Texas Wesleyan Law Review

Volume 8 | Issue 3 Article 3

7-1-2002

Legal Protection for Software: Still a Work in Progress

Robert W. Gomulkiewicz

LEGAL PROTECTION FOR SOFTWARE: STILL A WORK IN PROGRESS

Robert W. Gomulkiewicz†

I. INTRODUCTION

Software began as geekware—something written by programmers for programmers. Now, software is a business and consumer staple. Cryptic character-based user interfaces have given way to friendly graphical ones; multi-media is everywhere; people own multiple computers of varying sizes; computers are connected to one another across the globe; email and instant electronic messages have replaced letters and telephone calls for many people.

The issue of whether the law should protect software seems quaint to us now. Over the past twenty-five years, legislatures and courts have concluded that copyright, patent, trade secret, trademark, and contract law all can be used to protect software. Yet, the debate about how much protection the law should provide is as vigorous today as ever.

The debate has two conflicting perspectives. On one side are those who say that the current collection of laws does not protect software enough. They point out that every year the software industry loses billions of dollars to copyright infringers known as software pirates, and that renegade programmers use their craft to corrupt software with viruses, break into computer systems to steal data, and shut down entire computer networks for sport. They remind us that some companies spend significant resources developing digital databases that may not be copyrightable, yet Congress has not passed database protection legislation.

On the other side are those who believe that legal protection of software is too strong already. They accuse software publishers of us-

† Mr. Gomulkiewicz is an attorney in private practice who advises clients on licensing, intellectual property, e-commerce, and technology-related legal issues. He lectures at the University of Washington School of Law. Formerly, he was an Associate General Counsel with Microsoft Corporation and chair of the UCITA Working Group of the Business Software Alliance. The views expressed in this essay are the personal views of the author.

DOI: https://doi.org/10.37419/TWLR.V8.I3.2

ing patents to stifle innovation. They protest that recent amendments to the Copyright Act suppress the fair use of information. They worry that standard form contracts unfairly extend intellectual property protection.

This Essay traces the debate about legal protection for software from its early days to the present. It describes the issues that legislatures and courts have faced over the years and why many of those issues are back on the table today.

II. WHAT IS "SOFTWARE"?

Programmers develop software by writing instructions in a computer language such as Basic, C++, or Java.[1] These human-written instructions are known as source code.[2] The programmer then uses a tool called a compiler to convert source code into machine-readable object code. Object code is often called machine-readable code because it tells the computer what to do. When many users think about software, they think about the object code that is on a diskette or hard drive.

Users also think about software in terms of what they see when it runs. The user sees the visual displays that the software generates. In the early days of computers, software generated character-based displays, but now most software programs generate graphical visual displays.

Defining software as object code, source code, and visual displays is only one way to think about software. Another way to think about software is to examine what the software does. People often put software into categories such as operating systems, applications, middleware, utilities, and developer tools. Over time, however, these categories have changed and the boundaries have blurred or even vanished.

These traditional ways of describing software categorize the software in relatively technical terms, the way a programmer would look at things. The current trend is to talk about software in terms of the user experience or the service it provides. Software publishers sometimes use the catch phrase "software as a service" to describe this shift in perspective.[3] The ultimate goal of some software developers is to create "an intuitive interface" that would permit "computers, net-

1. *See* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1243, 219 USPQ 113, 115 (3d Cir. 1983); BILL GATES, THE ROAD AHEAD 1–2, 24–29 (1996).

2. Source code often includes comments from the developer explaining the purpose of the instructions he or she wrote.

3. *See* David Coursey, *Why Web Services Will Be the Next Big Thing*, ZDNET, *at* http://www.2dnet.com/anchordesk (Feb. 28, 2002) (on file with the Texas Wesleyan Law Review).

works, operating systems and commands . . . to become invisible" to the user.[4]

III. COPYRIGHT PROTECTION: THE OLDEN DAYS

One of the early questions facing software developers was whether or not source code and object code were copyrightable. Congress established a commission to study the matter.[5] This commission, known as the National Commission on New Technological Uses of Copyrighted Works (CONTU), concluded that computer programs were copyrightable in both their source and object code forms.[6] Congress passed several amendments to the Copyright Act[7] to implement CONTU's recommendations.[8]

