specifies that software is unpatentable and to implement it legislatively—but if so, it will be necessary to develop a clear criteria for distinguishing what is patentable hardware from unpatentable software. This is not easy, given that it involves policy, legal, and technical considerations, and it may not even be feasible to arrive at an acceptable criteria. The approach suggested by the League of Programming Freedom[57]—which provides that software implementations of patented inventions are not an infringement—is an interesting example of such a criteria that has the advantage of clarity. Other criteria need to be developed and assessed.

Despite the meager evidence concerning software patents, I offer the following assertions for discussion:

1. Because patent claims define inventions conceptually, rather than in terms of hardware or software, it is futile to attempt to allow patents for hardware but not software.

2. The desirability of patent protection for computer hardware, as well as software, is questionable, with little or no evidence that the benefits exceed the costs. A weakened patent system, such as that using compulsory licensing, may offer more net benefits.

3. Patents are well established and attempts to abolish the patent system generally or eliminate patents for information-processing-related inventions is probably politically impossible. However, it may be possible to introduce a compulsory licensing system.

4. The present criteria for patentability of software is unclear and results in undesirable uncertainty about whether particular software inventions are patentable and whether patents if issued will be upheld by the courts.

These assertions lead to the following proposal for changes to the patent statute. There are two parts to the proposal:

First, a clear criteria for software patentability would be provided that allows any algorithm to be patented. This would expand software patentability slightly, and, most significantly, remove a heavy cloud of uncertainty that now hangs over a minority of software patents and a slight cloud that hangs over nearly all software patents. To reduce any concern that patenting basic algorithms may prevent students or researchers from using algorithms in legitimate ways, a clause would also be added to the statute that would clarify and codify the case law that already holds that experimental use of a patented algorithm for student or research purposes is not an infringement, as long as that algorithm is not embedded in a product and sold.

Second, patents relating to information processing and computation would be subject to compulsory licensing. In this procedure, patents would be presumed when issued to be available for licensing on reasonable terms. If, however, special circumstances existed, the patent applicant could apply for an exception to compulsory licensing. Legitimate conditions for an exception might be small companies who need exclusive rights to a patent to establish a niche in the market. The Commissioner of Patents and Trademarks would be empowered, in such special cases, to allow exclusive rights for a portion of the patent term, after which licensing would be required.

Specific language for these statutory changes is provided in an appendix.

57. League of Programming Freedom, *Software Patents: Is This the Future of Programming?* DR. DOBB'S JOURNAL, November, 1990, 56.

This proposal is only one possible approach to reducing some of the problems that exist now with hardware and software patents in the computer industry, and that may help maintain a balance between benefits and costs of patents for software in the future. The overall effect would be to reduce the power of the patent system and to increase competition in the industry. Technology would be more broadly available, and the tendency for patent holders to demand excessive royalties even when they were willing to license their patents would be eliminated because they would be held to a cap of "reasonable" levels under threat of arbitration.

CONCLUSIONS

The rapidly increasing number of patents issued for software in recent years, questions about the validity of these patents, and the tendency for companies in the software industry to attempt increasingly strong legal protection for software all raise concerns about the impact of patents on innovation in the software industry.

The recent surge in patents is the result of a 25-year legal battle during which software was largely considered unpatentable that continued until 1981, and significant numbers of software patents were not issued until the late 1980s. While software is now broadly patentable, a complex and vague set of guidelines are used to test patentability that designates many patents—those involving mathematical equations—as unpatentable or puts them under a considerable cloud of uncertainty.

The newness of patents to software developers and the nature of the software industry have provoked resistance to the idea of patents on the basis that software is somehow "special" and should not be subject to patents. The problems posed by software patents, however, are not significantly different in nature from problems reported over the years about the patent system generally.

The existence of the patent system owes far more to history than empirical evidence of its success in motivating innovation. The extent to which patents are relied upon varies considerably for different technologies, and there are, for many technologies, real questions about whether the overall benefits of the patent system exceed its costs. In the case of the computing industry, in which economic efficiency and innovation are increasingly seen as resulting from collaboration, cooperation, and the lack of monopolies, the monopoly power of patents may be excessive.

Although software patents are now being issued in significant numbers, the potential number of patents that software technology could likely result in is far higher, by perhaps an order of magnitude, than the rate at which patents are now issued.

