Western New England Law Review

Volume 7 *7 (1984-1985)*
Issue 3

Article 4

1-1-1985

SOFTWARE DEVELOPMENT CONTRACTS AND CONSULTING ARRANGEMENTS: A STRUCTURE FOR ENFORCEABILITY AND PRACTICALITY

Mark L. Gordon

Steven B. Starr

SOFTWARE DEVELOPMENT CONTRACTS AND CONSULTING ARRANGEMENTS: A STRUCTURE FOR ENFORCEABILITY AND PRACTICALITY†

MARK L. GORDON*
STEVEN B. STARR**

TABLE OF CONTENTS

---

* Partner, Chicago law firm of Gordon & Glickson, P.C. Mr. Gordon received a B.A. (with distinction) in 1973 from the University of Michigan, and a J.D. (cum laude) from Northwestern University School of Law in 1976. Mr. Gordon is currently a Director of the Computer Law Association.

** Associate, Chicago law firm of Gordon & Glickson, P.C. Mr. Starr received a B.A. in Economics and Philosophy from Vanderbilt in 1975, and a J.D. from Vanderbilt in 1978.

I. INTRODUCTION

Software development contracts and consulting agreements are perhaps the most difficult of all technology related contracts to draft and effectively implement.[1] In the case of a consulting agreement, the results for which the user contracts may merely be recommendations or ideas which are difficult to quantify in advance. In a software development contract, the end result (a software program) may be defined in advance; however, the process of detailing the definition is usually part of the contract, thus creating a "catch 22."[2]

While users frequently attempt to contract for results in software development contracts, vendors typically attempt to limit themselves to contracts for the provision of services.[3] It is perhaps this tension

1. This is due primarily to the nature of the transaction undertaken. In software development contracts and consulting arrangements, the parties contract for an unknown or relatively undefined product. Hoffman, *Software Development and Service Agreements*, 24 JURIMETRICS 58,58-9 (1983). For more in depth treatment of associated problems, *see* Harris, *Complex Contractual Issues in the Aquisition of Hardware and Software*, 4 COMPUTER/L.J. 77-100 (1983).

2. This is a "catch 22" because one cannot detail the definition of a software package that has not yet been created.

3. Contracts for computer services typically include the services of programmers and consultants. *See* Hoffman, *supra* note 1. *See also* Adam, Gordon & Starr, *Contractual, Financial, and Tax Issues in Major Procurements*, 4 COMPUTER/L.J. 465, 487-96

that is responsible for vagueness, a frequent feature of development and consulting contracts. The user cannot adequately define the end product and the vendor desires merely to provide manpower to attempt to achieve an indefinite result. Consequently, there are vague statements in such contracts regarding end results and the estimated man-hours necessary to complete those results. The parties may frequently neglect to include statements which define their expectations.

Many vendors of development or consulting services assume that the more vague the contract with the user, the better their position because they are less committed and therefore have reserved more "freedom." This is true only in a limited, superficial way. Substantively, the more detailed and precise a contract, whether a contract for concrete results or for services, the better the contract will serve the needs of both the vendor and the user.[4] A vague or ambiguous contract serves only to fuel disputes and increase potential dissatisfaction on both sides. A well drafted contract, on the other hand, will anticipate and provide for possible disputes and dissatisfaction. It should delineate contractual means for dispute resolution.[5]

II. MULTIPLE VENDOR SITUATIONS

Often, when a user contracts with a vendor for consulting services or software development services, the services will merely be a part of a larger installation. Such an installation may involve other vendors and products including hardware, maintenance, package software, telecommunications, and financing companies. The consulting or development firm will frequently be an integral part of the planning process from the perspective of both the vendor and the user. The consulting or development firm must coordinate the responsibilities assumed by the multiple vendors. Where responsibilities overlap, or where problems in assessing fault for performance delays can be anticipated, it is essential that the contractual documentation be structured to deal as effectively as possible with such situations with a view to achieving maximum efficiency in the implementation process.

There are various ways to approach this problem. One way which may be more suitable for less complex multiple vendor situations is to hold a single vendor responsible for overseeing or monitor-

(1984)(authors assert that vendor wishes only to contract to procure a finished product that conforms to the user's service specifications; however, user should attempt to contract for a "bug free" product).

4. *See* Corbin, CONTRACTS § 95 (1952).

5. *See infra* notes 55-58 and accompanying text.

ing compliance with the contractual obligations of all the vendors. The "lead" vendor's responsibilities include communicating to the user any problems or delays arising during the implementation. If it fails to do so, it may be liable for increased costs assumed by the user. If the user contracts with only one vendor for the entire implementation, and that vendor subcontracts to another vendor, the coordination problem rests with the first vendor. The original vendor is therefore responsible for resolving difficult "finger-pointing" problems.

On the other hand, where, as may be expected, the vendor refuses to accept such responsibility, the parties must carefully draft documents to coordinate various implementation stages between the user and the vendors. More importantly, however, the parties must establish a contractual framework to prevent disputes among vendors and to assign liability for damages caused by one or more parties. One element necessary in all contractual arrangements of this nature is the concept of the periodic status meeting. "Project Managers" should be appointed for each user and each vendor. Periodic meetings should be held to allow the project managers to discuss the progress of the implementation. More importantly, however, the project managers should discuss any delays or problems encountered by the user or the vendors.[6]

An effective aid in all major installation contracts involving multiple vendors is the "master implementation schedule." A master implementation schedule should provide target dates and completion schedules for the user and the vendors.[7] While a vendor may prefer not to commit itself to an implementation schedule, such a commitment is advantageous because it details the obligations of each party within the total implementation. One disadvantage will likely surface, however, if there is a delay: The master implementation schedule will become meaningless unless it is continuously updated.[8]

The concept of "rolling estoppel" is a valuable addition to any contract involving performance scheduled over a substantial period. Basically, rolling estoppel requires a party to object at certain intervals if it is aware of a problem or delay, regardless of whether that party is at fault. A party that fails to object is estopped from complaining later about that problem or delay. Each contract should expressly provide

---

6. An example of a contractual "status meeting" provision used for a major acquisition is attached as EXHIBIT A at 516.

7. It may not always be feasible to provide target dates for all installation contracts. *See* Adam, Gordon & Starr, *supra* note 3 at 487-96.

8. Even when the master implementation schedule is continuously updated, it may lose its relative value. Its utility must be compared to the effort required to revise it. *Id.*

that the project managers must, at the periodic status meetings, raise any objections to problems or performance delays arising since the previous meeting. A vendor or user waives its objection to performance if its project manager fails to object to a problem of which it knew or should have known, regardless of the cause of the problem. In the event that a user or vendor does announce a problem or delay caused by the neglect of another, and fault is disputed, each affected party should be required to provide a written statement indicating that it has not defaulted and it did not cause the delay or problem in question. In the event that the innocent parties discover that the party at fault denied responsibility, either direct or liquidated damages should be paid to the injured parties.[9]

A final contractual resort, a "force majeure" clause, may be appropriate. A force majeure clause is particularly useful in multiple vendor situations. A force majeure clause provides that an innocent party is not liable for unforseen delay. Additional language may allow a party to terminate its contractual obligations after the expiration of a specified time. Therefore, if a party experiences a delay due to "force majeure" and its performance is critical to the performance of another, the latter may terminate its agreement and employ its resources elsewhere.

Furthermore, standard procedure in drafting such contracts should include specific control language to "roll forward" or extend target dates for performance in the event of an unforseen delay. Target dates should be extended for delays caused by either party. The extension should equal the period of the delay.

Implementations with multiple vendors obviously require detailed planning and cooperation. Furthermore, they require full delineation of the in-depth contractual integration of user and vendor responsibilities and liabilities.

III. CONSULTING AGREEMENTS

Consulting contracts should delineate in some detail the scope of services to be rendered. The contract should also contain a list of "deliverables" or documentation from which the user can implement the

9. It should be kept in mind while reviewing these materials, and without going into an extended discussion, that certain cautions should be observed in structuring liquidated damage provisions to insure their validity and their impact on other available remedies, either express or as provided by law. In particular, a user or vendor must be cognizant of the possibility that by inserting a liquidated damage clause, it may have "elected a remedy" to the exclusion of others which may not be advantageous. *See* Farnsworth, CONTRACTS § 12.18 (1982).

results as well as aid the consulting firm in carrying out its contractual obligations.

Consulting services contracts are perhaps the most difficult to document accurately and confidently. This is due to the fact that it is extremely difficult to contract for concrete results, simply because the contracting company will not usually be in a position to articulate the results desired in advance. It is in many cases the job of the consulting company to define adequately its goals and the means to achieve them. It is for these reasons that most consulting services arrangements are structured in a labor/hours type of compensation, rather than a fixed fee. This puts the user in a somewhat unavoidably precarious position since it is almost always impossible to define in advance the amount of time required to complete a given project. In a labor/hours fee arrangement, the user's only protection against excessive costs is, if contractually permitted, to terminate the contract once the costs obviously outstrip the results. That dilemma usually leaves an unhappy user with an unfinished product, and a vendor with an unsatisfied client.