At the same time, litigants challenged the copyrightablility of computer programs in court. A computer maker, Franklin Computer, copied Apple Computer's operating system software onto Franklin's computers without Apple's permission, and argued that obtaining Apple's permission was unnecessary because Apple's software was not copyrightable.[9] The court, citing CONTU's report and Congress's amendments to the Copyright Act, concluded that both source and object code qualified for copyright protection.[10]

Harder questions followed. For example, should copyright law protect both the literal and the non-literal aspects of software?[11] The literal aspects of software are the source and object code, as well as the visual displays. The non-literal aspects are the structure, sequence, and organization of the program's code and visual displays.

Courts have varied in their willingness to allow programmers to use copyright to protect the non-literal aspects of software. A high-water mark of copyright protection came in *Whelan Associates v. Jaslow*

4. Tim Berners-Lee, *Raising the Full Potential of the Web*, WORLD-WIDE WEB CONSORTIUM (Dec. 3, 1997), *at* http://www.w3.org/1998/02/Potential.html (last visited June 26, 2002) (on file with the Texas Wesleyan Law Review).

5. Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201, 88 Stat. 1873–75.

6. Copyright scholar Melville Nimmer filed a concurring opinion, however, in which he expressed concern that copyright law might be stretched to the breaking point if applied to software. *See* CONTU, FINAL REPORT, 66–69 (1978) (Nimmer, Comm'r, concurring).

7. *See* 17 U.S.C. §§ 101, 117 & notes (2000) (Historical and Revision Notes).

8. For a contrary view on whether software should be copyrightable, see Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663 (1984).

9. Apple Computer Corp. v. Franklin Computer Corp., 714 F.2d 1240, 1243–44, 219 USPQ 113, 115–17 (3d Cir. 1983).

10. *Id.* at 1246–49, 219 USPQ at 118–21.

11. A long line of non-software copyright cases held that both the literal and non-literal aspects of a copyrighted work deserve copyright protection. *See, e.g.*, Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1164–71, 196 USPQ 97, 102–09 (9th Cir. 1977). In a musical composition, the literal aspects are the notes and lyrics; the non-literal aspect is the overall concept and feel of the composition—its style, if you will.

Dental Laboratory, Inc.[12] In *Whelan*, the court allowed a programmer to keep another programmer from copying the overall design of software used to run the business operations of a dental office.[13] The court in *Whelan*, in separating the non-copyrightable idea from the copyrightable expression, ruled that the idea was automating a dental office and that everything else was copyrightable expression. The result was broad protection for the non-literal aspects of the software program.

Most courts did not follow the *Whelan* decision. Instead, courts concluded that although the non-literal aspects of software are copyrightable, the scope of protection should be scrutinized carefully.[14] These courts used traditional tools of copyright analysis to get at the core of what non-literal aspects of software were deserving of protection and, in many cases, not much copyrightable material remained at the end of the analysis. Courts used the same approach in deciding whether and how much to protect the non-literal aspects of software visual displays.[15]

The court in *Lotus Development Corp. v. Borland International, Inc.*[16] offered the most skeptical view of copyright protection for the non-literal aspects of software. In the *Lotus* case, the court decided that many of the non-literal elements of *Lotus*'s 1-2-3 spreadsheet were not copyrightable at all. This case went to the U.S. Supreme Court where the circuit court's ruling was affirmed by a 4-4 tie vote.[17]

During this same period of time, courts wrestled with other copyright-related issues. Courts were eager to allow programmers to protect their software, but became wary when programmers pushed the claimed protection too far. For example, the court in *Lewis Galoob Toys, Inc. v. Nintendo of America, Inc.*,[18] ruled that Nintendo could not use copyright law to thwart the development of add-on software, and in *Sega Enterprises Ltd. v. Accolade, Inc.*,[19] the court ruled that Sega could not use copyright law to thwart software interoperability. In *Lasercomb America, Inc. v. Reynolds*,[20] the court ruled that a programmer's use of a license agreement to extend the copyright pe-

12. 797 F.2d 1222, 230 USPQ 481 (3d Cir. 1986).