One possible revision to the patent statute that could improve economic competitiveness for software is proposed for discussion. This proposal has two principal parts: (1) a definition of software patentability that makes it far clearer what software is patentable, while expanding slightly the scope of patentability; and (2) a compulsory licensing provision that requires that information processing inventions—with some exceptions—be licensed on reasonable terms to anyone.

This solution would reduce many of the problems that have been raised concerning patents—both software and hardware—in the computing industry. It could be implemented as an experiment and provide both an interim solution and potentially useful evidence about the effects of slightly weakening the patent monopoly on technological innovation in the computer industry.

APPENDIX

Text of Proposed Revisions to Patent Statute

Title 35 of the United States Code is revised as follows:

The following paragraph is to be added at the end of § 101, "Inventions patentable":

An algorithm, or effective procedure, for processing information is a patentable process, and an appratus that makes use of such a method is a patentable apparatus, without regard to whether that algorithm makes use of a mathematical representation, as long as that procedure is applied to a useful end.

The following paragraph is to be added to the end of § 271, "Infringement of Patent":

(h) It shall not be an act of infringement for a patented apparatus to be manufactured, or a patented apparatus or process used, as long as it is for student, scientific research, or experimental purposes, and that the process or apparatus is not sold or used in a product that is sold.

The following paragraph is to be added to the end of § 154, "Conditions and term of patents":

All inventions that consist primarily of information processing processes or apparatus are subject to compulsory licensing of patent rights, on reasonable terms, to any person who desires such a license, except that patent applicants may, at any time prior to issuance of the patent, apply to the Commissioner for a waiver of the compulsory licensing requirement. Such a waiver may be granted in exceptional circumstances in which patent rights are likely to play a significant role in allowing a small company to enter a new market, help in the establishment of a software reusable components industry, or for other purposes as determined under regulations promulgated by the Commissioner. Such a waiver would apply only to a specified portion of the patent term, not to exceed 7 years. No waivers shall be granted for patents for human-computer interfaces or interfaces between hardware or software components of information processing systems. Any person entering into a license agreement with a patent holder, who believes that the patent assignee is demanding excessive royalties for such a license, can apply to the Commission, who shall arrange for arbitration.

A Note on Citations and Sources. The federal statutes and court decisions cited in this paper can typically be found in any law library. Statutes relating to patents are found in Title 35 of the United States Code. The citation 35 U.S.C. § 103 (1988), for example, refers to title 35, U.S. Code, section 103, as codified in 1988. Supreme court cases are cited, for example, in the *U.S. Reports*. Decisions of the appeals courts are cited, for example, as 726 F.2d 734 (CAFC, 1984), which refers to volume 726 of the *Federal Reporter*, 2nd series, page 734, in a 1984 decision of the Court of Appeals for the Federal Circuit. Specific patents can be obtained by sending the patent number and $1.50 for each patent to the Patent and Trademark Office, Washington, DC 20231.

CRITERIA FOR SOFTWARE PATENTABILITY

Craig A. Will

Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311
(703) 845-6635

The following description of the criteria for patenting a software-related invention is based on decisions by federal courts, primarily in cases where unsuccessful patent applicants have appealed to the courts.

To be patentable, an invention must meet the standard criteria of being new (novel), not obvious to a person of ordinary skill in the art, and useful. While obviousness is a significant issue, the standard for what is obvious can be very subjective and there is a tendency for patent examiners to focus on novelty rather than obviousness because novelty is much easier to assess. There is some justification for this focus on novelty because inventions frequently appear in retrospect to be obvious despite the fact that skilled personnel may have failed to conceive of the invention over a period of years. In general, the standard for nonobviousness for a computer software-related invention is not necessarily very high, and much criticism has been directed at the Patent and Trademark Office for issuing patents for inventions that critics view as obvious.

Although usefulness is also required for patentability, this criterion has been interpreted liberally and is rarely a problem, although it will prevent the patenting of an invention that depends, for example, on hardware that is clearly beyond today's technological capability to build. An algorithm only works some of the time would be patentable, however, as long as it has some arguable utility.

