JOURNAL OF ORGANIZATIONAL COMPUTING

Volume 3 **Number 3, 1993**

SPECIAL ISSUE:
Economics, Information Systems, and Organizations

Guest Editors:
Erik Brynjolfsson and Haim Mendelson

JOURNAL OF ORGANIZATIONAL COMPUTING, 3(3), 329–361 (1993)

Hierarchical Elements in Software Contracts

Soon Ang
Nanyang Technological University, Singapore

Cynthia Mathis Beath
Southern Methodist University

Recent literature in information systems notes that software development outsourcing is increasingly prevalent, despite the complexity of managing development across organizational boundaries. Information systems researchers have used transaction cost and agency theories to propose incentive schemes to address this problem. Drawing on legal and organizational theories about contractual relations between firms, this article describes and illustrates a set of contractual elements, essentially hierarchical control mechanisms, that can contribute to the governance of external software development. Software outsourcing contracts using such elements should be viewed as hierarchical, rather than market, organizational forms, in that they are sheltered from the disciplining influence of market forces. Following transaction cost theory, the article proposes that the use of hierarchical elements will vary with transaction characteristics. Actual software contracts are content analyzed to lend empirical support to the propositions. Future research directions and content-analytic research designs appropriate for analyzing software contracts are then elaborated.

software development, relational contracting, outsourcing

1. INTRODUCTION

According to the markets and hierarchies paradigm, a transaction may be governed by an interfirm market relationship or by hierarchical administration within a firm [1]. Given no substantial differences in production costs, the choice of market or hierarchical governance turns on the transaction costs inherent in coordination [2]. In effect, whenever requirement specifications are difficult to determine in advance, whenever costs, prices, or quantities are uncertain,

The authors wish to thank F. Hiew for his valuable contribution and assistance in this research. We also deeply appreciate the suggestions and comments received from Edward Adams, Uday M. Apte, Larry L. Cummings, Lisa Downs, Scott Masten, Steve Nickles, the special issue editors, and two anonymous reviewers.

Correspondence and requests for reprints should be sent to Cynthia Mathis Beath, Edwin L. Cox School of Business, Southern Methodist University, Dallas, TX 75275–0333.

whenever specific assets are required [1], or whenever separate measurement of performance is obscured because of team interdependencies [3], hierarchical administration is advocated.

Nevertheless, as Stinchcombe [4] observes, market contracts are sometimes signed for complex and uncertain transactions in which many of these conditions would be expected to exist. Using examples in research and development, automobile manufacturing, and weapons procurement, he concludes:

> Although research and development in commercial life is ordinarily carried out by a subordinate R&D staff . . ., the government buys weapons R&D by contracts under the same conditions of uncertainty of performances. Uncertainty about costs, prices, and quantities frequently leads to vertical integration as Thompson predicts [5], but automobile franchises and weapons procurement often involve contracts for shifting quantities, uncertain costs, and prices to be determined. . . . (pp. 121–122)

If we apply transaction cost logic to the case of software development, we might predict that many information systems would be developed internally, because of inherent uncertainties in requirement or cost estimation, performance unobservability, or the presence of transaction-specific investments on the part of clients and contractors. However, software development has been and continues to be outsourced [6–8], presumably to capture benefits of specialization or certain distinctive competencies held by contractors.

According to transaction cost logic, this outsourcing of software development should lead to problems associated with opportunism, excessive coordination costs, or even transaction failure in some cases. In the information systems literature, proposals have been made to reduce these risks by structuring contracts that align the incentives of the contracting parties [9,10]. These proposals determine *ex ante* incentive structures for contracts that are essentially contingent claims contracts or a series of such contracts. However, the complexities of software development may make the implementation of these proposals problematic.

An alternative approach is to plan more flexibility into contracts by incorporating elements common to hierarchical governance, as suggested by Stinchcombe [4, p. 124]. The thesis of this article is that contracts for software development outsourcing should not necessarily be viewed as market governance mechanisms. Rather, drawing on concepts in legal and organizational theory concerning contractual relations between firms, the article argues that software contracts can embody elements of hierarchical governance and can thereby afford the contracting parties considerable latitude in governing software development transactions within a market context. This article poses two questions: (1) What hierarchical elements can be included in software development contracts? and (2) What determines variations in these contracts? Answers to these questions can help information systems researchers theorize more precisely about the management and control of software development outsourcing. The research also has implications for practitioners, as they draft and negotiate contracts for external software development.