An experienced user may attempt to convert the consulting contract into a fixed fee arrangement.[10] As a compromise, the contract can be broken down into segments. These segments may be billed on an hourly rate for the purpose of defining the scope of the consulting services to a point at which a fixed fee is determinable.

In any lengthy or major project, periodic status meetings should be required under the contract. Furthermore, a timetable for delivery of documentation (both to user and from user) should be set forth. The parameters of the documentation should be detailed as fully as possible.

Many user companies that contract with consulting firms insist on the contractual right to approve or reject the individuals performing the services for the consulting firm.[11]

Another provision to consider when contracting for consulting services is the right of ownership to any proprietary information which may be generated during the contract. Furthermore, a consulting service contract should include a provision for protecting confidential in-

10. A fixed fee arrangement includes "the concept of estimates with cost over-run ceilings." Adam, Gordon & Starr, *supra* note 3 at 478.

11. For instance, some companies will restrict the consulting firm's right to substitute or replace individual consultants. While this is undesirable from the consultant's point of view, it may enhance its ability to argue later that the exercise of those rights by the user caused unnacceptable results to that user. *Id.*

formation acquired by either party.[12]

IV. SOFTWARE LICENSE AND DEVELOPMENT AGREEMENTS

Software development contracts can take many forms but are usually geared to one of two areas. The user either attempts to obtain a totally new software program specifically tailored to its particular needs, or it desires some customized modification to an existing program. Below are some key factors which should be addressed in structuring the type of contract which will provide a smooth and orderly basis for performing obligations and dealing with disputes.

A. *Specifications*

Specifications are at the heart of the contract. Parameters for functional, operational, and performance specifications should be made a part of the contract. It will then usually be up to the vendor to develop a set of detailed design specifications for developing (or modifying) a package to conform to the user's requirements. These design specifications should be submitted for the user's written approval on a planned timetable. If they are not satisfactory, the vendor will correct and resubmit them. This process should be repeated until the parties reach a set of mutually acceptable specifications.

B. *Programming Development: Documentation*

Once the parties have agreed upon the detailed specifications, the vendor should commence coding and programming. During this phase, it is desirable to maintain the concept of regular status meetings.[13] The purpose of scheduling status meetings is not only to alert the parties of potential problems or delays. Regular status meetings reduce the possibility or potential of either party blaming the other, or its staff, for inadequate results due to non-cooperation. It is probable that the development in a major transaction will be accomplished in phases. This is especially true if any portion of the development or modification is dependent on another phase. This also presents the opportunity for intermediate review of documentation by the user to prevent early misunderstandings or mistakes from causing the entire project to be scuttled at the end, thereby causing considerable loss in time and money to both parties. In addition, the phased development will allow both parties to gauge more accurately the progress of the

12. A sample index for a consulting and software development agreement is attached as EXHIBIT D at 521.

13. *See supra* note 6 and accompanying text.

project. At the conclusion of each phase of the programming development, the vendor should provide documentation to the user for approval on a periodic basis.[14] Any disapproval of documentation by the user should be accompanied by a detailed statement setting forth with particularity those items that do not conform to contract requirements.

C. *Acceptance Testing*

Acceptance testing provisions in a contract can be as simple or as complex as negotiation leverage will allow. In fairly routine development transactions, the vendor may achieve some benefit by providing that the software will be deemed accepted and installed when the vendor so certifies. In more complex developments, however, such a provision may have the potential for creating a good deal of user dissatisfaction. It may result in unnecessary expenses.

Assuming that user satisfaction is a goal, acceptance testing criteria should be as objective as possible in complex development transactions. Other standards are less acceptable, such as requiring the software to operate on a rolling forward basis for x number of consecutive days without a "specification non-conformity." If one occurs, the cycle must start over until the standard is met. Preferably, clear objective criteria such as benchmark testing should be established in advance.

In some acquisitions, performance incentives for the vendor may be appropriate. These can take many forms. For example, assume the vendor is late in meeting some intermediate delivery deadlines and, according to the contract, is liable for x dollars of liquidated or other damages. In the event the vendor nevertheless completes final acceptance testing on the original date set forth in the contract, the user should agree to waive such damages. Perhaps a bonus should be offered to the vendor for early completion of its installation and acceptance testing obligation.

Actual testing of the developed software presents a number of complexities. In a software development project of major proportions, testing of a hardware system should be completed at the vendor's site. This testing is considered to be the initial or interim acceptance test. After the interim results have been accepted, the software will be tested at the user's site in an on-line, full, or test data base environ-

14. The authors suggest that documentation be provided to user by vendor on a weekly basis during the programming development phase. *See* Adam, Gordon & Starr, *supra* note 3 at 487-88.

ment. In a multiple vendor situation a user may insist that the hardware vendor cooperate with the software vendor. For instance the user, in its contract for hardware, may insist that the developed software be tested on the equipment before it leaves the hardware vendor's premises.

An acceptance procedure geared to achieve a smooth installation could be structured as follows. First, after the user approves the documentation for a particular phase of the software development,[15] the vendor, at its own site, will test that phase.[16] After test site acceptance, the software will be tested in an on-line productive processing environment on the user's system. The successful running of this test at this phase can be labeled "Interim Acceptance." Each subsequent phase will be tested on the user's equipment while all previously accepted phases are in operation. Thus, there will be a pyramid testing effect until all phases are installed at the user's site.[17]

In a multiple vendor situation the contract should be drafted with some flexibility. Flexibility is necessary in order to deal reasonably with delay. For example, the software vendor may complete all documentation and in-house testing before the user's equipment is available for live testing at any phase.

D. *Midstream Modifications*

Appropriate flexibility should be provided in the contract to allow for subsequent modification of the contractual specifications. The user may, in light of ongoing studies or implementations, seek modification. Naturally, the software vendor will then be forced to revise its prices. The contract should therefore detail the appropriate administrative procedure for dealing with such a situation.[18]

E. *Payment Alternatives*

Payment provisions can be as varied and complex as negotiation will allow. Payment provisions may be based on materials, time, or a fixed fee payable on final acceptance. Naturally, the end result may fall somewhere in the middle.

Software vendors often object to a fixed fee arrangement during

---

15. Hereinafter, "Documentation Acceptance."

16. Conformance with the test results with the accepted documentation will be the "Test Site Acceptance." *See* Adam, Gordon & Starr, *supra* note 3 at 488.

17. "Final Acceptance" will occur after all phases of the developed or modified software are up and running in a live environment. *Id.*

18. A sample provision detailing modification and change procedures appears as EXHIBIT B at 518.

the development of the function specifications. Such an objection is understandable because these specifications are usually reworked repeatedly at the request of the user. It is possible, however, to structure an interim agreement. In an interim agreement, the vendor develops the functional specifications and charges in accordance with the time expended and the materials used. The user may insist that it retain the power to terminate the entire contract in the event that the costs far exceed the results. Once the functional specifications are developed, the vendor may be obligated to quote a fixed fee for the programming development and installation. In the event the fixed fee is excessive, the user will likely take the specifications for which it has already paid and seek other bids for the development. When the development is actually a modification of an existing package which is presumably owned by the software vendor, the vendor may possess a "lock" on the development work.

Assuming that the vendor has accepted a fixed fee for some portion of the job, it should attempt to collect a major portion of the fee up front. This practice helps to protect the vendor from the need to sue for costs incurred when a user unfairly refuses payment due to unforseen problems. On the other hand, if the user has sufficient negotiating leverage, the cost may be broken down by development phases and paid upon the completion of each phase. At least part of the total fee should be tendered as a down payment at the execution of the contract.[19] Such down payments are generally regarded to be reasonable in a labor intensive product such as software development.

The user may seek a compromise position if the software vendor insists that the entire project be completed on a time and materials basis. For instance, the user may insist on the right to terminate the agreement at any time. It may also insist on a monthly billing by the software vendor, in which the user is required to pay a fixed percentage of the total fee. For example, the user may pay 80% of each in-

19. Under this arrangement, the user might pay a total fee of $200,000.00 by breaking down the cost as follows:

Down Payment	(20%)	$40,000.00
Documentation Acceptance	(20%)	$40,000.00
Test Site Acceptance	(40%)	$80,000.00
Interim Acceptance (on user's equipment)	(10%)	$20,000.00
Final Acceptance (on all aspects of software implementation)	(10%)	$20,000.00
Total	(100%)	$200,000.00

voice and withhold the remaining 20% until the completion of all the periodic phases. The user may also assert its right to audit the project records maintained by the software vendor. The purpose of such an audit is to confirm the invoices rendered. It would be proper to provide that the user must pay for the audit unless an error in billing was discovered. If a significant error was discovered, it may be appropriate to require the vendor to pay for the audit.

The vendor should attempt to define as many costs arising out of these transactions as possible.[20] Furthermore, the vendor should indicate clearly which costs the user will be expected to pay and when the payment is due.