13. *Id.* at 1229, 1248, 230 USPQ at 482–83, 500.

14. *See, e.g.,* Gates Rubber Co. v. Bando Chem. Indust., 9 F.3d 823, 803–46, 28 USPQ2d 1503, 1503–19 (10th Cir. 1993); Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 22 USPQ2d 1429 (9th Cir. 1992); Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 696–717 (2d Cir. 1992).

15. *See, e.g.*, Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 32 USPQ2d 1086 (9th Cir. 1994); Data East USA, Inc. v. Epyx, Inc., 862 F.2d 204, 9 USPQ2d 1322 (9th Cir. 1988).

16. 49 F.3d 807, 815, 34 USPQ2d 1014, 1021 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996).

17. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996) (4-4 decision).

18. 964 F.2d 965, 22 USPQ2d 1857 (9th Cir. 1992).

19. 977 F.2d 1510, 1527–28, 24 USPQ2d 1561, 1573–74 (9th Cir. 1993).

20. 911 F.2d 970, 979, 15 USPQ2d 1846, 1854 (4th Cir. 1990).

riod amounted to copyright misuse, resulting in the suspension of the programmer's copyright until the misuse was cured.

IV. TRADE SECRET, PATENT, TRADEMARK, AND CONTRACT: THE OLDEN DAYS

In the 1970's, 1980's, and 1990's, software developers focused on securing copyright protection for software. At the same time, many developers also held their source code as a trade secret. Few seemed to question whether software could be held as a trade secret although some trade secret-related issues did emerge. One important issue centered on the right of programmers to take ideas and code from one development effort to another, and the use of non-compete agreements to curb this practice.[21] Another issue was whether a mass market license agreement with a "no reverse engineering" clause could effectively protect trade secrets that could otherwise be obtained by reverse compiling the object code to discover the underlying source code.[22]

Most software developers did not patent software in the 1970's and 1980's. The reasons for ignoring patent protection varied. Some software developers concluded that patents were probably not obtainable.[23] Others believed the expense and hassle of obtaining patents was not worth the potential payback. Still others opposed the patenting of software on philosophical grounds.[24] As the 1990's dawned, however, many software publishers began to realize the importance of

21. *See, e.g.,* Integrated Cash Mgmt. Servs., Inc. v. Digital Transactions, Inc., 920 F.2d 171, 172–74, 17 USPQ2d 1054, 1054–57 (2d Cir. 1990); Dynamic Research Corp. v. Analytic Scis. Corp., 400 N.E.2d 1274, 1285, 209 USPQ 321, 330–31 (Mass. App. Ct. 1980).

22. *See generally* Andrew Johnson-Laird, *Software Reverse Engineering in the Real World*, 19 U. DAYTON L. REV. 843 (1994).

23. *See* Gottschalk v. Benson, 409 U.S. 63, 71–73, 175 USPQ 673, 676–77 (1972).

24. *See* The League For Programming Freedom, *Software Patents: Is This the Future of Programming?*, DR. DOBB'S J., Nov. 1990, at 56.

obtaining patents[25] and courts began to rule that the patent laws covered software.[26]

Software developers used trademark law to protect product names and, in some cases, product feature names. They used trademarks to indicate that certain software was compatible with other software, such as Microsoft's "Windows Compatible" campaign, and to highlight one aspect of a computer system, such as Intel's "Intel Inside" campaign. Some companies used trademarks in an attempt to prevent other companies from creating compatible software, although these attempts were largely unsuccessful.[27]

As software became a mass-market consumer item, software developers began to offer their software under standard-form license contracts.[28] These contracts described what the end user could do with the software, what warranties accompanied the software, and other contract terms. Many commentators challenged the usefulness[29] and enforceability[30] of these mass-market licenses, and the earliest cases examining these licenses seemed to cast doubt on the issue.[31] Despite