In the case of software inventions, the key issue is the requirement that an invention must be in one of the four classes of "subject matter" specified in the patent statute—a "process, machine, manufacture, or composition of matter." A software invention can be patented as either a machine, a process, or both, depending upon how it is described in the patent claims. Each patent claim defines a unique aspect of the invention, with patent applications typically including from one to four or five claims for simple inventions and up to perhaps 30 to 50 for particularly complex inventions. A claim for a machine defines the invention in terms of a combination of components that interact in a specified way to achieve a particular end. Claims that define combinations of components in which the components are described in terms of their function rather than their structure are specifically allowed by the patent statute.[1] These "functional" claims, as well as a tendency in patent claims to describe machines as "systems" and components in abstract terms that can be either software or hardware, often make it difficult to determine whether a given patent claim is more likely to be implemented in software, hardware, or a combination.[2] In addition, according to the "doctrine of equivalents," a patent claim covers not only the literal description of the claim but also inventions that do "substantially the same thing in substantially the same way."[3]

Claims for a "process" (also called a "method") are written as a sequence of steps that make up the process. It is common for software-related inventions to be described by a series of claims, with some claims describing the invention as a machine and others describing the methods used in the invention.

Court decisions since the late 1960s have interpreted this statutory subject matter requirement for software. According to current Patent and Trademark Office practice, the principal barrier to patenting a software invention—assuming that it meets the usual tests of being new and not obvious—is that an invention not primarily consist of a "mathematical algorithm."

Unfortunately, it is not completely clear what constitutes a mathematical algorithm, nor are the conditions under which an invention's use of a mathematical algorithm makes the invention unpatentable particularly clear.

---

1. 35 U.S.C. § 112.
2. See, e.g., the analysis of a sample of software patent claims provided in Will, *Software Patents and Economic Competitiveness*, PROCEEDINGS OF THE WASHINGTON ADA SYMPOSIUM, June, 1991.
3. Chisum, CHISUM ON PATENTS, 1990 Edition.

The term "mathematical algorithm" clearly has a much narrower meaning than does the term "algorithm"—it is recognized that any claim for a method or process is an algorithm as the term is conventionally used in computer science—a sequence of steps that are executed to carry out a procedure. The Supreme Court has defined a mathematical algorithm rather unhelpfully as "a procedure for solving a given type of mathematical problem."[4]

Despite the confusion of differing interpretations and explanations of a mathematical algorithm, the intent of the rule is the notion that "laws of nature," "scientific principles," "physical phenomena," and "abstract ideas" cannot be patented. There has never been a clear analysis of just how a mathematical algorithm (or formula or equation) relates to these forbidden entities, and thus the court decisions and patent office policies have had difficulty interpreting this concept in an entirely consistent manner.

It is also apparent that the mere use of mathematical symbols and operations or equations to describe an invention does not necessarily imply that a mathematical algorithm is involved. However, there is much loose language in court decisions referring to mathematical equations in a negative way. For example, the court in *Walter* referred to a mathematical algorithm as "methods of calculation, mathematical formulas, and mathematical procedures generally." Just as patent attorneys in the 1970s and early 1980s tended to write patent claims for software in terms that made them seem like hardware, claims drafters tend to avoid mathematical descriptions when at all possible, to avoid difficulties.[5]

It is the attempt to patent a scientific principle itself, not the attempt to apply a scientific principle (which all inventions do) that it is prohibited. In order to apply this rule, the courts have developed what is known as the "two-part" test resulting from the cases of *Freeman, Walter and Abele.*[6] The steps of the two-part test are as follows:

1. The claim is analyzed to determine whether it "recites" a mathematical algorithm. If it does not recite such an algorithm, the invention is patentable. If it does recite an algorithm, go to Step 2.
2. The claim is analyzed to see if the algorithm is specifically applied to physical elements (in a machine) or to steps (of a process).

Reciting of mathematical algorithm.

The presence of mathematical symbols and operations in equation form is not the only test for a mathematical algorithm. The court in *Freeman,* for example, stated that "A claim which substitutes, for a mathematical formula in algebraic form, 'words which mean the same thing,' nonetheless recites an algorithm in the *Benson* sense."

Specific application of algorithm.