F. *Warranties and Representations: Limitations of Liability*

Warranties and representations of software performance are important to the user. The software vendor should seek to limit its exposure by limiting its warranties.

Warranties on software performance are invaluable to the user. Such warranties may, however, contain hidden liabilities to the vendor. Therefore, the vendor must exercise its discretion when it offers warranties.

The vendor must consider the applicability of the Uniform Commercial Code to software licenses. Under the U.C.C., it may wish exclude certain warranties which would otherwise lead to substantial liability.[21]

The most limited warranty is obviously no warranty at all. In other words, the vendor promises merely that the software will be delivered and installed "as is." Even the most unsophisticated user will likely object to such a broad exclusion.[22] The vendor may, however, want to offer a limited warranty. For example, the vendor may promise that service will be performed in a competent workmanlike manner or that the software, when installed, will perform according to certain accepted documentation. The vendor should be careful to state ex-

---

20. Articulated costs should include:

a. Costs for providing conversion services for converting a user's data base

b. media costs (tapes and disks)

c. use of vendor's computer time in certain instances

d. costs for providing training to user's staff

e. travel and lodging expenses

21. *See infra* APPENDIX A at 525.

22. Furthermore, such a warranty does not help the vendor build good will in its product reputation.

pressly that all other warranties, expressed or implied, are excluded.[23]

In connection with any warranties of performance by the vendor, the contract should include specific and limited remedies in the event of a breach. For example, if the software has material defects, the user's sole remedy would be to have the vendor "fix" the bugs to conform to the warranty. Any warranty should specifically exclude claims based on defects caused by the user, such as modifications to the software. Furthermore, the warranty period should be limited in duration because developed software is rarely free of coding errors even after years of use.

One warranty frequently requested by users in a software licensing agreement is that the software contain no preprogrammed termination routine, which will cut off the user's ability to process data in certain events.[24]

The liability of the vendor should be limited by specific statements. Because the potential damages resulting from faulty software can be devastating,[25] the contract should specifically state the circumstances for which the vendor is liable. Typically, a vendor will limit its liability to the amount paid by the user, thus giving the user a mere money-back guarantee. When user payments are made on a continuing time and materials basis, the vendor will limit its liability to amounts paid by the user only for a certain period such as the three most recent monthly payments. While clauses that completely eliminate monetary liability are used, such clauses may, in a litigation scenario, cause a court to go out of its way to find a reason to discard such an exclusion.[26]

The vendor should certainly exclude all incidental and consequential damages arising out of the agreement whether such damages

---

23. *See infra* APPENDIX A at 525.

24. Such events may include use of the program beyond a certain date or the transfer of the program onto different CPU. For an example of a warranty covering this issue, *see infra* EXHIBIT C at 520.

25. The danger of potential physical and financial damage is obvious where faulty software is implemented in airline radar or banking hardware.

26. If the software agreement is in a non-commercial setting, such a limitation clause is prima-facie unconscionable under U.C.C. §2-719(3). In a commercial setting, however, in order for a clause limiting remedies to be unconscionable, there must be either a severe imbalance in the bargaining positions of the parties or a lack of essential purpose in the clause. Posttapes Assocs. v. Eastman Kodak, Inc., 450 F.Supp. 407, 411 (E.D.Pa. 1978); *see* Johnson v. Mobil Oil Corp., 415 F.Supp. 264 (E.D.Mich. 1976).

A court might also strike such a clause as an attempt to limit liability for negligence. *See Neville Chemical Co.*, 294 F.Supp. 649 (W.D.Pa. 1968), *aff'd.* 422 F.2d 1205 (3d Cir. 1970). *But see* Burroughs Corp. v. Hosbsion, 28 U.C.C. Rpt. Serv. 1378 (M.D.Fla. 1980); Bakal v. Burroughs Corp., 74 Misc.2d 202, 343 N.Y.S. 541 (Sup.Ct. 1972).

were foreseeable or unforeseeable. Such an exclusion is advisable even if the user indicated in advance that such damages were possible. The breadth of possible incidental and consequential damages arising out of a relatively modest problem may force a company out of business.

G. *Ownership Rights*

Ownership rights to developed or modified software should be clearly stated in the contract. Express ownership rights help to avoid future disputes.

Since standard package software is developed by the software vendor and is only licensed to the user, the vendor will properly assert ownership. Likewise, the vendor should attempt to assert ownership rights over custom software as well. A user may, however, have a proper position to resist such an assertion. If the software vendor retains ownership rights in custom software, the user must finance the development of a valuable software product. The user, however, cannot subsequently relicense the software to defray the costs of its development. The vendor on the other hand, has already been fully paid by the user to develop the custom software. It then may relicense that software for additional profit. If the vendor in developing custom software from scratch has simply provided development or conversion services to a user, the user might assert full ownership rights over that software. If the software vendor owned and modified an existing software package, it may retain ownership rights. The vendor may, however, compromise and allow the user to obtain limited licensing and marketing rights.[27]

H. *Source Code and Modification Rights*

1. Modifications

The user may find it necessary to retain the right to further mod-

---

27. Alternative compromises between the software vendor and the user may include the following:

a. Joint ownership. Joint ownership may be problematic because both parties are potentially exposed to liability from the marketing activity of the other party.
b. Sole ownership by the user with the grant of limited marketing rights to the vendor.
c. Sole ownership by the vendor with the grant of limited use and/or marketing rights to the user.
d. Sole ownership by the vendor with royalties payable to the user.
e. Sole ownership by the vendor in return for reduced development charges, future services, or other products.

See Adam, Gordon & Starr, *supra* note 3 at 493.

ify the software with its own resources. The user must, however, be aware that such internal modification may impact the obligations of the vendor. For instance, internal modification may impact the vendor's contractual obligation to maintain the software. If possible, the contract should state that user modification will release the vendor's obligation to provide maintenance unless, for maintenance purposes, the user restores the software to its pre-modification form. In the alternative, the user may notify the software company in advance of the proposed modification and obtain approval and confirmation that such modification will not impact vendor obligations.

The contract should specify the extent of ownership rights in any software modifications made by the user. The "grant back"[28] is one alternative to consider in connection with user modification.

2. Control Over and Access to Source Code and Technical Documentation

Source code is written in program language and can be read by a person knowledgable in that language. Source code is necessary for the maintenance or enhancement of software because it contains the programmer's original notes regarding particular portions of the program. The source code may also describe specific types of data which may be helpful in debugging a program and, therefore, facilitate future alterations. Object code, on the other hand, is written in machine language which contains operating instructions prepared from the source.

It is assumed in the following discussion concerning control over and access to the source code, that the parties have agreed that the vendor will retain ownership rights in the developed software. Furthermore, it is assumed the that the user merely retains a restricted license to use the developed software. In a sophisticated software development transaction, the user may employ a sophisticated data processing staff which is heavily involved in assisting in the development. The vendor may, therefore, have no practical means of preventing the user from obtaining access to the source code. Since source code is human-understandable, any person given access to the source code would be able to copy the source code, change the code, and produce a fresh object code which effectively disguises the original object code to create a competitive software product. Unfortunately, it is difficult for a vendor to prove such unauthorized use of its source code. Thus, in order for a vendor to protect its market structure, it

28. Such a provision may require a user to assign its modifications to the vendor, or grant to the vendor an unrestricted license to the modifications.

usually maintains control over and restricts access to its source code and related documentation. It is true that customers may simply copy the object code in order to create competitive products. Copies of the object code, however, are not easily disguised or modified. The vendor should consequently be able to detect such unauthorized use of its object code.

At the negotiating table, the user might demand either control over, or at least access to, the source code and related technical documentation. Even if the user believes that the vendor is unlikely to release its source code, it may attempt to procure concessions from the vendor. To convince a vendor to release the source code, the user will typically describe the various risks to which it is exposed without control over or at least access to the source code and related technical documentation. For instance, the vendor may encounter future financial difficulties resulting in bankruptcy or insolvency proceedings which often tie up the source code in receivership and legal problems for many years. Meanwhile, the user is unable to maintain, modify, or enhance the software without access to the source code. Likewise, the user faces the possibility that a disaster may destroy the existing copies of the source code held by the vendor.

The vendor usually refuses to grant the user control over and access to its source code. Vendors argue that they must retain ownership of the source code because the source code is used for maintenance and modification. If the user is allowed to make its own changes to the source code, the vendor may be unable to fulfill its obligations to maintain the software. Failure to maintain the software may result in a breach of warranty by the vendor. The vendor usually attempts to protect its market structure by maintaining control and ownership over its confidential and proprietary information. Therefore, the vendor will usually refuse to comply with such a demand and may merely allow the user access to the object code. It should be noted that the increasing availability and sophistication of decompilers and disassemblers which are used to translate object code into source code has somewhat diminished a vendor's ability to protect its source code. In fact, a company's survival in the rapidly changing computer technology field may require the use of reverse engineering. Reverse engineering has become a universal practice in some businesses.[29]

Like many points covered during the negotiation process, the resolution of the source code control question generally depends on the

29. *See* Klein, *Reverse Engineering of Microchips is Slow, Costly — and Universal*, Wall St. J., Aug. 5, 1982 at 1, col. 6.

bargaining strength of each party. A difficult situation arises when a vendor and a user with equal bargaining power must reach a compromise. One solution is for the vendor to allow the user to hold the source code with certain safe-keeping conditions. The source code should then be used only if circumstances preclude the vendor from continuing to provide the necessary support. Most vendors will think such an arrangement too insecure, however, and insist on additional security.