25. *See* John A. Gibby, *Software Patent Developments: A Programmer's Perspective*, 23 RUTGERS COMPUTER & TECH. L.J. 293 (1997). Companies such as I.B.M. obtain the lion's share of computer-related patents. Margaret Kane, *IBM Receives the Most 2001 Patents*, ZDNET NEWS, *at* http://cma.zdnet.com/texis/techinfobase/techinfobase (Jan. 10, 2002). In 1991 Microsoft owned 10 patents; by February 2001 Microsoft owned over 1500 United States patents. In 1994 Oracle owned no patents; by 2001 Oracle owned 249 patents. Commercial software companies are not the only ones who own and enforce patents. Universities are in the act as well. *See* Goldie Blumenstyk, *Cornell U. Sues Hewlett-Packard Alleging Patent Infringement*, CHRON. HIGHER EDUC. (June 7, 2002); Goldie Blumenstyk, *Value of University Licenses on Patents Exceeded $1 Billion in 2000, Survey Finds*, CHRON. HIGHER EDUC. (Mar. 5, 2002), *at* http://www.chronicle.com/daily/2002/03/2002030502n.htm (on file with the Texas Wesleyan Law Review); *MIT alleges patent violation*, SEATTLE TIMES, JAN. 5, 2002, at C4.

26. *See* State St. Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368, 47 USPQ 1596 (Fed. Cir. 1998); *In re* Alappat, 33 F.3d 1526, 1540–45, 31 USPQ2d 1545, 1554–58 (Fed. Cir. 1994); *Examination Guidelines for Computer-Related Inventions*, U.S. Patent and Trademark Office (1996); *see also* Amazon.com, Inc. v. Barnesandnoble.com, Inc., 239 F.3d 1243, 57 USPQ2d 1747 (Fed. Cir. 2001) (demonstrating the limits of patent protection).

27. *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1528–29, 24 USPQ2d 1561, 1574–75 (9th Cir. 1993).

28. *See* Robert W. Gomulkiewicz & Mary L. Williamson, *A Brief Defense of Mass Market Software License Agreements*, 22 RUTGERS COMPUTER & TECH. L.J. 335, 338–41 (1996); Robert W. Gomulkiewicz, *The License Is the Product: Comments on the Promise of Article 2B for Software and Information Licensing*, 13 BERKELEY TECH. L.J. 891, 895–99 (1998).

29. *See* Gary W. Hamilton & Jeffrey C. Hood, *The Shrink-Wrap License—Is it Really Necessary?*, COMPUTER LAW., Aug. 1993, at 16.

30. *See, e.g.*, Mark A. Lemley, *Intellectual Property and Shrinkwrap Licenses*, 68 S. CAL. L. REV. 1239 (1995).

31. *See, e.g.*, Step-Saver Data Sys. v. Wyse Tech, 939 F.2d 91, 95–106 (3d Cir. 1991); Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 7 USPQ2d 1281 (5th Cir. 1988). *But see* Ariz. Retail Sys. v. Software Link, Inc., 831 F. Supp. 759 (D. Ariz. 1993) (upholding the enforceability of a mass market license in the initial transaction

these shadows of doubt, however, software publishers continued to use mass-market licenses, and in the mid-1990's, courts began to enforce them regularly.[32]

V. LEGAL PROTECTION FOR SOFTWARE: TODAY'S ISSUES

Software developers look at the legal protection that is now available for software and find that it falls short in many ways. It has become clearer than ever that having copyright laws on the books is not enough—enforcing the laws is just as important. Software publishers organized many years ago to bring civil actions against software pirates, and those efforts continue. Recently, the federal government has become active in prosecuting software pirates on criminal charges and confiscating pirated software at the border.[33]

Nonetheless, software developers continue to lose billions of dollars annually from piracy.[34] The deterrent effect of the law and the gains from enforcement campaigns are insufficient to curtail software piracy. The success of the copyright holders in *A&M Records, Inc. v. Napster, Inc.*, [35] should have given software developers comfort that the system is working to vindicate copyrights. Instead, the pervasive verbatim copying reinforced the perception of many software publishers that simply owning a copyright in software is not enough.

To back up legal protection, therefore, many software developers are deploying technological devices to prevent unauthorized use. There are many such devices, but examples include devices that meter how many simultaneous copies are in use, digital rights management code, product activation code, and copy protection. To make sure that clever programmers have a strong disincentive to disable or help

---

between a software publisher and a value-added reseller, but finding it unenforceable in a subsequent transaction).

