LOTUS v. BORLAND VIEWED THROUGH THE LENS OF INTEROPERABILITY

by Jonathan Band[1]

Lotus v. Borland has long been characterized in both the popular and the computer press as a "look and feel" case. Indeed, District Court Judge Keeton, although he criticized the term "look and feel," largely viewed the case as such. His focus on the user interface was completely understandable, given that in the forerunner to *Borland* -- *Lotus v. Paperback* -- the Paperback and Lotus user interfaces shared many features. Further, at the outset of *Lotus v. Borland,* Lotus alleged that Borland had copied a host of user interface features.

By the time the case got to the First Circuit, however, the only similarity at issue was the Lotus 1-2-3 command structure. Free from the baggage of the *Paperback* decision and the early complaint against Borland, the First Circuit correctly perceived that this case did not concern the "look and feel" of the user interface at all, but instead concerned interoperability. And once the First Circuit understood the role of the command structure in achieving interoperability, Lotus had no chance of winning the case. This article examines this shift in emphasis from the District Court to Court of Appeals, and evaluates the significance of the First Circuit's decision for the computer industry.

I. BACKGROUND

In the late 1970s Visicalc developed the first computerized spreadsheet, which ran on Apple II computers. Soon after the introduction of the IBM PC, Lotus Development Corp. released an IBM PC compatible spreadsheet, Lotus 1-2-3. Lotus 1-2-3 quickly dominated the spreadsheet market, eclipsing Visicalc, and its popularity contributed to the success of the IBM PC.

Although Lotus 1-2-3 incorporated some of Visicalc's commands in its user interface, the Lotus product had an original menu tree structure arranging over four hundred commands in a

---

[1] Jonathan Band is a partner in the Washington, D.C. office of Morrison & Foerster.

clearly defined hierarchy. Using different code, Paperback Software developed a program that re-created the entire Lotus 1-2-3 command structure, as well as other features of the Lotus 1-2-3 user interface. Lotus sued for copyright infringement, and prevailed in 1990.[2] Paperback did not have the resources to appeal the judgment.

Another Lotus competitor, Borland International, sensed that it was next in line, so it initiated a declaratory judgment action against Lotus in what it hoped would be a more hospitable forum in California. Lotus then filed suit against Borland in Massachusetts, where it had brought the ***Paperback*** action, and the case was assigned to the same judge who had decided in its favor in ***Paperback***: Judge Robert Keeton, a former Harvard Law School professor. A jurisdictional tussle between the federal district courts in California and Massachusetts ensued, and Lotus succeeded in having the two cases consolidated before Judge Keeton in Massachusetts.

Borland's Quattro Pro could operate in two different modes: a native mode completely different from Lotus 1-2-3, and a 1-2-3 mode which offered the user the same command structure as Lotus 1-2-3.[3] As in ***Paperback***, there was no allegation that Borland copied any Lotus code. Unlike ***Paperback***, however, the Borland user interface, even when in the 1-2-3 mode, looked completely different from Lotus 1-2-3. Despite this absence of visual similarity, Lotus claimed that Borland infringed its copyright by copying the 1-2-3 command structure.

While familiarity with the technical facts is important in every software copyright case, it is particularly important here because of the dual function of the Lotus 1-2-3 command structure. First, the Lotus commands appear on the screen in a logical order to inform the user about the available options at that stage of operation, and the user invokes a sequence of the commands to instruct the program directly to perform certain spreadsheet functions. In this role, the commands act as an interface between the user and the program.

---

[2] Lotus Development Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990). Lotus' victory caused 300 picketers to protest Lotus' litigation strategy outside of the Lotus headquarters in Cambridge. See Michael Alexander, Lotus Litigation Draws Protest, Computer World, Aug. 6, 1990, at 6.

[3] Borland's 1-2-3 mode also offered the user additional commands not found in Lotus 1-2-3.

Second, the spreadsheet user can employ the Lotus commands to write her own program known as a "macro." A macro employs a sequence of menu commands to perform a series of the spreadsheet operations in a particular order.[4] In this role, the command structure acts as an interface between the spreadsheet program and the user-written programs.