Another solution is to require the vendor to release the source code to the user upon the happening of certain events. Unfortunately, such an approach is useless if the software is destroyed, if the vendor becomes insolvent, or if there is simply a dispute. Under such circumstances, the approach is useless because the triggering event occurs too late[30] to protect the vendor.

If the vendor retains control over and access to the source code, it might consider including a clause to prohibit the customer from using reverse engineering to develop a source code from the vendor's object code. If, however, a vendor is forced to release its source code to the user, the vendor should investigate various methods of protecting its code.[31]

As a compromise between the legitimate concerns of both the vendor and the user, the parties may enter into a separate escrow agreement. For instance, the parties may agree that the source code will be deposited with an escrow agent. The appointed escrow agent will be instructed to release the source code to the user only upon the happening of specified events. Such an escrow account is similar to the traditional escrow accounts used in real estate purchase agreements. In preparing the escrow agreement, the following considerations should be reviewed.

a. Define escrow materials and documentation

Some escrow agreements include a residual clause that requires

30. *See* Laurie, *Protection of Trade Secrets and Object Form Software: The Case for Reverse Engineering*, 1 COMPUTER/L.J. 1 (1984). Preferable alternatives such as escrow accounts and trust agreements are discussed below.

31. *See* Hoffman, *The Software Legal Book*, Part II, E-1, 2 (1981). Some vendors insert into the source code various program commands which do not perform any useful function, but rather serve as a method of detecting unauthorized use of the source code. Similarly, some vendors give the user a unique identifier imbedded in the source code, while others include a disabling code which causes the software to self destruct on certain events. It should be noted, however, that vendors using such protection should be advised to fully inform the user of the potential effects of the disabling code. *See supra* note 24 and accompanying text.

the vendor to place into escrow all necessary materials to allow a skilled third party programmer to independently maintain and enhance the software.[32]

b. Verification of escrowed materials

Any inspection procedures that allow a user or a third party to verify that the vendor has deposited the proper documentation in the escrow may pose a threat to the vendor's trade secrets. This is true especially if the third party is a potential competitor.

c. Fees

Vendors usually require that escrow and third party verification fees be divided equally between the vendor and the user.

d. Definitions of triggering events

Each event that may trigger the release of the source code from the escrow agent to the user should be adequately defined. Triggering events may include a vendor going out of business or declaring bankruptcy.

e. Release mechanism

The parties must determine whether the source code is to be released immediately upon the occurrence of a triggering event or whether the vendor is entitled to notice prior to the actual release.

f. Conflicting demands

The contract should detail what will happen if the user claims that a triggering event has occurred and the vender denies that claim.

g. Trust agreements

Certain commentators have suggested the use of trust agreements. It has been suggested that a trust agreement be used as a mechanism to avoid some of the problems posed by the Bankruptcy Code.[33] While issues similar to those presented in an escrow account exist, the principal difference is that a trust is established solely to ben-

32. *See* Gilburne, *Source Code Escrows: Meaningful Solution or inadequate protection*, CN Report Vol. 4 Issue 6.

33. 11 U.S.C. §§ 1 - 1255 (1982). This article does not encompass the full impact of the Bankruptcy Code on software licensing agreements. Failure to disclose the potential impact on source code escrows could result in misrepresenting the effectivesness of those arrangements.

efit the user. A trust will allow the user to assert that the arrangement is not executory.

The Bankruptcy Code provides the trustee with the right to assume or reject the debtor's executory contracts and unexpired leases.[34] Since delivery of the source code to a user has not occurred under a source code escrow, such an arrangement would likely be considered executory. It gives the trustee the right to preclude delivery — a result that is completely contrary to that presumably intended by both parties. Furthermore, Section 365(e)(1) of the Bankruptcy Code[35] declares that *ipso facto* clauses are unenforceable. Such a clause, if enforceable, would automatically hold a party in default at the time it files for relief under the Code. Accordingly, standard default provisions in favor of either the user or the vendor that suggest default on filing under the Code are simply not enforceable.[36]

I. *Miscellaneous Issues*

A list of miscellaneous issues that must be considered when contracting for software license and consulting agreements follows.[37]

1. Conversion

Frequently, software development agreements provide that the software vendor and the user will both assist in the conversion of the user's data base. Details of any such conversion service should be included in the agreement. Open-ended cooperation clauses, wherein the user or the vendor agrees to provide full cooperation in the conversion of the system, may become a primary issue in subsequent disputes. To avoid future complications, the contract should detail the conversion service. The contract may further require the user to give notice to the vendor of the need for cooperation. Providing such notice may reduce claims that the vendor failed to cooperate in the conversion.

2. Machine Use Restrictions

Although less likely in a straight development situation, software

34. 11 U.S.C. § 365 (1982).

35. *Id.*

36. For a fuller description of the actual rights of either party in these situations, see Gordon, Starr & Weathers, *Special Problems in Computer Contract Litigation*, CONTESTING COMPUTER DISPUTES (1981).

37. For an index of sample provisions contained in a heavily negotiated development agreement, see EXHIBIT D at 521. For tax considerations in licensing software, see APPENDIX A at 525.

agreements commonly restrict use to the licensed software. Those permitted to use the software may include named users and single CPU's at named locations which support only those terminals operated by the user. Such a provision operates to protect the vendor's market structure. It also allows additional (if perhaps discounted) license fees for additional users of the software. The agreements may also limit the user to processing its own data and forbid the user from processing any third party data. A topic of growing import is whether the employees of the corporate user should be allowed free access to the software for personal use at home.

3. Training

Adequate training of the user's employees will help to bolster user satisfaction. Training will hopefully provide realistic levels of performance expected by the user. Any formal training classes, on-site training, sufficient user's manuals, and other forms of training should be set forth with particularity.

4. Transfer and Assignment Rights

Rights and restrictions depend heavily on the structure of the ownership rights to the software. The ownership rights should be expressed with specificity in the contract.

5. Approval Rights of Employees

The above discussion of the user's rights to approve or reject the vendor's employees in the context of a consulting agreement is equally applicable to a software licensing agreement.

6. "Force Majeure" Clause

The above discussion of force majeure in the context of a multiple vendor situation is equally applicable to a software licensing agreement.

V. CONFIDENTIALITY AND PROPRIETARY RIGHTS ISSUES

A. *Confidentiality*

Depending upon the nature and the size of an implementation, it is possible that employees of the vendor and of the user will be parties to the agreement. It may be necessary in a major development transaction to protect the confidentiality of information available to employees. This is true especially when the user may have rights to

oversee or assist in development at the vendors' place of business. Each party is in the best position to ascertain the extent of information to which the other party and its employees will be exposed. Each party is also in the best position to measure the level of protection necessary to provide adequate security. Perhaps a well-drafted confidentiality section in the agreement will be sufficient. A comprehensive, but generic, list of potential confidential information is recommended. Additionally, in the event that a party becomes aware of any incidence of a breach of the confidentiality provisions, it should be obligated to notify the other party. A party may, however, desire a more comprehensive safeguard provision. It may require that the other party's employees sign individual confidentiality agreements. Each individual agreement may mirror the terms of the master agreement.

B. *Non-Solicitation*

It is often appropriate for a vendor to insist on a clause to prohibit the user from soliciting the vendor's employees for its own employment. This is especially true in software development agreements in which one or more of the vendor's employees might be invaluable on the user's staff. Negotiable variables may include the time frame for such non-solicitation as well as the level of personnel covered by such a clause.

C. *Proprietary Rights*

The great demand for software, which is primarily due to increased computer use and a shortage of programmers, has increased the value of existing and newly developed software. Not suprisingly, the pressure to misappropriate or misuse proprietary software is likewise increasing. Vendors now recognize that it requires significant effort and investment to reproduce hardware products. It is a low cost proposition to replicate accurately software that may have cost millions of dollars to develop. It is, therefore, no longer desirable for vendors to bundle hardware and software or to provide free software to secure hardware orders. Instead, vendors license their software through detailed licensing agreements in which they include various protection methods.[38]

---

38. *See* Root, *Protecting Computer Software in the '80s: Practical Guidelines for Evolving Needs*, 8 RUTGERS COMPUTER AND TECH. L.J. 205 (1981).