To be patentable, the algorithm must be specifically "applied in any manner to physical elements or process steps."[7] There are no clear tests for distinguishing what this means; there are instead guidelines, based on court decisions. It is possible to extract six specific guidelines from the decisions, as follows:

1. Simply taking the result of a computation and using it in some way —so-called "post-solution activity"—does not transform an unpatentable algorithm into a patentable invention, as the Supreme Court ruled in *Flook.*
2. Attempts to simply state that the use of an algorithm is limited to a particular problem or application—so-called "field of use limitations"— will also not result in a patentable invention.[8]
3. Third, including "data-gathering" steps that determine values for the variables used in the equations will not make an unpatentable algorithm into a patentable invention.[9]

---

4. Gottschalk v. Benson. 409 U.S. 64 (1972). The language used in the *Benson* decision has been cited again in Parker v. Flook (437 U.S. 584, 1978) and in Diamond v. Diehr (450 U.S. 173, 1981).
5. See, e.g., the claims for the controversial Karmarkar patent, which describe the algorithm involved in terms of a visual solution space. U.S. Patent 4,744,028.
6. In re Freeman. 573 F.2d 1237 (CCPA, 1978); In re Walter. 618 F.2d 758. (CCPA, 1980); In re Abele. 684 F. 2d 902 (CCPA, 1982).
7. In re Abele. 684 F. 2d 902 (CCPA, 1982).
7. Parker v. Flook. 437 U.S. 584 (1978).
8. Diamond v. Diehr. 450 U.S. 173 (1981).
9. In re Richman. 563 F.2d 1026 (CCPA, 1977). In re Grams. 888 F.2d 835 (CAFC, 1989).

4. A process that transforms a signal representing a physical state to another state is patentable.[10] However, the mathematical manipulation of abstract data is not patentable[11]

5. Attempts to limit methods to use in specific machines will not transform an unpatentable algorithm to a patentable invention.[12] However, an apparatus consisting of multiple components, of which one is defined in physical implementation terms and the rest as a mathematical formula, is not unpatentable simply because it operates according to an algorithm.[13]

There is much ambiguity in the court decisions leading to the above guidelines and and thus in cases near the boundaries it is unclear whether the Patent and Trademark Office will issue a patent for a particular software invention, or whether such a patent, if issued, will be upheld should it be tested in court in an infringement action. However, most software inventions do not involve mathematical equations and formulas and thus most software is clearly patentable.

For more detailed information and discussion of the criteria for software patents, a number of additional sources are available. A 1989 legal analysis published by the Patent and Trademark Office[14] presents their interpretation of the court decisions, as does the PTO *Manual of Patent Examining Procedure*.[15]

Two papers present particularly detailed reviews of the literature on software patentability. Chisum[16] presents the case for broad patent protection of software, while Samuelson[17] presents the case against patent protection for software.

A Note on Citations and Sources. The federal statutes and court decisions cited here can typically be found in any law library. Statutes relating to patents are found in Title 35 of the United States Code. The citation 35 U.S.C. § 103 (1988), for example, refers to title 35, U.S. Code, section 103, as codified in 1988. Supreme court cases are cited, for example, in the *U.S. Reports*. Decisions of the appeals courts are cited, for example, as 726 F.2d 734 (CAFC, 1984), which refers to volume 726 of the *Federal Reporter*, 2nd series, page 734, in a 1984 decision of the Court of Appeals for the Federal Circuit.

---

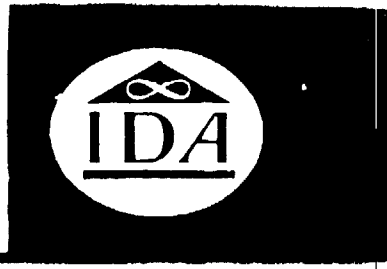
10. In re Johnson. 589 F.2d 1070 (CCPA, 1978). In re Taner. 681 F.2d 787 (CCPA, 1982). In re Sherwood. 613 F.2d at 819 (CCPA).
11. In re Walter. 618 F.2d 758. (CCPA, 1980).
12. In re Castelet. 562 F.2d 1236 (CCPA, 1977).
13. In re Iwahashi. 888 F.2d 1370 (CAFC, 1989).
14. *Report on Patentable Subject Matter: Mathematical Algorithms and Computer Programs.* 1106 *Official Gazette*, 5. (August 9, 1989). See also the following critique and discussion of these guidelines: *The Patentability of Computer Programs: The PTO Guidelines, In re Grams and In re Iwahashi.* 6 COMPUTER LAWYER 21 (December, 1989).
15. Patent and Trademark Office. U.S. Department of Commerce. *Manual of Patent Examining Procedure*. Washington, DC. Publication 605. See especially section 2106.
16. Chisum, *The Patentability of Algorithms*. 47 UNIVERSITY PITTSBURGH LAW REVIEW 959 (1986).
17. Samuelson. *Benson Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-related Inventions.* 39 EMORY LAW JOURNAL 1025 (1990).

