Issues and Opinions

Logic Programming as a Paradigm for Financial Modeling: Response to a Comment by McLintock and Berry

By: Robert P. Minch
Computer Information Systems and Production Management
College of Business
Boise State University
1910 University Drive
Boise, Idaho 83725 U.S.A.

Introduction

I wish to thank A.J. McLintock and R.H. Berry of the University of East Anglia for their comment to my March 1989 *MIS Quarterly* article "Logic Programming as a Paradigm for Financial Modeling" (Minch, 1989). It is my hope that their comment and this response together will add a lively addendum to my previously published research.

The gist of McLintock and Berry's argument seems to be that the benefits of advanced financial modeling functions do not outweigh their costs for users, who don't want or need the functionality I propose in the article. However, they address a much narrower set of financial modeling software users and tasks than I did. They focus on non-specialists building simple financial models using spreadsheet software. I addressed managers and financial planners building, analyzing, and interfacing models of varying complexity, using spreadsheets, modeling languages, and the proposed logic programming-based tools. They generalize from a survey of some 200 professional accounting offices in the U.K. and interviews with those firms' financial modelers, who are not exposed to operational functions of the types proposed in my article. In their study, the overwhelming majority of modelers within the firms were "not specialists in computers or model building but were qualified accountants." Modelers were virtually all self-taught aside from some short in-house training, used only the most basic facilities of the spreadsheet software (largely ignoring even graphics), and mostly built "small, crude, superficial, simplistic, and issue-specific" models. This is a much different group of users than the typical business school graduate I envision as potential users of my proposed system, at least in the United States, where many if not most students are exposed to spreadsheet software and financial modeling software such as IFPS (Frand and Britt, 1991), and receive formal education in the appropriate use of such tools (Frand, et al., 1990).

To support their contention that the benefits of more advanced modeling facilities are few, McLintock and Berry review the six limitations of current financial planning software outlined in my article. For each limitation, they either ask the sampled accountant users about the increased functionality I proposed or judge its usefulness themselves, presumably from comments made by those users or general observation. They conclude that these users make "quick and dirty" estimates of relationships, do what-if analysis merely by "running the program several times with different parameters," "only understand numbers," have no problem with model storage space because their models are small, don't think meta-level features such as spreadsheet auditing functions are "worth considering," do database querying "simply by looking at the spreadsheet output in the relevant columns or rows," and apparently do very little systematic model validation and only some verification (the lack of adequate validation and verification acknowledged as "distressing" by McLintock and Berry). To justify the lack of validation and verification as not a "practical problem," they propose that validating the modeler rather than the models, applying a black box reasonableness test, or even merely "eyeballing the output" is adequate. Given the relatively low level of user motivation, user sophistication, and task complexity apparent in this context, it is not surprising that more advanced features were not greeted with enthusiasm.

McLintock and Berry suggest that the cost of obtaining more advanced modeling features is high and that "learning a completely new programming language is a drastic solution." I quite agree that this *would* be a drastic solution but an unnecessary one. In my article I clearly state: "It would be unreasonable to ask (and is not in this article) that managers and other users of financial planning software become familiar with a logic programming language such as Prolog in its 'raw' form" (Minch, 1989, p. 76). I then go on to describe how meta-level interpreters can be used to create spreadsheet-like or other desired user interfaces to insulate users from implementation details (just as current spreadsheet systems insulate users from their underlying assembler and C code). Thus, the cost to users of obtaining advanced capabilities is actually quite low, and, of course, the learning of new features would typically be at the option of the users themselves.

In light of the very different perspectives and conclusions reached by McLintock and Berry and myself, perhaps the most effective way to respond to their comment is to introduce new evidence and discussion to shed light on two basic questions: (1) Are McLintock and Berry right when they make the generalization that financial planners don't want and don't need the added modeling functionality and power I proposed in my article? and (2) Even if they were right, should software developers stop adding modeling functionality and power to existing and future systems?

Are McLintock and Berry Right?

To support their comment, McLintock and Berry refer to their own empirical research. My article is conceptual in nature—no attempt was made to conduct empirical research. Nevertheless, I'd like to examine a small portion of evidence from other published sources that will shed some light on the question: Are McLintock and Berry right when they make the generalization that financial planners don't want and don't need the added modeling functionality and power I proposed in my article?