1. Patents

Patent law provides the broadest protection for technology,[39] but it remains an open question whether patent protection is available for software technology.[40]

2. Copyrights

A copyright, unlike a patent, protects the specific expression of an idea rather than the idea itself.[41] It is unclear whether certain forms of computer software are protected by copyright.[42]

3. Trade Secrets

A trade secret, like copyright protection, does not entitle the owner to preclude others from independently developing similar technology. Unlike patented or copyrighted works, however, trade secrets are materials which have been given *restricted* dissemination to the public. Secrecy is the essence of this form of protection. Once a trade secret is placed in the public domain either by inadequate internal controls or unrestricted dissemination to the public in the course of marketing, the trade secret is lost. Thus, software licensing is more likely to protect trade secrets embodied in the software packages. Likewise, restrictions on the use or disclosure of software are more likely to be enforced when software is licensed rather than sold. Moreover, under software licensing agreements, a vendor may retain proprietary rights in and title to the software.

D. *Methods of Preserving Confidentiality*

To protect its proprietary rights in the software adequately, a vendor must advise the user of the proprietary nature of the software and the need to maintain trade secret protection. The vendor must also

---

39. Under 35 U.S.C. § 154 (Supp. VI 1976) developers of various types of technology may exclude others from making, using, or selling similar inventions for a period of seventeen years, even if the invention is independently developed. Infringers may be required to pay damages (up to treble damages) and attorneys fees to the developer.

40. *See* Diamond v. Bradley, 450 U.S. 381 (1981); Diamond v. Diehr, 450 U.S. 175 (1981); Parker v. Flook, 437 U.S. 584 (1978); Dann v. Johnston, 425 U.S. 219 (1976); Gottschalk v. Benson, 409 U.S. 63 (1972). For a discussion of patent protection for computer software, see Bender, *Software Protection: The 1985 Perspective*, 7 W. NEW ENG. L. REV. 405, 412-18 (1985).

41. *See* Mazer v. Stein, 347 U.S. 201,217 (1954); Warrington Assoc. v. Real-Time Eng'g Systems, 522 F.Supp. 367, 368 (N.D. Ill. 1981). For a discussion of copyright protection for computer software, see Bender, *supra* note 40 at 419-33.

42. *See* Corp. Prac. Series, Washington Memorandum (BNA) No. 198, at 5 (1982); and 24 Patent, Trademark Copyright Journal (BNA) No. 592, at 387, 388 (1982).

include the following disclosure restrictions in the actual licensing agreement.

1. Acknowledgement

The vendor may consider including a provision wherein the user acknowledges that the software is proprietary and the property of the vendor.[43]

2. Furnish Only Object Code

If a vendor releases its source code to the user, it runs the risk of someone reproducing that code with sufficient modifications to disguise the original code. Therefore, if a user is given access to the source code, the vendor may want to include copyright and trade secret notices in the source code itself.[44]

3. Notification Requirements

The vendor may require the user to notify it immediately if it learns of unauthorized use or possession of the licensed software. Furthermore, some agreements require a user to cooperate with the vendor in tracking down and punishing unauthorized users, or even to pursue litigation against such users.

4. Copy Restrictions

Vendors should usually prohibit users from copying or duplicating any personal version of the software, object codes, or related documentation. In addition, some vendors now prohibit users from creating, by reverse engineering or otherwise, the source code from the object code or other information supplied to the vendor.

5. Trade Secrets and Copyright Notice

Vendors may include trade secret and copyright notices in both the source code and hard copy of the software. Such notice adds to the protection of proprietary rights.[45] If proper copyright notices are omitted, the vendor may lose its copyright protection. For instance, in

43. A vendor-oriented acknowledgement provision states that "user acknowledges that the software and related materials supplied by vendor to user are subject to the proprietary rights of vendor and constitute vendor trade secrets." Obviously the user will find the later portion of that provision somewhat oppressive and may wish to replace "constitute trade secrets" with "alleged by vendor to be trade secrets."

44. *See* Hoffman *supra* note 21.

45. *Id.*

Data Cash Systems, Inc. v. JS&A Group, Inc.,[46] the seventh circuit concluded that the failure to affix copyright notices of ROM[47] distributed as part of a computer chess game to over 2,500 end-users placed the underlying computer program into the public domain under the 1909 Copyright Act and rendered the program unprotectable.

6. Backup Restrictions

Many users reasonably request that they be permitted to produce backup copies of the furnished object code and data stored in various user files in case the original licensed software is destroyed or damaged. Vendors often acquiesce to such a request. An alternative is to incorporate certain restrictions into the licensing agreement. For example, vendors may limit the number of backup copies, place restrictions on the backup procedure, and require the user to submit to vendor information regarding the location, number, and media type of any backups.

VI. CONSULTANT'S LIABILITY

Data processing consultants and software developers basically sell their *services* rather than an inventoried *product*. Unfortunately, disputes involving service are more susceptible to conflict than are disputes involving physical goods. That is probably why more and more users (and counsel) seek new ways to impose liability on these vendors. Basically, a dissatisfied user of consulting or development services has both a contractual and a noncontractual avenue to support a claim against such a vendor.

A. *Contractual*

Primarily, the user will claim liability on the basis of the contractual provisions. If the vendor has limited its warranties in the contractual agreement and if there is an affective integration clause,[48] the vendor will certainly have decreased its risk of liability for damages under contractual provisions. A vendor may, however, have other contractual obligations such as performance on which a user may base a claim for damages. This is where provisions which limit liability

46. 628 F.2d 1038 (7th Cir. 1980).

47. The ROM (read-only memory) comprises part of the memory of a computer system which is indelibly imprinted with a program. *See* Bender, *supra* note 40 at 405, 417. Mr. Bender describes the ROM as "an entity which stands conceptually between the hardware and the software . . . ." *Id.*

48. An integration clause may state that any prior agreements or statements, written or oral, are superceded by the written contract.

play a major role in software development contracts.[49] Vendors with a contract clause containing an integration clause along with disclaimer of warranties and limitations of liability, will achieve relatively solid protection against contractual claims of substantial liability.

B. *Non-Contractual*

Given the fact that many vendor contracts contain provisions which limit contractual remedies and liabilities, users and their counsel increasingly turn to non-contractual theories to impose liability on vendors.

1. Negligence

Ordinary negligence has generally not been accepted as a basis for liability in cases involving mere financial loss.[50] One inherent problem in a negligence theory involves the duty of care owed to the user by the consultant. In ordinary negligence cases, the consultant must be judged according to its own skill and experience. This is difficult to apply in cases involving service and economic loss because there is no established standard to apply. Mere proof of error is not proof of negligence.[51]

2. Misrepresentation

The user may rely on claims of fraudulent misrepresentation. Such claims, however, are more difficult to pursue in terms of rendering future services, as compared to the sale of hardware. To support a claim for misrepresentation, five elements must be present:

a. A representation must be made by the consultant with the intent that the user rely on it;
b. Knowledge by the consultant that the representation is false;
c. The user must believe that the representation is true;

---

49. A limiting provision may state: "In no event shall vendor be liable to user for any claims arising under the agreement in excess of $————————."

50. *See* Chicago City Colleges v. Boeing Computer Services, Civ. No. 78C 1100 (N.D. Ill. 1978).

51. W. PROSSER & W. KEETON, THE LAW OF TORTS (5th ed. 1984).

> Professional persons in general, and those who undertake any work calling for specialized skill, are required not only to exercise reasonable care in what they do, but also to possess a standard minimum of special knowledge and ability. Most of the decided cases have dealt with surgeons and other doctors, but the same is undoubtedly true of dentists, pharmacists, psychiatrists, veterinarians, lawyers, architects and engineers, accountants, abstractors of title, and many other professions and skilled trades.

Id. at 185-86 (footnotes omitted).

d. The user must place some reliance on the representation; and
e. The user must suffer damages.

3. Strict Liability

While most strict liability cases have dealt with products and physical injury,[52] future litigation may expand the scope of strict liability to include economic loss. The basis of strict liability is a social public policy remedy, not a long-standing traditional legal rule. It is grounded in the idea that those providing the products (or possibly services) are better able to pay (or arrange for payment through insurance) than an individual somehow harmed by a defective product.[53] At this point, it is correct to say that this doctrine has been unsuccessful in cases involving services or commercial transactions involving parties of relatively equal bargaining power.

4. Professional Negligence (Malpractice)

There is a trend emerging in the computer industry to view computer consultants and software developers as "professionals" in the same context in which accountants, architects, lawyers, and physicians are traditionally associated.[54] This is partly due to the increased attention given to computers as well as the high degree of technical knowledge necessarily possessed by such individuals. Moreover, there is an increasing number of national associations evolving with respect to high technology activities. National standards are also increasing. Such developments tend to improve public opinion of computer technicians and consultants. The appeal to a user considering a computer consultant as a professional is that a professional owes the user a higher degree of care than it would owe in an ordinary negligence case. Negligence claims are typically not affected by contractual limitations or exclusions.[55]

A professional, generally speaking, must exercise reasonable care and that measure of skill and knowledge ordinarily posessed by mem-

52. *See* Wade, *Strict Tort Liability of Manufacturers*, 19 Sw. L.J. 5 (1965).