PATENTS AND SOFTWARE REUSE

Craig A. Will, Ph.D.

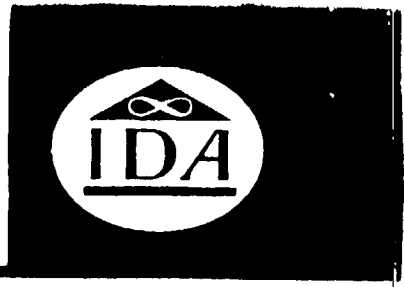
Computer & Software Engineering Division
Institute for Defense Analyses
Alexandria, Virginia

ISSUES



- Does the increased tendency to patent software pose difficulties that might discourage software reuse?
- If so, how can these difficulties be minimized?
- Can software patents help encourage software reuse?

BACKGROUND ON PATENTS



What do patents protect?

- "inventions" at a conceptual or algorithm level

What rights does inventor get? Right to exclude others from:

- making
- using
- selling

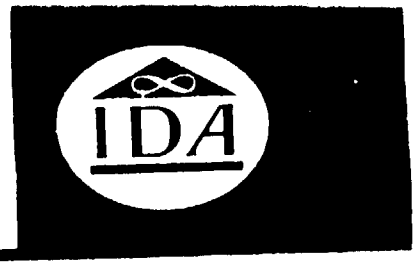
an invention for 17 years after the patent is issued.

Who is considered the true inventor?

- The first to conceive it
- But right can be lost by concealing or abandoning it

As a way of protecting intellectual property, patents are relatively expensive and uncertain

THE PATENT EXAMINATION PROCESS



Patents are issued for:

- machines
- processes
- some others (compositions of matter)

For a patent to be granted, inventions must be:

- novel
- non-obvious
- useful

Patent applications are examined to insure that they meet the criteria

- applications kept secret during examination process
- examination usually takes 12 to 36 months

IMPLICATIONS OF PATENT RIGHTS

Patent holder can demand:

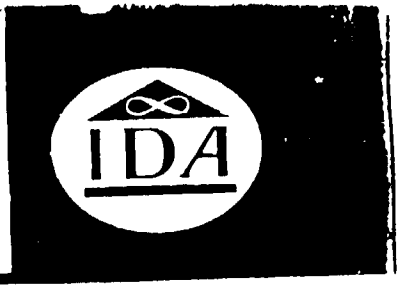
- royalties
- that use or sale of infringing product be stopped

Patents issued are presumed to be valid

Independent invention is not necessarily a defense and is never a defense unless it was first

Secret examination and search difficulty poses risk

HISTORY OF SOFTWARE PATENTS



Software was at first considered unpatentable

- mental step
- scientific principle
- practical difficulties of searching prior art

From 1960s to 1981 software patents were subject of legal battle

- appeals court generally in favor
- Patent and Trademark Office strongly opposed
- 1972 Supreme Court decision held that most software is apparently unpatentable
- a few patents still issued even after 1972

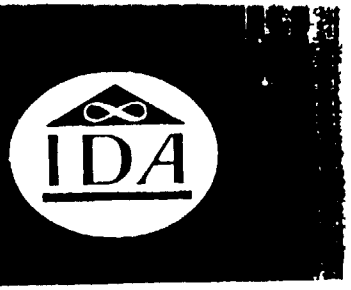
HISTORY OF SOFTWARE PATENTS (continued)

A 1981 Supreme Court decision in favor had strong effect:

- Patent and Trademark Office reversed its position
- Industry slow to realize it
- By 1988 large numbers of software patents issued
- Currently strong tendency to file applications

Much criticism of potential impact of software patents

HOW VALID ARE SOFTWARE PATENTS NOW BEING ISSUED?



Patent examiners have great difficulty searching patented prior art and, particularly, published prior art

Examiners may be weak in computer science knowledge

- have trouble determining whether invention is non-obvious

Difficulty in applying patentability criteria

- some patents issued arguably do not meet criteria (e.g., Bell Labs patent for new solution to Traveling Salesman Problem).