Before drawing conclusions about user *wants* based on McLintock and Berry's sample results, we might want to know if their results are consistent with the findings of other researchers. An example of another empirical research effort that examined the use of advanced financial modeling functions in the context of spreadsheet users might lend some much-needed perspective. Fordyce (1987) examined the actual use of implemented, advanced spreadsheet modeling enhancements. These included equation ordering (allowing statements to be entered in nonprocedural order), formula reversibility (to calculate all possible unknowns from known values), calculation explanation, variable breakdown (showing which variables affect other variables), and plain English queries against the model data. Any reader comparing these functions with those I proposed in my article will find them similar and highly complementary. To evaluate the enhancements, Fordyce examined user satisfaction, amount of use for the enhancements, and user-perceived value of the enhancements, using 20 subjects who were business professionals enrolled in MBA and MPA programs. Unlike McLintock and Berry's comment, Fordyce included details of his methodology and his complete survey instrument. His subjects used all the additional enhancements or functions noted in my article, with the exception of the natural language query of worksheet databases (the databases were quite small). Furthermore, most of the enhanced functions, including calculation explanation, variable analysis, and equation ordering, were judged to be valuable additions by the users (Fordyce, 1987, p. 267). These results are clearly inconsistent with those presented by McLintock and Berry's comment and suggest considerable caution in adopting their conclusions.

A second way to examine user *wants* is to observe what has happened in the spreadsheet and financial planning software market over the past few years. If we do this, we find clear evidence that more advanced features in fact *are* being incorporated into the software and are being demanded and purchased by users. Add-in products for spreadsheet software have established a significant market niche, providing users with additional functions such as more sophisticated equation solvers (Ferranti, 1990), meta-level consistency checkers, spreadsheet auditors, cell dependency tracers, circular reference identifiers (Ditlea, 1987), and database

access facilities (McMullen, 1991). Market-leading microcomputer spreadsheets Lotus 1-2-3/W and Microsoft Excel now include Assumption Manager and Scenario Manager components, allowing users to run and solve different ranges of numbers for use in what-if analyses (Low, 1992). More sophisticated financial modeling languages such as IFPS have for several years successfully marketed what-if, goal-seeking, scenario management, and other advanced capabilities (Execucom, 1989). With each new release of spreadsheet software, vendors respond to users' desires to build larger and more complex models. Again we see evidence that should cause us to question McLintock and Berry's conclusions.

Examining the *need* for advanced financial modeling features is difficult. Unfortunately, many modelers not wanting more sophisticated software support in fact *do* need additional support but fail to recognize that need. Let us consider just one example: the problem of errors in user-built spreadsheet models. In a widely reported case, the controller of a Florida construction company made a simple formula range error costing his company $254,000 (the company filed suit against the software publisher, Lotus Development Corp. (*Wall Street Journal,* 1986a); the suit was later dropped (*Wall Street Journal,* 1986b) in what was seen to be an acknowledgement of a user error). A trade journal reported spreadsheet errors in one out of three business spreadsheets (Ditlea, 1987). Perhaps most convincing, a rigorous and thoroughly described study found that even experienced spreadsheet users had at least one non-trivial error in 44 percent of the spreadsheets created (Brown and Gould, 1987). The most common errors were serious in nature (such as formula construction errors) rather than less serious problems such as rounding errors, and users were just as confident of the accuracy of the spreadsheets containing errors as those that did not. Brown and Gould recommended improved representation of formulae and debugging tools (such as those proposed in my article) as ways to address the problem of errors. Unfortunately, McLintock and Berry apparently did not study (or at least did not report in their comment) the error rates of their spreadsheet users. Once again caution is advisable in evaluating their conclusions.

What If They Were Right?

Even if McLintock and Berry were right, should software developers stop adding modeling functionality and power to existing and future systems?

The evidence presented above seems to refute their claim that users do not want or need additional advanced modeling facilities. Nevertheless, as an exercise in closure, let us consider what we should advocate if their conclusions were correct. This requires a more philosophical and normative approach rather than mere descriptive empiricism.

My initial reading of McLintock and Berry's comment brought back memories of a seminar I participated in some years ago. About half the participants believed the proper way to develop systems is to carefully survey user needs and then build a system that attempts to meet those needs in the way the users want them met. The other half felt that while this is perhaps appropriate some of the time, system developers should also be encouraged to satisfy users by creatively redefining the problem or restructuring the solution through software innovation (not unlike the currently popular "reengineering through information technology" techniques). The former approach seems to be the one taken by McLintock and Berry, while I support the latter. I pointed out that users are sometimes unaware of their needs, and even wants, and are sometimes ignorant of the best ways to meet those wants and needs. Were the first immensely popular computer video games designed by relying on surveys and focus groups? Are the highly successful microcomputer graphical user interfaces solely the result of meticulous solicitation of user needs? The answer to both questions is no: many implemented systems are the result of a *co-evolution* involving the meeting of user wants and needs as well as creative approaches to the problem, its context, and environment. Successful systems often leapfrog expressed user needs to innovatively offer richer support for those users willing to experiment with and adopt new technologies. Software should not be judged valueless if it fails to receive total acceptance by every type of user in every context. Furthermore, it is well known than many users of sophisticated software systems choose to use only a fraction of the total available feature set and yet are very satisfied.