Although Lotus 1-2-3, taken as a whole, is an application program, it assumes some of the characteristics of an operating system with respect to the user written application programs -- the macros -- that attach to it. In other words, Lotus 1-2-3 serves as a platform upon which the macros run. The syntax and semantics of the communication between a macro and the Lotus platform are those of the Lotus commands.

Lotus users had invested substantial time and resources developing libraries of customized macros appropriate to their business needs. Because of this investment in the macros, the Lotus users were "locked-in" to the Lotus environment; as they expand their operations, they simply would not purchase spreadsheet programs developed by a Lotus competitor such as Borland unless the Borland spreadsheet could execute their macros.[5]

The most basic form of macro-compatibility required the Borland platform to have the ability to translate the user-written macros instructions into instructions intelligible to the Borland platform, and visa versa. Because the set of instructions used by the macro was a subset of Lotus' commands, the Borland platform had to translate those instructions from the macros by means of a file that replicated the Lotus 1-2-3 command structure, syntax and semantics. Borland called this file the "Key Reader."

II. THE DISTRICT COURT DECISIONS

Judge Keeton primarily considered the Lotus command structure in its user interface role. He found the command structure to be protected expression, relying heavily on the Third

---

[4] Lotus Development Corp. v. Borland Int'l Inc., 799 F. Supp. 203, 206 (D. Mass. 1992).

[5] Neil Gandal, Hedonic Price Indexes for Spreadsheets and an Empirical Test for Network Externalities, 25 Rand. J. Econ. 160 (Spring, 1994).

Circuit's decision in *Whelan v. Jaslow*.[6] Although Judge Keeton rejected *Whelan*'s simplistic one idea per program rule, he nonetheless adopted *Whelan*'s abbreviated upright analysis, where the idea/expression dichotomy was reduced to the merger doctrine. In Judge Keeton's view, if an author had alternatives available to him, there was no merger, and if there was no merger, there was protected expression. Because a programmer theoretically could construct many different spreadsheet command structures, copyright protected the Lotus 1-2-3 command structure. Accordingly, the Borland user interface in the 1-2-3 mode infringed Lotus' copyright.

In one of his four decisions in the case, Judge Keeton specifically addressed the Key Reader and macro-compatibility. Keeton ruled that the Key Reader infringed Lotus' copyright just as the Borland user interface did. Keeton acknowledged that under *Computer Associates v. Altai*,[7] issued just one month prior to his July, 1992, macro-compatibility ruling, "aspects of computer software cannot be subject to copyright if they are greatly circumscribed by the hardware or software with which they are designed to interact."[8] Nonetheless, Keeton concluded that this proposition did not apply to the Key Reader. Judge Keeton reached this result by limiting *Altai*'s rejection of copyright protection for interface specifications to situations where "what the program was designed to fit was already in existence before the program was designed to fit."[9] In other words, whether a program characteristic received protection turned on the constraints existing at the time of that program's creation, and not on the constraints existing at the time of the allegedly infringing program's creation.

Under Judge Keeton's reasoning, a second programmer writing a new application program could copy the interface specifications of an earlier application program designed to run on a pre-existing operating system because these interface specifications had been constrained by the operating system and thus did not receive copyright protection. A second programmer

---

6 797 F.2d 1222 (3rd Cir. 1986), cert. denied, 479 U.S. 1031 (1987).

7 982 F.2d 693 (2nd Cir. 1992).

8 799 F. Supp. at 212.

9 Id. at 213. The similarities between the Computer Associates and Altai programs resulted in large part from constraints imposed by the pre-existing IBM operating systems with which they interoperated.

writing a new operating system, however, could not copy the pre-existing operating system's interface specifications to permit the new operating system to run the two application programs. This is because the interface specifications of the first operating system were not constrained at the time of their creation, and thus *did* receive copyright protection. This result would effectively eliminate competition in operating systems or any software product that functions as a platform for other software products. Judge Keeton, therefore, interpreted *Altai* as permitting only the development of products that attached to, but did not compete with, a pre-existing platform. The interoperable developer could write programs to run on the opposite side of the interface from the pre-existing platform, but not on the same side.