32. *See, e.g.*, ProCD v. Zeidenberg, 86 F.3d 1447, 39 USPQ2d 1161 (7th Cir. 1996); Storm Impact, Inc. v. Software of the Month Club, 13 F. Supp. 2d 782, 48 USPQ2d 1266 (N.D. Ill. 1998); Hotmail Corp. v. Van$ Money Pie, Inc., No. C-98-20064 JW, 1998 WL 388389, 47 USPQ2d 1020 (N.D. Cal. Apr. 16, 1998); M.A. Mortenson Co. v. Timberline Software Corp., 970 P.2d 803 (Wash. Ct. App. 1999), *aff'd*, 998 P.2d 305 (Wash. 2000) (en banc).

33. *See 27 Arrested in Probe of Software Pirating*, SEATTLE TIMES, April 20, 2002, at C1.

34. According to the Business Software Alliance (BSA), in the United States alone, the software industry lost over $2.6 billion in revenue due to piracy. BSA estimates that piracy also resulted in the loss of 107,000 jobs, $5.3 billion in lost wages, and $1.8 billion in lost tax revenues in the United States. Business Software Alliance, *Software Piracy and the Law*, *at* http://www.bsa.org/usa/freetools/consumers/swandlaw_c.phtml (last visited Mar. 12, 2002) (on file with the Texas Wesleyan Law Review); *see also* Dan Carnevale, *Software Piracy Seems Rampant Among Students at 2 Universities*, CHRON. HIGHER EDUC., Mar. 4, 2002, *at* http://www.chronical.com/daily/2002/03/2002030401t.htm (on file with the Texas Wesleyan Law Review).

35. 284 F.3d 1091, 62 USPQ2d 1221 (9th Cir. 2002).

others to disable these devices, Congress passed legislation making it illegal to circumvent or help circumvent the devices.[36]

Anti-circumvention legislation may help solve the piracy problem for software publishers, but other problems have proven to be just as significant. Many software companies seek to sell services that are produced by software rather than license the software on disks. To provide these services, software companies often need to obtain and retain a customer's personal information in a secure manner; to make the services valuable to the customer, the services must be reliable. Consequently, companies that wish to sell software as services must be scrupulous about security and reliability.[37]

However, at a time when security and reliability have become increasingly important, computer programmers have been using their craft to break into computer networks, either to steal personal information or just to cause chaos. In this environment, software publishers turned for help to the Computer Fraud and Abuse legislation,[38] which makes it illegal to break into computer systems without authorization.[39]

Commercial software developers cheered these additional legal protections for software. Other constituencies had a different reaction—they became concerned.[40] For example, librarians worried that anti-circumvention legislation would impair library fair use rights and computer scientists protested that the devices might impair research. Software end users chaffed at the technological measures deployed by

36. 17 U.S.C. § 1201(a)–(f) (2000). *See* COPYRIGHT OFFICE REGULATIONS ON EXCEPTIONS TO ANTI-CIRCUMVENTION.

37. *See* Bill Gates, *Trustworthy Computing*, WIREDNEWS *at* http://www.wired.com/news/business/0,1367,49826,00.html (Jan. 17, 2002) (on file with the Texas Wesleyan Law Review).

38. *See generally, e.g.*, Pamela Samuelson, *Intellectual Property and the Digital Economy: Why the Anti-Circumvention Regulations Need to Be Revised*, 14 BERKELEY TECH. L.J. 519, 525–28 (1999); Jason M. Schultz, Comment, *Taking a Bite out of Circumvention: Analyzing 17 U.S.C. § 1201 as a Criminal Law*, 6 MICH. TELECOMM. & TECH. L. REV. 1, 16–29 (2000). *See also* 18 U.S.C. § 1030 (2000); *Record Panel Threatens Researcher With Lawsuit*, N.Y. TIMES, April 14, 2001, at A1; Farhad Manjoo & Michelle Delio, *Adobe Hackers: We're Immune*, WIREDNEWS, *at* http://www.wired.com/news/politics/0,1283,50797,00.html (Mar. 4, 2002) (on file with the Texas Wesleyan Law Review); Michelle Delio, *Skylarov Indictment 'Not Unusual'*, WIREDNEWS (Aug. 29, 2001).