CRITERIA FOR PATENTING SOFTWARE

Machines or processes that include software are patentable, however:

- A "mathematical algorithm" cannot be patented.
- The *use* of a mathematical algorithm in a particular invention can be patented under certain conditions.

Criteria requires making distinctions that can be very difficult

Whether an invention is patentable or not can be very dependent on the way the patent application claims are drafted

Criteria may change with:

- New court decisions
- Possible legislation

(Supreme Court has never issued a clear decision)

Most inventions involving software do not involve a mathematical algorithm and criteria is relatively clear.

REUSE AND THE RISK OF INFRINGEMENT

Software patents and trend toward software legal protection increase the risks of infringement even for innocent parties.

Reuse causes these risks to be magnified because:

- It increases the number of players
- Those with the knowledge to avoid infringement may not be those with responsibility for it

Patent infringement can result from reused components if:

- The component itself infringes
- An invisible internal process in the component infringes
- It is used as a part in a combination or application that infringes

Component manufacturers are not responsible for infringement as long as component has a substantial non-infringing use

PATENTS AS ENCOURAGING SOFTWARE REUSE

Patents can encourage reuse in three ways:

- The risk of infringement can make developers wary. A component that is sold as part of a package, with a patent analysis, search, and perhaps indemnity for infringement would be valuable.
- Components that are protected by patents are more likely to survive because they are much less likely to run into infringement problems, and if so can potentially escape by trading licenses.
- Patent protection for components can provide monopoly power and economic rewards that allow developers to put considerable resources into making that component reusable.

PROSPECTS FOR CHANGE

There is considerable opposition to software patents within the software industry.

Congress is studying intellectual property protection for software.

Wholesale elimination of software patents by legislation is extremely unlikely due to lack of consensus within the industry.

Fine-tuning by legislation quite possible:

- First-to-file rather than first-to-conceive gets patent
- Mandatory licensing under some conditions
- Raise standard for non-obviousness for software
- Clearer criteria for software patentability
- Eliminate patent for user interfaces and component interfaces
- Require patent holder to develop invention within specified time frame

CONCLUSIONS

- Software is generally patentable and patents will almost certainly play an increasing role in the software industry in the future
- Software patents can be risky and the criteria for patentability uncertain, but most patents are likely to be valid
- Software reuse can result in increased risk of infringement liability, and care should be taken to minimize this liability
- Patents can provide strong mechanisms that encourage software reuse

ATTACHMENT D

USER'S NON-DISCLOSURE AGREEMENT

The User requests some or all of the following from the Defense Software Repository System ("DSRS"): data, technical data, computer software, computer programs, source code, firmware, and other information of like kind, type or quality, either commercial or non-commercial, all of which may be subject to limited rights, restricted rights, Government purpose license rights, patents, copyrights, trade secret rights, or other confidential or proprietary constraints (collectively, the "Data"). In consideration therefore, the User agrees:

1) that the Data extracted from the DSRS shall be used only for Government, non-commercial or non-profit purposes;

2) to strictly abide by and adhere to any and all restrictive markings placed on the Data, and the User shall not knowingly disclose or release the Data to third parties who are not engaged in work related to Government, non-commercial, or non-profit purposes;

3) that any restrictive markings on the Data shall be included on all copies, modifications, and derivative works, or any parts or portions thereof, in any form, manner or substance, which are produced by the User including but not limited to incorporation of the Data into any other data, technical data, computer software, computer programs, source code, or firmware, or other information of like kind, type or quality. In all such events, User shall clearly denote where such Data initiates and concludes by use of annotations or other standard markings.

USER'S WAIVER OF WARRANTIES AND LIMITATION OF DAMAGES AGREEMENT

THE USER AND THE DEFENSE SOFTWARE REPOSITORY SYSTEM ("DSRS") AGREE THAT:

1) NO GUARANTIES, REPRESENTATIONS, OR WARRANTIES EITHER EXPRESS OR IMPLIED SHALL BE CONSTRUED TO EXIST IN ANY LANGUAGE, PROVISION, OR TERM CONTAINED IN THESE MATERIALS OR IN ANY OTHER DOCUMENTATION PROVIDED HEREWITH (ALL SUCH ITEMS ARE COLLECTIVELY REFERRED TO AS THE "AGREEMENT"), AND FURTHERMORE, THE DSRS DISCLAIMS AND THE USER WAIVES AND EXCLUDES ANY AND ALL WARRANTIES OF MERCHANTABILITY AND ANY AND ALL WARRANTIES OF FITNESS FOR ANY PARTICULAR PURPOSE;