Judge Keeton in the *Paperback* decision articulated an interesting policy rationale that appears to underlie this outcome. Paperback, like Borland, had raised the macro-compatibility issue. Paperback argued that because the Lotus command structure had become a *de facto* standard among spreadsheet users, Paperback should be free to copy it. Judge Keeton responded by attributing Lotus' *de facto* standard status to the creativity of the Lotus programmers. In his view, it would be perverse for the command structure somehow to lose its copyright protection simply because the Lotus programmers had done such a good job that they established a *de facto* standard that now constrained competitors.[10] Had the Lotus programmers been less creative, and had 1-2-3 been less successful, it would not have become a *de facto* standard and Paperback would not seek to use it. Copyright should reward creativity, he argued, not penalize it.

Judge Keeton's narrow reading of *Altai* flows directly from this analysis. From his perspective, it would be perverse to allow Borland to copy the Lotus command structure in order to compete with Lotus 1-2-3; Lotus should not be disadvantaged by the fact that its innovative product became so popular. This reasoning reveals that, at bottom, Judge Keeton still viewed the case through "look and feel," rather than interoperability, lenses. That is, he viewed Borland as trying to free ride on a popular product developed by Lotus, much as a jeans or tennis shoe company tries to sell products that look like those of the market leader. He saw only two parties

[10] 740 F.Supp. at 79.

in the copyright calculus: Lotus and Borland. Almost completely absent from his analysis were users, without question the most important party from the perspective of the U.S. copyright system. Had he looked at the case through interoperability lenses, he would not have committed this egregious error.

The scant attention Judge Keeton gave users is visible in three distinct ways. First, he failed to take the user macros seriously. In the decision he used trivial examples of macros involving only a handful of steps. In the business world, however, macros can have thousands of steps, and can represent significant user investment. Second, his decision gave no meaningful weight to the problem of user lock-in. Because of their significant investment in the macros, users were not willing to switch to a rival platform, even if it were technologically superior. Third, Judge Keeton did not acknowledge the user contribution to the *de facto* standardization of the Lotus command structure. 1-2-3 became the standard in large measure because of the significant user investment in macros.[11] To be sure, Lotus 1-2-3 contained many innovative features when it hit the market, but its success is at least partly attributable to many other factors, most notably user development of macros dependent on 1-2-3.

While Judge Keeton did not fully appreciate the interoperability dimension of the case, the First Circuit did. The First Circuit was aided by no less than four *amicus* briefs stressing interoperability, filed by: the Software Entrepreneurs Forum, an organization of more than 1000 independent software developers, consultants, and providers; a group of twenty-seven leading computer scientists; a collection of twenty personal computer user groups; and the American Committee for Interoperable Systems, whose thirty members include Sun Microsystems, Storage Technology, Bull HN, Amdahl and Broderbund. Judge Keeton did not have the benefits of these briefs.

III. THE FIRST CIRCUIT'S OPINION

[11] See Federick R. Warren-Boulton, Kenneth C. Baseman & Glenn A. Woroch, Copyright Protection For Software Can Make Economic Sense, 12 Computer Lawyer, Feb. 1995.

On March 9, 1995, the First Circuit reversed Judge Keeton. The opinion of the Court, written by Judge Stahl, focused on interoperability from the outset. Judge Stahl described the macro-compatibility function of the command structure in the third paragraph of the opinion. In the fifth paragraph of the opinion, he addressed Borland's motivation for copying the 1-2-3 command structure in interoperability terms: "Borland included the Lotus command hierarchy in its programs to make them *compatible* with Lotus 1-2-3 so that spreadsheet users who were clearly familiar with Lotus 1-2-3 would be able to switch to the Borland programs without having to learn new commands or rewrite their Lotus macros."[12]