After reading McLintock and Berry's comment and before writing this reply, I had the pleasure of attending a very interesting panel discussion on product innovation at the January 1992 HICSS conference.[1] The session was lead by Bob Johansen of the Institute for the Future and included many well-known academics and practitioners in information systems. One of the unique characteristics of the session was that it included the ability for audience members to respond in real time to issues under discussion through electronic polling/feedback devices. Interestingly, the following two informally worded propositions were put to opinion votes: "The best way to come up with technological innovations is to study user needs" and "Innovation is mostly driven by technology; then we figure out what to do with it." To the first proposition, 59 of 99 voting participants disagreed with the statement and 40 agreed, while to the second 68 of 100 agreed with the statement and 32 disagreed. This provides evidence that many practitioners and researchers share my perspective that technological innovation is more than a simple reaction to expressed current user needs.

Conclusions

McLintock and Berry respond to a previously published conceptual article proposing advanced financial modeling functions by citing results from their own empirical study. They neither provide details of their research methods nor indicate that their study has been reported more fully elsewhere. They conclude that their sample of admittedly unsophisticated spreadsheet model builders would benefit little from the advanced functions proposed in the article and appear to suggest that this is the case in general. I have attempted to show that we should question their conclusions about the wants and needs of financial modelers because:

- Other published and more completely described studies examining the actual hands-on use of similar advanced financial modeling functions found those functions were not only used but were perceived as valuable by users;
- Spreadsheet and financial modeling software vendors continue to add additional advanced features (many similar to those I proposed), and users continue to demand, purchase, and use the enhanced software; and
- Users often need advanced features such as spreadsheet auditing tools without realizing the need, as is evidenced by the alarmingly high spreadsheet error rates reported in the literature.

In addition, I have suggested that creative advancements in software tools will be needlessly stunted if enhancements must be limited to only those explicitly proposed by users or judged in advance by prospective users without hands-on use. We also should encourage the creation of innovative tools that may prove very valuable after users come to appreciate them and adapt them to their particular tasks.

In the future, I hope we can take an optimistic and supporting view of both users and developers by promoting a co-evolutionary process of fulfilling user requirements and advancing software technology. In this way, users will have the opportunity to adopt and adapt those tools that best meet their perceived and real needs. We must not lose sight of our long-term goal to maximize support for many types of users by becoming too concerned with meeting short-term minimum needs for one type of user.

[1] "Open Forum on Practical Product Innovation," coordinated by R. Johansen, held January 9, 1992 at the Twenty-Fifth Annual Hawaii International Conference on Systems Sciences, Koloa, Hawaii.

References

Brown, P.S. and Gould, J.D. "An Experimental Study of People Creating Spreadsheets," *ACM Transactions on Office Information Systems* (5:3), July 1987, pp. 258-272.

Ditlea, S. "Spreadsheets Can be Hazardous to your Health," *Personal Computing* (11:1), January 1987, p. 60.

Execucom Systems Corporation. *IFPS/Plus User's Manual,* Executive Systems Corporation, Austin, Texas, 1989.

Ferranti, M. "Lotus 1-2-3 3.1 Add-In Crop Grows," *PC Week,* Dec. 10, 1990, p. 33.

Fordyce, K. "Looking at Worksheet Modeling through Expert System Eyes," in *Expert Systems for Business,* B. Silverman (ed.), Addison-Wesley, Reading, MA, 1987.

Frand, J., Roldan, M., and Sutcliffe, N. (eds.). *Syllabus Book: IBM Management of Information Systems 1985-1990 Grant Program,* Anderson Graduate School of Management, University of California, Los Angeles, CA, February 1990.

Frand, J. and Britt, J. *Eighth Annual UCLA Survey of Business School Computer Usage,* Anderson Graduate School of Management, University of California, Los Angeles, CA, September 1991.

Low, L. "Excel 4.0 Looks Like Lotus, Borland Spreadsheets," *Infoworld,* December 30, 1991/January 6, 1992, p. 8.

Minch, R.P. "Logic Programming as a Paradigm for Financial Modeling," *MIS Quarterly* (13:1), March 1989, pp. 65-84.

McMullen, J. "Making the Spreadsheet Connection," *Datamation* (37:1), January 1, 1991, pp. 57-58.

Wall Street Journal. "Firm Sues Lotus, Claiming Software Fault Led to Loss," July 29, 1986a, p. 8.

Wall Street Journal. "Lotus Says Firm Dropped Claims of Software Defect," December 10, 1986b, p. 3.