The article proceeds as follows: In Section 2 we describe briefly how contracting complications are precipitated by uncertainties and asset specificities in software development. In Section 3, we develop the concept of the relational contract and show how contracts designed to sustain a relationship between business partners accommodate uncertainties and asset specifities. Then in Section 4, we describe in greater detail hierarchical elements that may be incorporated into software contracts. The rest of the article deals with the determinants of contractual varieties. Section 5 presents propositions on software contract variation and describes an empirical study testing these propositions. Section 6 contains conclusions and suggestions for future research. Specifically, additional propositions about contract variation and research strategies using content-analytic techniques for testing these propositions are elaborated.

2. NATURE OF SOFTWARE DEVELOPMENT

From the perspective of classical contract law, a desirable contract is "sharp in by clear agreement; sharp out by clear performance" [11, p. 738]. Such contracts specify exactly how much of what is to be delivered by each party, and they are concluded with similar precision. The theoretical cornerstones of these contracts are "discreteness" and "presentiation" [12]. "Discreteness" requires contracts to ignore past and future exchanges among the parties. It is assumed that no duties exist between the parties prior to contract formation or subsequent to contract completion [13, p. 46]. "Presentiation" requires performance contingencies to be fully determined at the inception of any contractual agreement. Macneil [12] elaborates:

> Presentiation is a way of looking at things in which a person perceives the effect of the future on the present. It is a recognition that the future has been brought effectively into the present so that it may be dealt with just as if it were in fact the present. Thus, the presentiation of a transaction involves restricting its expected future effects to those defined in the present. (p. 863)

In the economic and industrial organization literature, the conceptual counterpart to the contingent claims contract is market contracting or market governance [14]. The notion is that if contracts are discrete and presentiated, they can be subjected to market forces of competitive bidding and market control. According to Williamson [15], in cases where transactions recur frequently, are uncertain, or require substantial transaction-specific investments, classical or contingent claims contracts offer inadequate transaction governance. We consider in this section, therefore, the extent to which software development is likely to be repetitive, uncertain, or dependent on transaction-specific investments.

2.1 Frequency of Development of a Software System

Since firms rarely develop or make recurring purchases of the same software, acquiring software from contractors seems a logical choice. Outsourcing software development seems especially appropriate when external information

technology (IT) service providers are more likely than individual firms to reap economies of scale via specialization [16,17], or to possess special skills, knowledge, or technology for developing a particular software system.

2.2 Uncertainties in Systems Development

In many cases, information systems are innovations, and by their very nature, innovations embody specification uncertainties [18]. These uncertainties often arise because software development is more exploratory than definitive in nature; the client understands the possibilities of a system only as development progresses [9, p. 308]. As Richmond et al. [10] point out, many contractors do not really know what they have been asked to take on at the outset of software development projects since clients only know what they want when they actually "see" the system. Rapid technological developments also make initial specifications archaic [17]. With specifications highly uncertain, development cost estimation becomes problematic. Recent research shows neither algorithmic models nor human judgment consistently perform well in software cost estimation [19,20].

Furthermore, the combination of specification and cost uncertainties makes it difficult to predict what performance will be desirable from contractors [21]. Even when the desired performance is perfectly predictable, actual performance may be costly or difficult to measure. In the case of software development, physical estrangement of the contractor is common,and clients may have too little expertise to evaluate the development process or the quality of the delivered software [21]. Overall, contingent claims contracting for software development in the face of these uncertainties is likely to be very problematic.

2.3 Transaction-Specific Investments in Systems Development

Both the client and the contractor sometimes are required to invest in specific assets prior to or during the course of a software development project. For the client, necessary transaction-specific investments can include effort invested in teaching a particular contractor exactly what is desired of the software or the project (critical aspects of which often defy reduction to a written specification); effort invested in learning idiosyncratic aspects of coordinating with or communicating with a contractor (e.g., learning their lingo or techniques); or effort invested in learning how to monitor the performance of a particular contractor. Of course, the less straightforward the software requirements are, the greater these investments are likely to be, and the less likely they will be transferable to another contractor. In addition, a client's post-implementation adoption investments (e.g., training or changes in organization structure or incentive systems) may only be transferable to alternative software within limits. This dependency may expose the client to possible opportunistic appropriation of rents by the contractor during maintenance and enhancement phases. This vulnerability heightens if such systems are long-lived.