The Court defined the question before it very narrowly: "whether a computer menu command hierarchy constitutes copyrightable subject matter." It then opined that this was a question of first impression. With no hesitation, the Court proceeded to conclude that the Lotus command structure was a method of operation unprotected under Section 102(b) of the Copyright Act. The Court explained that:

> The Lotus menu command hierarchy provides the means by which users control and operate Lotus 1-2-3 . . . . Users must use the command terms to tell the computer what to do. Without the menu command hierarchy, users would not be able to access and control, or indeed make use of, Lotus 1-2-3's functional capabilities.[13]

The First Circuit rejected the District Court's argument that copyright protected the Lotus command structure because Borland could -- and did -- develop its own command structure using different terms and a different arrangement. "If specific words are essential to operating something," the First Circuit stated, "then they are part of a `method of operation' and, as such, are unprotectable."[14] This is true even if a parallel set of words could operate a parallel program.

---

12 Lotus v. Borland, 49 F.3d 807, 810 (1st Cir. 1995) (emphasis supplied).
13 Id. at 815.
14 Id. at 816.

In other words, the mere fact that a programmer could construct other methods of operation in no way renders a given method of operation protected expression.

The Court then turned to the underlying theme of interoperability:

That the Lotus menu command hierarchy is a "method of operation" becomes clearer when one considers program compatibility. Under Lotus's theory, if a user uses several different programs, he or she must learn how to perform the same operations in a different way for each program used . . . . We find this absurd. The fact that there may be many different ways to operate a computer program, or even many different ways to operate a computer program using a set of hierarchically arranged command terms, does not make the actual method of operation chosen copyrightable; it still functions as a method for operating the computer and as such is uncopyrightable.[15]

The Court amplified this theme in the macro-compatibility context:

Under the district court's holding, if the user wrote a macro to shorten the time needed to perform a certain operation in Lotus 1-2-3, the user would be unable to use that macro to shorten the time needed to perform that same operation in another program. Rather, the user would have to rewrite his or her macros using that other program's menu command hierarchy. This is despite the fact that the macro is clearly the user's own work product. We think that forcing the user to cause the computer to perform the same operation in a different way ignores Congress's direction in Section 102(b) that "methods of operation" are not copyrightable.[16]

The opinion closed by proclaiming its consistency with ***Feist v. Rural Telephone***,[17] and by paraphrasing the policy embodied by ***Feist***:

---

15 Id. at 817-18.

16 Id. at 818.

17 499 U.S. 340 (1991).

We also note that in most contexts, there is no need to "build" upon other people's expression, for the ideas conveyed by that expression can be conveyed by someone else without copying the first author's expression. In the context of methods of operation, however, "building" requires the use of the precise method of operation already employed; otherwise, "building" would require dismantling, too. Original developers are not the only people entitled to build on the methods of operation they create; anyone can. Thus, Borland may build on the method of operation that Lotus designed and may use the Lotus menu command hierarchy in doing so.[18]

IV. JUDGE BOUDIN'S CONCURRING OPINION

Judge Boudin[19] wrote a concurring opinion that delved even deeper into the economic and policy ramifications arising from the interoperability dimension of the case. In his opinion, Judge Boudin noted that most of the law of copyright had developed in the context of literary works such as novels, plays, and films, and that the "problem presented by computer programs is fundamentally different"[20] because of their functional and utilitarian nature. Judge Boudin explained that because of this utilitarian nature, the danger of over-protection is greater than in the case of traditional literary works. "[A] 'mistake' in providing too much protection [for traditional works] involves a small cost: subsequent authors treating the same themes must take a few more steps away from the original expression."[21] But in the case of computer programs, the improper grant of copyright protection "can have some of the consequences of patent protection in limiting other people's ability to perform a task in the most efficient manner."[22]

Judge Boudin then applied these principles to the instant case:

---

[18] Lotus at 818 (footnote deleted).

[19] Judge Boudin replaced Judge Breyer on the panel after Breyer was elevated to the Supreme Court.

[20] Lotus at 819.

[21] Id. at 819.

[22] Id.