39. 18 U.S.C. § 1030. *See* Michelle Delio, *A Virus Writer Heads to Prison*, WIREDNEWS *at* http://www.wired.com/news/politics/0,1283,52261,00.html (May 3, 2002) (on file with the Texas Wesleyan Law Review); *but see Judge Rules in Favor of Hackers*, REUTERS NEWS (April 18, 2002) *at* http://www.tlc.discovery.com/news/reu/20020415/hacker/_print.html (reporting that Argentina's supreme court has ruled that hacking is not illegal under Argentina law).

40. *See* Declan McCullagh, *Copyright Law Foes Lose Big*, WIREDNEWS, *at* http://www.wired.com/news/politics/0,1283,48726,00.html (Nov. 29, 2001) (on file with the Texas Wesleyan Law Review); Declan McCullagh, *Digital Security Fomenting a Feud*, WIREDNEWS, *at* http://www.wired.com/news/politics/0,1283,50702,00.html (Feb. 27, 2002) (on file with the Texas Wesleyan Law Review).

software companies, claiming that the measures revealed personal information or that they get in the way of legitimate uses. Some scholars looked at the panoply of legal protection for software and began to argue that, in the aggregate, the laws added up to too much protection.

The concern about "too much protection" spilled over into the National Conference of Commissioners on Uniform State Law's efforts to put together a contract code for software and information licensing. The drafters of the Uniform Computer Information Transactions Act (UCITA) heard consumers express concern that standard form licenses unfairly enlarge intellectual property protection for software developers and pare back warranties and consumer protection laws. Large end users worried that UCITA encourages electronic repossession of software. Librarians asserted that standard form licensing threatens library fair use rights. Others argued that UCITA affects the ability to reverse engineer software or to criticize defective software.[41]

It is probably fair to say that the UCITA drafting process became a forum for airing the concerns of those who perceive that the law has gone too far to protect software, whether the particular concerns expressed were relevant to contract law or not. The fact that people continue to voice concerns even as NCCUSL representatives and others point out that UCITA is not the proper place to address them,[42] is arguably evidence of the depth of the concerns.

VI. CONCLUSION

The ongoing debate about whether there is enough or too much legal protection for software echoes back to a concern expressed by copyright scholar Melville Nimmer many years ago during the CONTU deliberations—would current laws, when applied to software, be stretched past the breaking point? Legislatures and courts have striven mightily throughout the 1980's and 1990's to make sure that the fit is right, but their task is still a work in progress.[43] At

41. An organization of diverse interest groups called AFFECT has been formed to express opposition to UCITA. The various concerns of AFFECT's membership are posted on its website, http://www.4cite.org.

42. *See, e.g.*, EXECUTIVE COMM., NAT'L CONFERENCE OF COMM'RS ON UNIF. STATE LAWS, REPORT OF UCITA STANDBY COMMITTEE (2001), *available at* http://www.nccus/org/nccus1/UCITA-2001-comm-fin.htm.; National Conference of Commissioners on Uniform State Laws, *UCITA Summit Held in DC, at* http://www.nccus1.org/nccus1/pressrelease/pr11_9_01.asp (Aug. 3, 2000) (on file with the Texas Wesleyan Law Review); Raymond T. Nimmer, *Breaking Barriers: The Relation Between Contract and Intellectual Property Law*, 13 BERKELEY TECH. L.J. 827 (Fall 1998). *See generally* UCITA ONLINE, *at* http://www.ucitaonline.com (last revised July 3, 2000).

43. *See, e.g.*, McCullagh, *Digital Security Formenting a Feud, supra* note 39; Richard Morgan, *Senate Committee Discusses Curbs on Colleges' 'Unfair Advantage' in Intellectual Property Cases*, CHRON. HIGHER EDUC. (Feb. 28, 2002).

stake is the continued success of one of the most important industries in the United States economy.