Contractors often make some transaction- or client-specific investments prior to contracting with a client, in particular to develop a client relationship, acquire client-specific domain knowledge, or even to negotiate the contract itself. Or, a contractor may underbid a contract, in effect investing in future contracts for, say, enhancements or additional software. Contractors may thus become reliant on a client for subsequent businesses in maintenance and enhancements. The dependence worsens when the client is one of the few major clients of the contractor or otherwise in a monopsony relationship to the supplier.

2.4 Summary

Software development frequently takes place under conditions of shifting client specifications, and cost and performance uncertainties. Because uncertainty prevents the parties from specifying complete contracts, it generates pressures to utilize internal hierarchical governance mechanisms. Moreover, when specifications are very complex or uncertain, clients are more likely to make significant irretrievable, transaction-specific investments during the course of a development project, in their attempts to communicate their needs to the contractor. Lengthy system lives further expose clients to post-implementation expropriation of rents from contractors. These exposures discourage the use of external sourcing and are often cited as reasons for the maintenance of an internal development staff [10; 22, p. 77]. In Section 3 we show how contingent claims contracts can include hierarchical elements, so that contracts can be drawn up between business parties who desire to establish and sustain ongoing business relations involving transactions with high uncertainty and significant specific asset investments.

3. EXPANDING THE NOTION OF CONTRACTS: CONTRACTS FOR ONGOING RELATIONSHIPS

3.1 Contingent Claims Contracts

In Section 2, we noted that the paradigmatic contract is a presentiated, discrete contract conveying a well-defined object in exchange for specific consideration [13]. In fact, this common definition of a contract is given in the *Restatement (Second) of Contracts*:

> A contract is a promise or a set of promises for the breach of which the law gives a remedy, or the performance of which the law in some way recognizes as a duty. [23, Section 1]

Contractors accommodate future uncertainty, to some extent, by stipulating contingent claims. However, once a contingent claims contract is entered, the

obligations of the parties are prescribed for the duration of the agreement. Overall, the incentives in the contract are definitive, and the agreement is not subject to further negotiation. Underscoring these considerations is the presumption that the courts will either direct specific performance or apply appropriate damages to assure that the intentions of the parties are fulfilled [14]. Thus when a contract is contested, the courts bring the contract, and its relationship, to a close.

Although this may be suitable for some, not every market transaction fits comfortably into the contingent claims scheme. First, as Macaulay [24,25] observes, for contracts executed under conditions of extreme *ex post* uncertainties, complete presentiation is difficult if not impossible. Second, there exists a large set of contracts for which the discrete transactional assumption is inappropriate. Buyer–supplier relationships often comprise a continuing set of exchanges each of which is not easily separated from the others [13,24,25]. Third, the disciplining effect of markets may be eluded if the system of social relations in which business contracts are embedded influences the contract or its execution. Institutional contexts, social norms, or business cultures may cause clients to limit the number of contract bidders or to accept some contract provisions regardless of their rationality [26].

According to Williamson [15], faced with the prospective breakdown of contingent claims contracting, firms have three alternatives. One would be to forgo such transactions altogether. A second option would be to remove those transactions from the market and organize them internally. Instead of frequent cycles of reneging on the original contract and renegotiating new ones, firms are then assumed to replace a market system of transactions with nonmarket organizational forms. Adaptive, sequential decision making would then be implemented under unified ownership, backed by the motivating force of hierarchical incentive and control systems. The third option would be to devise a contracting relation that provided for an ongoing process of negotiations and renegotiations over the terms of trade. To sustain the trading relationships and foster cooperation, business partners would incorporate flexibility into their contracts. This last alternative brings us to relational contracts. Similar to Williamson, we contend herein that firms need not abandon interfirm contracting to achieve sufficient flexibility to manage uncertainty and opportunism.