Requests for the protection of computer menus present the concern with fencing off access to the commons in an acute form. A new menu may be a creative work, but over time its importance may come to reside more in the investment that has been made by users in learning the menu and in building their own mini-programs -- macros -- in reliance on the menu.[23] If Lotus could obtain a monopoly in the 1-2-3 command structure, users who have learned the command structure of Lotus 1-2-3 or devised their own macros are locked into Lotus . . . . Apparently, for a period Lotus 1-2-3 has had such sway in the market that it has represented the de facto standard for electronic spreadsheet commands. So long as Lotus is the superior spreadsheet -- either in quality or in price -- there may be nothing wrong with this advantage. But if a better spreadsheet comes along, it is hard to see why customers who have learned the Lotus menu and devised macros for it should remain captives of Lotus because of an investment in learning made by the users and not by Lotus. Lotus has already reaped a substantial reward for being first; assuming that the Borland program is now better, good reasons exist for freeing it to attract old Lotus customers: to enable the old customers to take advantage of a new advance . ..[24]

Given the obvious benefits to the user resulting from withholding copyright protection from program elements necessary for interoperability, the question for Judge Boudin "is not whether Borland should prevail but on what basis."[25] Judge Boudin identified two possible bases: first, the basis adopted by the Court, viewing the command structure as an unprotected method of operation; and second, permitting Borland's use under the fair use doctrine.

Judge Boudin explored the fair use concept further. Borland's use here would be privileged because it is not seeking to appropriate the advances made by Lotus' menu; rather, having provided an arguably more attractive menu of its own, Borland is merely trying to give

---

23 Id.

24 Id. at 821.

25 Id.

former Lotus users an option to exploit their own prior investment in learning or in macros.[26] Judge Boudin stressed that the privilege would only be available because Borland was providing the user with additional functionality; had it simply copied the menu using different code, but contributed nothing significant of its own, Judge Boudin would not have sanctioned the use.

Judge Boudin conceded that Borland's use may not fit squarely within the fair use doctrine under its current formulation, but added that "the doctrine of fair use was created by the courts and can be adapted to new purposes."[27] Judge Boudin further acknowledged that widespread application of the fair use doctrine in such circumstances, *i.e.*, for purposes of achieving interoperability, "would entail a host of administrative problems that would cause cost and delay, and would also reduce the ability of the industry to predict outcomes."[28]

Judge Boudin's reasoning is unabashedly results-oriented. The fair use doctrine, however, encourages precisely such reasoning. As the Supreme Court stated in *Stewart v. Abend*, the fair use doctrine is an "equitable rule of reason which permits courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which the law is designed to foster."[29] Judge Boudin recognized that preventing interoperability invariably stifles creativity; new firms cannot introduce new products for the locked-in base, and the monopolist has little incentive to innovate -- the ultimate goal of the intellectual property system. Interoperability, conversely, permits competition, which stimulates innovation -- the ultimate good of the intellectual property system.

V. THE TREATMENT OF *ALTAI*

---

[26] Id.
[27] Id.
[28] Id. at 821-22.
[29] 495 U.S. 207, 237 (1990) (citations omitted).

Much of the briefing before the First Circuit centered on how the Second Circuit's decision in *Altai* should be applied to the facts in this case. Interestingly, Judge Stahl in a section devoted entirely to *Altai* concluded that the *Altai* abstraction-filtration-comparison test was "of little help." Judge Stahl explained that the *Altai* test may be useful in cases involving non-literal copying, but that this case involved "Borland's deliberate, literal copying of the Lotus menu command hierarchy."[30]

Judge Stahl further stated that the *Altai* test in this context may actually be "misleading" because it could cause the identification of expression at a low level of abstraction while obscuring the fact that the expression may be part of a method of operation at a higher level of abstraction. In other words, Judge Stahl flips the standard critique of the *Altai* test on its head. IBM's Tony Clapes and others have argued that the *Altai* test could lead to the "atomization" of a computer program, thereby obscuring expression in the selection, coordination, and arrangement of non-protectable elements.[31] For Tony Clapes, atomization leads to under-protection; for Judge Stahl, it leads to over-protection!