53. *See* Pound, *The End of Law as Developed in Legal Rules and Doctrines*, 27 HARV. L. REV. 195 (1914). "[There exists] a strong and growing tendency, where there is no blame on either side, to ask, in view of the exigencies of social justice who can best bear the loss, and hence to shift the loss by creating liability where there has been no fault." *Id.* at 233.

54. A detailed analysis of current arguments for and against imposing strict liability on providers of services for mere economic losses is beyond the scope of the article. For a discussion of strict liability *see* Prosser & Keeton, *supra* note 51 at 692-94.

55. *See supra* note 51.

bers in good standing in that profession.[56]

In some instances, professionals are not able to hide behind a defense of client or user participation in the specialized work.[57] This is of special importance to computer professionals. Therefore, simply because a user participates in drafting specifications, it does not follow that a computer consultant or developer can be excused if the end result of that professional's work is defective. This is true especially when it is clear that the user relied on the skill and judgment of the professional.[58]

VII. DISPUTE RESOLUTION

A. *Breach of Agreement*

In the event an uncontemplated[59] dispute arises between the user and vendor in the implementation process, there are several steps that the vendor should take in order to effectively assess and deal with such a dispute.

1. Nature of Transaction

Depending on the nature of the transaction, there may be substantial difference in determining an appropriate vehicle for dispute resolution. The vendor should therefore analyze the dispute in terms of the nature of the transaction or multiple transactions involved.

2. Involved and Related Parties

Whether in the context of answering litigation or determining contact for purposes of dispute resolution, recognizing the full range of involved parties, whether potentially culpable or otherwise influential, will be a necessary ingredient of the preliminary analysis.[60]

---

56. A contractual exclusion of liability is typically not effective against a negligence claim. *See* Farnsworth, *supra* note 9 at 333-34.

57. *See supra* note 51.

58. To date, no cases have been decided imposing professional negligence liability on computer consultants or developers. Some cases have discussed the issue favorably. *See* F & M Schaefer Corp. v. Electronic Data Systems, Inc., 430 F.Supp. 988 (S.D.N.Y. 1977). For a more comprehensive discussion of the topic see Brooks, *Professional Malpractice Liability*, U.S.C. Second Annual Computer Law Institute, 1981.

59. This includes a dispute contemplated by the parties in which the contractual remedies are regarded impractical in a given situation.

60. Consider the following entities or parties that may be involved, either directly or indirectly, with a given transaction:

a. Consultants
b. Direct Vendors
c. Manufacturers

3. Information gathering process

It is important to realize that the information helpful in determining an appropriate mode for resolving a dispute can be garnered from a great number of individuals. Such information is not necessarily possessed exclusively by individuals thought of in the first instance.

The vendor or its advisor should interview a full range of its employees. Many may be capable of informing the advisor of the difficulties sustained from the use of the system. Such employees may help to illuminate the historical progress of the implementation in the attempt to resolve the dispute between users or outside resources.

There may exist substantial documentation for review. The information that one will preliminarily want to review will not necessarily be "technical documentation." It may include logs, computer printouts that demonstrate the failure routines and error messages, and correspondence. It should not be forgotten, however, that a well-documented transaction at the contractual stage may be the key in determining the nature and scope of the dispute and how it should be resolved. Accordingly, one will want to review documents that were exchanged between the parties early in their relationship. A party involved in such a dispute should review the preliminary drafts of the contract, previous correspondences between the vendor and the user, promotional material, and other literature exchanged between the parties.

4. Role of the Attorney and Other Professionals

A dispute that has arisen out of lengthy data processing implementation will likely have a high degree of legitimate emotional concern. That emotional concern will likely impact the user's or vendor's perception of the next step. The attorney's role should include making an objective analysis of the magnitude of the dispute and, more importantly, to develop a pragmatic approach to resolve the dispute. The first suggested approach includes bringing the installation back on

---

d. Suppliers
e. Distributors
f. Employees of Vendor
g. Third Party Maintenance Companies, Divisions, or Separate Entities
h. Individual Sales People or Representatives
i. Financial Institutions
j. Leasing Entities
k. Licensors or Sub-licensors of Software Segments
l. Providers of Telecommunication Services
m. Contractors Involved in Site Preparation

track or arranging for the commencement of a newly negotiated installation arrangement with the user. The second suggested approach includes the introduction of an objective and calculated thought process to the determination of whether the user is entitled to seek relief from a vendor which has allegedly failed to fulfill its responsibilities.

In determining an appropriate course of action, counsel may wish to consider the participation of other individuals in the process. Such individuals may include accountants, experts, consultants, or other vendors.

B. *Resolution Alternatives*

Computer disputes often result in new acquisitions. Even though the user is dissatisfied with the present results, it may wish to continue business relations with the vendor. In this instance, the vendor should request that the parties enter a new transaction. A new transaction should be carefully documented to avoid the pitfalls of the failed transaction.

Arbitration is an alternative which should be reviewed if less formal attempts fail. There are, however, certain pitfalls to arbitration which may make the procedure less appealing than litigation in certain instances.[61]

VIII. CONCLUSION

Considering the increasing use of computer software products and the disputes surrounding those products, it is critical that the parties involved in any software or consulting arrangement structure an appropriate agreement and provide adequate protection to those parties. As Professor Zammit observes:

> Unlike the first lawsuits involving computers, the bulk of future litigation is not likely to focus on the reliability of hardware — that is, the physical components of a computer system. Hardware, by and large, is becoming more reliable. While there will always be the occasional lemon, the primary concern in the future will be related to software, particularly application software — the programs designed to perform a user's specific job. . . . To leave a vacuum in an agreement is to invite a court or jury to import its own notions of

61. An in depth analysis of arbitration in the context of a data processing dispute is beyond the scope of this article. For a discussion of arbitration in the context of a data processing dispute, *see* D. BROOKS, CONTESTING COMPUTER DISPUTES: LITIGATION AND OTHER REMEDIES IN CONTRACT, TRADE SECRET, AND COPYRIGHT CASES (1981). Jacobson & Gary discuss the resolution of data processing disputes in a chapter entitled "Arbitrating Computer Disputes."

what it thinks the parties meant or, worse, what it thinks is "fair." What a trier of facts thinks is fair, however, and what the attorney or his client thinks is fair are not necessarily the same thing. In the final analysis, "fair" terms are those mutually agreed on by the parties in an arm's-length transaction.[62]

62. Zammit, *Computer Software and the Law* Vol. 68 ABA JOURNAL 970, 970-71 (August 1982).

EXHIBIT A

PROJECT MANAGERS; REVIEW MEETINGS; PROJECT PROBLEMS; PROGRESS REPORTS SAMPLE PROVISION

1. User and vendor have designated one individual to serve as project manager and from time to time may designate in writing a temporary alternate Project Manager (the "Project Manager"), which individual or individuals shall be deemed to have authority to issue, execute, grant or provide any approvals, requests, notices or other communications required hereunder or requested by the other party hereto.

2. Once every other week (or as otherwise determined by the parties in writing) from the date hereof to the acceptance of the system, the vendor and user in the performance of their respective obligations hereunder since the last such meeting. At each such meeting, vendor and user may provide each other with a written status report specifying in detail any problem or circumstance (a "Project Problem") encountered since the last meeting (including without limitation the failure of user or vendor to perform any of its obligations hereunder of the inadequacy of any such performance by user or vendor) which might prevent vendor or user from meeting any of its obligations hereunder (any such delay being hereafter referred to as a "Project Delay").

3. Subject to the occurrence of (i) an event not within the control of or forseeable by vendor, (ii) adjustments required as a result of any authorized Modification/Change Request (as defined in EXHIBIT B), (iii) additional burdens incurred as a result of user's failure to perform any of its obligations hereunder, or (iv) a request or demand by user for the performance of services by vendor which vendor can reasonably demonstrate is outside the content or intent of the Design Specifications, vendor agrees on the terms and conditions set forth herein to perform all of its obligations hereunder in a timely fashion and at a cost to user as set forth specifically herein. If one of the events or circumstances specified in (i) through (iv) hereof should occur, vendor shall provide user in the report described in Section 2 hereof, a full description of any such Project Delay or Project Problem (as those terms are defined in Section 2 above) associated with any such occurrence at the next regularly scheduled meeting of the Project Managers following such an occurrence. In addition, vendor shall recommend alternative courses of action and/or design changes that will allow vendor to meet its obligations hereunder.

4. In the event that user is not reasonably satisfied that one of

the events or circumstances specified in (i) through (iv) of Section 3 above has occurred, user may direct vendor to proceed with any of the alternative services or actions recommended by vendor without prejudice to user's right to claim that vendor is not entitled to any adjustment in its obligations hereunder as a result thereof by so notifying vendor in writing, in which event vendor shall proceed promptly with such services or actions. However, vendor's proceeding shall not prejudice its rights to claim that it is entitled to an adjustment of its obligations; provided, however, both parties hereto agree not to impede the progress of the installation hereunder and to negotiate such claims in good faith within thirty (30) days of the acceptance of the System.