2) THE USER SHALL OBTAIN FROM THE DSRS ALL OF THE "DATA" (DEFINED IN THE USER'S NONDISCLOSURE AGREEMENT ABOVE), OR ANY OTHER PRODUCTS OR SERVICES CONTEMPLATED BY THE AGREEMENT, IN AN "AS IS" CONDITION;

3) IN NO EVENT SHALL THE DSRS BE LIABLE FOR ANY ACTUAL, DIRECT, GENERAL, CONSEQUENTIAL, SPECIAL, INDIRECT, INCIDENTAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO, LOST PROFITS, EXPECTATION DAMAGES, THIRD PARTY CLAIMS, OR ANY OTHER COSTS, FEES, OR EXPENSES OF ANY NATURE WHATSOEVER AS A RESULT OF THE USE OF DATA OR ANY OTHER PRODUCTS OR SERVICES, RELATED TO THE AGREEMENT, WHETHER OR NOT USED IN ACCORDANCE WITH ANY APPLICABLE INSTRUCTIONS OR MANUALS PROVIDED THEREWITH (IF ANY) OR DUE TO ANY WARRANTIES, GUARANTEES, OR REPRESENTATIONS, EITHER EXPRESS OR IMPLIED, WHICH MAY ARISE DUE TO THE TRANSACTIONS CONTEMPLATED BY THE AGREEMENT;

4) THE USER AND THE DSRS INTEND THAT THIS AGREEMENT SHALL BE GOVERNED BY THE LAWS OF THE UNITED STATES OF AMERICA.

Attachment E

Top Five Issues 10/26/92

Priority	Tracking Number	Date Entered	Issue Raised By	Issue/Risk Description	Required Action	Responsibility	Closure Date
1	8	4/30/92	Peer Review Æ	"Biting off more than we can chew" -- Lack of clarity and focus.	√ Record Level1 goal priorities	Æ	8/16/92
					√ Map list of goals to peer review comments, Concept, and other documents to keep within established focus	Æ	8/20/92
					√ Develop draft list of measurable objectives	Æ	8/20/92
					√ Review Goals and Objectives with Level 3, Level 2, Level 4 - Commitment/Consensus	Æ, Level 2, Level 3 PM, Chief Scientist	9/25/92
					• Rescope mission (White Paper now in process)	UH, JSC, Æ	
2	13	9/25/92	McKay, Garman	Neither RBSE or JSC is represented on the panel which is tasked to rewrite NMI 2210.2b	Present the problem to Frank Peñaranda	Æ (Garman accomplished this)	9/25/92
					Follow up with FEP	Æ	
3	2	4/30/92	Æ Peer Review	No clear assurance that ASV3 will meet customer needs – The only formal criteria are in the requirements document, which was written without strong input from key target customer groups.	• Involve key target customers in beta test. Systematically acquire user comments	UH, MountainNet	
					√ Lengthen NASA support through extended beta test period	UH	9/1/92
					√ Define pilot programs	UH, Æ	9/25/92
					• Build in systematic test into pilots	UH	
4	3	4/30/92	Æ	AdaNET has no clear claim to NASA-developed software/The distinction between AdaNET and COSMIC objectives is unclear to customers and sponsors	√ Establish informal working agreement with COSMIC	Æ/MountainNet	
					√ Describe points of agreement in draft MOU	Æ	7/5/92
					√ AdaNET/COSMIC initial draft agreement Signed	UH/Æ/MountainNet	9/25/92
					• Conduct workshop to iron out details	UH/Æ/MountainNet	
					• Formalize MOU		
5	9	4/30/92	MountainNet Æ	Legal issues: e.g. copyright/patent infringement liability	• Acquire and use the acquisition and distribution processes developed by ASSET	Æ/MountainNet	
					√-- Contact Foreman and request	Æ	7/31/92
					• Leverage COSMIC experience	MountainNet	
					• Only distribute NASA -developed software when processed for copyright/patent issues by NASA TUO's	MountainNet	
					√ Acquire government studies on subject - (attached to Sept. Prog Report)	Æ	