To be sure, the sloppy application of the *Altai* test can lead to either over- or under-protection. Although *Altai* and its progeny, notably *Gates v. Bando*,[32]have crafted a more structured legal test than most, the test still is only as good as the judge applying it. Thus, a careful administration of the *Altai* test to the facts of this case would not mislead a court.

Indeed, one could argue that even though Judge Stahl stated that *Altai* was of little help, both he and Judge Boudin actually applied the *Altai* test successfully. Because the case involved literal copying, the First Circuit could skip the abstraction step altogether and focus its energy on filtration: determining whether copyright protected the element Borland had copied word for

---

30 Lotus at 814.

31 Anthony L. Clapes & Jennifer M. Daniels, Revenge of the Luddites: A Closer Look at Computer Associates v. Altai, 9 Computer Lawyer, Nov. 1992, at 4.

32 9 F.3d 823 (10th Cir. 1993).

word, the Lotus command structure. In performing the filtration step, the First Circuit appears to have been informed by the teachings of *Altai* and its progeny on the utilitarian nature of computer programs. It appears to have been similarly informed by these cases that copyright should not be applied in a manner that hinders interoperability.[33] Accordingly, even though it seems to distance itself from *Altai*, the First Circuit's decision falls squarely within the *Altai* tradition.

VI. CONCLUSION

Although the *Altai* decision held that program elements constrained by interoperability requirements -- interface specifications -- could not receive copyright protection, Judge Keeton's Key Reader decision limited that holding to the specifications necessary to attach to, but not compete with, the pre-existing platform. Because of the large concentration of software development activity in Massachusetts and because of Judge Keeton's scholarly reputation, the Key Reader decision had the potential to significantly retard the development of interoperable software products. The First Circuit's reversal of Judge Keeton lifted the cloud of uncertainty hanging over the community of interoperable developers and their many customers. The First Circuit unambiguously ruled that program elements necessary to achieve interoperability -- to attach as well as to compete -- by definition were methods of operation unprotected by copyright. Judge Boudin further suggested that to the extent such elements were not methods of operation, their use should be permitted under the fair use doctrine.

The First Circuit also expanded the concept of interoperability to include not only the ability of different products to work together, but also the ability of a user to employ the same skill set with different products. The Court essentially reasoned that a method of operation is a method of operation, regardless of whether it is performed by a person or a program.

[33] See, e.g., Atari v. Nintendo, 975 F.2d 832 (Fed. Cir. 1992); Sega v. Accolade, 977 F.2d 1510 (9th Cir. 1992).

Some software companies surely will complain that withholding copyright protection from program elements necessary for interoperability defined this broadly will diminish their incentive to innovate. Notwithstanding these protests, many program elements remain subject to copyright protection. The First Circuit specifically stated that the screen displays apart from the command structure, and the computer code implementing the command structure in both its user interface and macro-compatibility roles, could still receive copyright protection.[34] This combination of proprietary and non-proprietary elements meets the copyright imperative of balancing the need to provide incentives to the developers of new technologies with the need to permit competition through interoperability.

Lotus has announced that it will seek Supreme Court review of the First Circuit's decision. The First Circuit cleverly diminished the likelihood that the Supreme Court would grant *certiorari* by explicitly *not* relying on *Altai* and thereby side-stepping the *Altai*-*Whelan* scope of protection debate. Indeed, the First Circuit framed its decision in such a manner that it is in conflict only with a footnote in the Tenth Circuit's *Autoskill v. NESS*.[35] In that footnote, the Tenth Circuit mistakenly suggested that copyright might protect a programmer's decision to have students select a response to a query by pressing the 1, 2, or 3 keys. It is highly unlikely that the Supreme Court will view the divergence between the First and Tenth Circuits on this issue as a split in the circuits worthy of Supreme Court resolution.

---

[34] Lotus at 815-16.

[35] 994 F.2d 1476, 1495 n.23 (10th Cir.), cert. denied, 114 S. Ct. 307 (1993).