5. Submission by vendor or user of the reports specified in Section 2 hereof shall not alter, amend or modify user's or vendor's obligations pursuant to any other provision of this Agreement and, further, in the event the parties cannot agree on how to proceed in the event of a Project Delay, the provision of Section 4 hereof shall govern the continuance of the installation and the procedures to negotiate said disagreements.

EXHIBIT B

MODIFICATION/CHANGE PROCEDURES SAMPLE PROVISION

1. At any time during the term of this Agreement, should User desire Vendor to provide any additional services in the form of a modification of, or a change to, Vendor's performance hereunder, Vendor and User shall comply with the following administrative control procedures:

1.1 User's Project Manager shall submit to Vendor on the form attached hereto as Exhibit "——" all requests by User for any such additional services which alter, amend, enhance, add to, or delete from the Minimum Specifications, the Detailed Design Specifications or the supplemental Specifications, and/or the time and/or place of performance (hereinafter referred to as a "Modification/Change Request");

1.2 Vendor will evaluate each such Modification/Change Request at its standard rates and charges and return a copy of the same Modification/Change Request to User's Project Manager as soon as possible but not later than ten (10) working days following Vendor's receipt of the Request. Vendor's written response on said form shall include a statement of the availability of Vendor personnel and resources and the impact, if any, on the Project Completion Cost, the Project Completion Date or any Task Completion Time or Cost;

1.3 Should User elect to authorize such request, User will, as soon as possible but not later that ten (10) working days, authorize Vendor to perform the requested Modification/Change Request by returning a duly authorized copy of the request form to Vendor's Project Manager.

1.4 Upon such authorization by User's of the Modification/Change Request, Vendor will commence performance in accordance with such Request;

1.5 Vendor shall not be obligated to perform any additional services in advance of written authorization from User on the required Modification/Change Request form. In the event that Vendor commits resources to the performance of a Modification/Change Request without such prior written authorization, it shall be presumed that performance of such Modification/Change Request will have no effect on the Project Completion Cost, Project Completion Date or any Task Completion Time or Cost;

1.6 For the purposes of this Agreement, each Modification/Change Request duly authorized in writing by User shall be deemed incorporated into and part of the Minimum Specifications or Detail

Design Specifications, as the case may be, and each such Request shall constitute a formal change to this Agreement adjusting the Project Completion Cost, Project Completion Date and/or Task Completion Dates or Costs as finally agreed upon for each authorized Modification/Change Request. In no event shall the Minimum Specifications, the Detailed Design Specifications, or any provision in this Agreement be deemed altered, amended, enhanced or otherwise modified except through written authorization by User of a Modification/Change Request in accordance with subsection 1.3 above.

EXHIBIT C

PRE-PROGRAMMED TERMINATION WARRANTY

Vendor represents and warrants that the Software System (and any portion thereof) does not contain any timer, clock, counter or other limiting design or routine which causes the Software System (or any portion thereof) to become erased, inoperable, or otherwise incapable of being used in the full manner for which it is designed and licensed pursuant to this Agreement after being used or copied a certain number of times, or after the lapse of a certain period of time, or after the occurrence of lapse of any similar triggering factor or event. Furthermore, Vendor represents and warrants that the Software System (or any portion thereof) does not contain any limiting design or routine which causes such software to be erased, become inoperate, or otherwise incapable of being used in the full manner for which it was designed and licensed pursuant to this Agreement solely because such Software System has been installed on or moved to a central processing unit or system which has a serial number, model number, or other identification different from that on which the software System was originally installed.

EXHIBIT D

SAMPLE INDEX FOR A CONSULTING AND SOFTWARE DEVELOPMENT AGREEMENT[63]

PREAMBLES:

A. Summary of Software Proposal indicating User's requirements for software.

B. Necessity, if any, of obtaining hardware equipment for use in conjunction with software.

INDEX OF PROVISIONS:

1. *Development of Software System; Design Changes*
   - 1.1 Obligations of Developer to develop, install and consult with respect to the Software System in accordance with Design Specifications.
   - 1.2 Time schedule for developer to complete Detailed Design Specifications.
   - 1.3 Administrative control procedures for change or modification of Developer's performance.
     - 1.3.1 User's submission of request for modification to Developer.
     - 1.3.2 Developer's response indicating feasibility and impact on cost and completion dates.
     - 1.3.3 User's authorization to proceed with modification.
     - 1.3.4 Developer's obligation to commence performance of modification.
     - 1.3.5 Any authorized modification/change request shall amend the Agreement and/or Specifications accordingly.
   - 1.4 Specific Warranties and Representations of Developer regarding Software.
     - 1.4.1 Particular programming language to be used and maximum number of bytes of core storage to be required.
     - 1.4.2 Ownership and control of Software System.
     - 1.4.3 Hardware of Software malfunction backup procedures to minimize data loss.
     - 1.4.4 Software System to include appropriate edit and error recovery routines.

63. This index is an initial checklist of considerations. It is not designed to be comprehensive nor is it tailored to specific transactions.

1.4.5 Documentation requirements for development of Software System.

2. *Charges to User; Project Completion Cost; Task Completion Costs*
   - 2.1 Invoice and billing requirements of Developer; payment terms.
   - 2.2 Cost limitation; payment for Development Tasks; payment for completion of total project.
   - 2.3 Cost modifications; Project Delay reports; requirement of alternative courses of action provided by Developer; requirements of User's authorization for any cost modification.
   - 2.4 Provisions for implementing Developer's recommended alternative services or actions; agreement not to impede progress of installation.
3. *Project Managers; Review Meetings; Project Problems; Progress Reports.*
   - 3.1 Designation of Project Managers for User and Developer.
   - 3.2 Two week interval meetings of Project Managers to discuss status of project; delivery of written status report specifying:
     - 3.2.1 Any Project Problem or Project Delay.
     - 3.2.2 Length of anticipated delay resulting from a Project Problem.
     - 3.2.3 Reasons for the Project Problem and steps to correct it.
   - 3.3 Conclusive presumption of absence of Project Problem in absence of written notice.
4. *Completion of Development Tasks*
5. *Review of Deliverable Items; Interim Acceptance Tests*
   - 5.1 Delivery of deliverable items; acceptance or rejection thereof; correction of non-conformities.
   - 5.2 Unit testing and/or subsystem testing of completed programmed modules; submission and acceptance of written interim test procedures; submission by User of data requirements for test.
6. *Final Acceptance Testing; Final Payment*
   - 6.1 Procedures for Final Acceptance testing:
     - 6.1.1 Live test period on site.
     - 6.1.2 Processing of additional test data.
     - 6.1.3 Developer's obligation to correct all errors and specification non-conformities.

6.1.4 Acceptance on conclusion of testing on certain conditions.

6.1.5 Repetition of live test period until error-free for — days.

6.2 Payment of Outstanding Amounts upon Final Acceptance by User.

7. *Personnel; Qualifications of Developer's Employees*

8. *Training; Conversion*

8.1 Developer to provide training consultation to User; User's manuals to be provided.

8.2 Developer to assist User in conversion of User's data base.

8.3 Costs.

9. *Warranties; Representations of Developer*

9.1 That Software System shall operate in accordance with Design Specifications for — months from Final Acceptance; requirement of investigation upon notice of deficiencies by User.

9.1.1 If no deficiencies actually exist — User shall reimburse developer for costs of investigation.

9.1.2 If deficiency exists, Developer will correct at its expense and User will acknowledge acceptance or rejection of such corrections.

9.1.3 After — month period, maintenance will be on time and materials basis; in the event other than Developer's personnel modify or repair the system before the end of the — month period, the warranty shall terminate and acceptance will be presumed.

9.2 Developer cannot subcontract its obligations.

9.3 No liens or encumbrances on Software System.

9.4 Corporate authority to enter into Agreement.

9.5 Exclusion of all other warranties; express or implied, including warranties of merchantability and fitness for a particular purpose.

10. *Proprietary Rights*

10.1 Software System and all related information is sole and exclusive property of Developer.

10.2 User's right to request copies of materials.

10.3 Confidentiality of proprietary and other informational materials of User and Developer.

10.4 Confidentiality of Developer's obligations under Agreement.

11. *Participation of User's Personnel*
User's restricted right to monitor Developer's performance of obligations.
12. *Default*
12.1 (i) Developer's failure to perform obligations.
(ii) Developer's acts of bankruptcy, etc.
12.2 Remedies
13. *Indemnity*
Patent, Copyright, etc.
14. *Additional Charges*
Out of pocket expenses and taxes to be included in invoices.
15. *Termination*
User's right to terminate with — days notice to Developer up to final acceptance date. Charges to then be based on time and materials.
16. *Limitations of Liability*
16.1 Maximum amount.
16.2 No indicated or consequential damages.
17. *Governing Law; Disputes*
18. *Notices*
19. *Entire Agreement*
Merger of prior agreements; modification of agreement.
20. *Captions; Counterparts*
21. *Assignment*
22. *Miscellaneous*
22.1 Waivers.
22.2 Costs of preparation of Agreement.
22.3 Developer as independent contractor; no third party beneficiary.
22.4 Severability of Agreement.
22.5 Delays due to Acts of God, etc.
22.6 Effective date of Agreement upon Developer's acceptance.

POTENTIAL EXHIBIT LIST

A. Hardware System
B. Proposal
C. Sequence of Tasks; Task completion Time; Project Completion Date; Task completion Costs
D. Modification/Change Request
E. Deliverable Items

APPENDIX A

APPLICABILITY AND NATURE OF UCC WARRANTIES

Although it is unclear whether software license arrangements are subject to the UCC, both the user and the vendor in a given transaction should be concerned with the implied warranties of merchantability and fitness for a particular purpose which accompany a "sale of goods" covered by the UCC.[64] The parties may even agree contractually to have the UCC apply to their transaction.

Before examining specific warranties found in software licensing agreements, it is important to discuss the question of whether the UCC is applicable to software licensing transactions. If the UCC applies to software licensing transactions, any disclaimers of warranties included in the licensing agreement are governed by UCC Article 2[65] and user may have available to it all the warranties and remedies included therein.[66] UCC Article 2 applies to the sale of *goods* and the courts have not yet determined whether software is considered a good for UCC purposes. Since software consists of a series of instructions, it may be viewed as intangible and, therefore, not subject to the UCC. When software is purchased in conjunction with hardware equipment, however, the courts generally view the acquisition in its entirety as an acquisition of a tangible system and find that UCC Article 2 governs the entire transaction.[67]

Since the UCC applies exclusively to the sale of goods, software licensing transactions may not be within the code because, in such transactions, a user generally acquires only the right to use the software while actual ownership of the software is retained by the vendor. Nevertheless, perpetual licenses more closely resemble a sale than do short term licenses and are therefore more likely to be subject to the UCC. Perpetual licenses are analogous to full pay-out leases which are more likely to be subject to the UCC than short term oper-

---

64. U.C.C. §§ 2-314, 2-315.

65. U.C.C. §§ 2-314, 2-316.

66. Such warranties include warranties of merchantability and fitness for a particular purpose. *See* U.C.C. §§ 2-314, 2-315.

67. *See* Triangle Underwriters, Inc. v. Honeywell, Inc., 457 F.Supp. 765, 769 (E.D.N.Y. 1978), *aff'd in part, rev'd in part, and remanded*, 604 F.2d 737 (2nd Cir. 1979); Chatlos Systems, Inc. v. Natl. Cash Register Corp., 479 F.Supp. 738,742 (D.N.J. 1979), *aff'd in part, remanded on other grounds*, 635 F.2d 1081 (3rd Cir. 1980) *cert. denied* 457 U.S. 1112; *See generally*, Note, *Computer Programs as Goods Under the U.C.C.*, 77 MICH. L. REV. 1149 (1979). Similarly, U.C.C. Article 2 applies to transactions in which hardware price includes programming services. *See* Carl Beasley Ford, Inc. v. Burroughs Corp., 361 F.Supp. 325, 334 (E.D. Pa. 1973), *aff'd*, 493 F.2d 1400 (3rd Cir. 1974).

ating leases.[68] For example, where user leases a system from either the manufacturer or a leasing company, the court must first determine whether the leasing arrangement is a "true lease" or a "sale" of equipment with the lessor receiving a security interest.[69] If the lease is a true lease, the rental charges are set at a specific rate to compensate lessor for the loss of the value of the product's use over the term of the lease. Furthermore, title to the equipment remains with the lessor and any alleged malfunctions can be asserted as failure of consideration and a defense to further lease payments. Thus, user's remedy against lessor is to withhold lease payments. This transaction would probably not be governed by the UCC.

On the other hand, if the lease is actually a sale with reservation of a security interest, total rental payments are approximately equal to lessor's purchase price. In addition, title to the equipment passes to user and UCC Article 2 and Article 9 apply. Thus, any disclaimers of warranties included in the leasing agreements are governed by UCC Article 2 and user has available all the remedies included therein. Such lease agreements usually contains a "hell or high water" clause making payments unconditional so that user may continue making lease payments regardless of a system breakdown.[70]

In conclusion, arguing that a licensing transaction constitutes a "sale" for UCC purposes may require one to analogize licensing transactions to leasing transactions. The parties should beware that even if the UCC is not applicable as a matter of law, the court may consider the UCC by analogy.[71]

---

68. Gilburne, *Licensing of Pre-existing Software Packages*, in Illinois Institute for Continuing Legal Education Seminar on Contracting for Computers and Related Products and Services.

69. *See* Citicorp Leasing, Inc. v. Allied Institutional Distributors, Inc., 454 F.Supp. 511, 513-14 (W.D. Okla. 1977).

70. *Id.* at 514. In this situation, the user may have a remedy under the UCC for breach of warranty, although the lessor usually disclaims all warranties. Therefore, the user's only remedy may be against the manufacturer.

71. *See* BERNACCHI & LARSEN, DATA PROCESSING CONTRACTS AND THE LAW, 138-139 (1974).

APPENDIX B

SELECTED TAX ISSUES ASSOCIATED WITH DEVELOPMENT AND LICENSING

I. SALES AND USE TAXES

While software is considered intangible for purposes of the ITC at the federal tax level, approximately 36 states currently consider software to be tangible and consequently subject to state sales and use taxes.[72] These states consider software to be a tangible item because vendors usually transport their product on some tangible media such as punched cards, tapes, or computer discs. Thus, the vendor's machine-ready software, which is not produced in a machine-ready format, may escape such taxes.[73]

To determine whether a particular form of software should be subject to state sales and use taxes, recent court decisions have applied the "essence of the transaction" test.[74] For instance, in a recent Illinois case in the sale of computer software was held to be a transfer of intangible personal property exempt from Illinois use taxes because the substance of the purchase was the information contained on computer tapes rather than the tapes themselves.[75]

Not suprisingly, vendors and purchasers of computer software have been shocked when confronted with sales and use taxes on supposed "tangible" software products, especially after considering that the same software is ineligible for the ITC because it is considered "intangible" property. Since these taxes may be substantial, software vendors and purchasers are somewhat concerned about the inequities between the federal and state classification of software products.[76]

In the software licensing scenario, it is less clear than in the software purchase scenario whether the licensing arrangement is subject to state sales and use taxes.[77]

II. RESEARCH AND DEVELOPMENT TAX CREDITS

The Economic Recovery Tax Act of 1981 may provide a direct

---

72. *See* White & Vanecek, *Taxpayer Beware: The Current State of Computer Software Taxation*, 60 TAXES 373 (1982).

73. *Id.* at 374.

74. *Id.* at 375-76.

75. *See* First Nat'l Bank of Springfield v. Dept. of Revenue, 85 Ill.2d 84, 91, 421 N.E.2d 175, 179 (1981).

76. White & Vanecek *Supra* note 51 at 373.

77. For a state survey of sales, use, and property taxes as they relate to data processing products, see Bigelow & Saltzberg, *State Computer Tax Report* (1982).

tax credit for developers and purchasers of computer software. IRS Section 44F was designed to stimulate technology research activities. It offers taxpayers a tax credit equal to twenty five percent (25%) of any incremental increase in research and experimental expenditures during a statutorily defined base period. The Internal Revenue service has recently issued proposed regulations covering various points concerning the credit. As the credit applies to software development costs, Code Section 44F refers to Code Section 174 (covering expensing research and experimental costs) for definition of qualified research. The regulations under Section 174 are currently being revised to clarify the treatment of computer software. An explanation of the proposed changes in the treatment of computer software follows:

> "Under proposed regs, the cost of developing computer software is not a research or experimental expenditure if the software's operational feasibility is not seriously in doubt. Likewise, the costs of modifying previously developed computer software programs, such as the costs of adapting an existing program to specific customer needs, or the costs of translating an existing program for use with other equipment, do not constitute research or experimental expenditures. But the programming costs for new or significantly improved computer software qualify. The determination of "new or significantly improved" will be based on the computer program itself rather than the end use of the program. For example, the costs of developing a program to perform economic analysis which involves only standard or well known programming techniques are not research or experimental expenditures even if the economic principles embodied in the program are novel. However, if the programming itself involves a significant risk that it cannot be written, the cost of developing the program are research or experimental expenditures regardless of whether the economic principles or formulas embodied in the programs are novel (Prop. Reg. Section 1.174-2(a)(3))." Most users and developers of software who have made their views known feel that these rules are both too exclusive and not specific enough to make a clear analysis whether a development project is or is not qualified for the tax benefits offered by the Code. Whether this confusion will be cleared up after the IRS' hearings which were held in March, it is too early to predict.[78]

78. *Weekly Alert* of Research Institute of America's *Federal Tax Coordinator*, issued January 27, 1983.