3.2 Relational Contracts

According to Goetz and Scott [27, p. 1091], a contract is relational to the extent that it embodies clauses in which important terms of the arrangement are not reduced to well-defined obligations. Definitive presentiation is sometimes simply impractical. Moreover, in relational contracts, discreteness is also not the objective; rather, emphasis is on "a more general statement of the process of adjusting the terms of the agreement over time—the establishment, in effect, of a 'constitution' governing the ongoing relationship . . ." [13, p. 428]. Instead of an arm's-length, fully specified, discrete exchange, a relational contract creates a minisociety with an array of norms beyond those created by legal and market

systems [12,28]. In relational contracts, parties do not agree on detailed plans of action but, rather, on goals and objectives, on general provisions that are broadly applicable, on criteria to be used in deciding what to do when unforeseen contingencies arise, on who has what power to act and the bounds limiting the range of actions that can be taken, and on informal dispute resolution mechanisms to be used if disagreements do occur [22, p. 131]. Discreteness and presentiation become factors that guide decisions during contract execution, not defining criteria of the contract.

A relational contract can ameliorate some of the difficulties inherent in ordinary contingent claims contracts by adding more flexible contract-term adjustment mechanisms—essentially hierarchical governance mechanisms—to contingent claims contracts. For example, the sequential and adaptive pattern of agreeing on process and procedure that is characteristic of relational contracts is also typical of hierarchies [22, p.132]. Another hierarchical device found in relational contracts is domains of authority. Once established, these can be used to defer decisions about price or other aspects of the exchange until well into the execution of the agreement. The trade-off is that market pressures on the terms of the contract are lost or diminished; market forces cannot be brought to bear on the terms of deferred decisions.

3.3 Summary

The relational contract approach provides a very different perspective for theorizing about governance for software development. Information systems theorists should not assume that software contracts represent discrete, presentiated contracts subject to the careful discipline of competitive market forces. As Granovetter [26] argues, ". . . [for transactions in] imperfectly competitive markets characterized by small numbers of participants with sunk costs and 'specific human capital' investments, . . . the discipline of competitive markets cannot be called on to mitigate deceit" (p. 488). Perhaps of greater importance, outsourcing of software development need not mean the loss of familiar, efficient, and powerful hierarchical control mechanisms. We contend that incorporating hierarchical elements into contracts increases the feasibility of the transaction. Five types of these hierarchical elements are elaborated in greater detail in the next section.

4. HIERARCHICAL ELEMENTS IN SOFTWARE CONTRACTS

According to Stinchcombe [4], the control functions afforded by hierarchies may be characterized as follows: (1) command structures and authority systems, (2) rule-based incentive systems, (3) standard operating procedures that stipulate behavior, (4) nonmarket pricing systems, and (5) informal or extralegal dispute resolution. Contracts emulate hierarchies to the extent that they incorporate these elements. Each of these elements is elaborated in the subsections below. Where appropriate, sample contractual provisions are provided, drawn from

prescribed software development contracts published in a variety of software legal handbooks and software management publications [29–38].

4.1 Command Structures and Authority Systems

Drawing on Barnard's [39] concept of authority systems, Stinchcombe [4] defines authority systems as "systems by which flows of information are certified as legitimate or authoritative, so that a person who acts in accordance with them has the risk of being wrong removed from him or her, and laid on the legitimators of the communication" (p. 156). Contractual terms that certify given orders or communications as authoritative constitute command structures or systems of hierarchical referral [40]. In relational contracts, authorizing elements identify who will decide when unforeseen contingencies arise. Software contracts can have provisions that authorize or assign people—either the contractor or the client or both—the right to make discretionary decisions, issue orders, or demand performance. A sample provision that assigns the client constrained authority for making choices on a software development project reads:

> Customer will have responsibility for all management decisions and general control of the Project, provided that Customer shall take no action nor allow any action to be taken that will inhibit or delay the performance required by Company hereunder. [31, p. 472]

Authorizing elements may give identified personnel the power to change the project scope without reneging on or breaching the contract. This is particularly germane in the software development context, as the products or services required by the client often evolve in the process of software development and implementation. Here is a clause that gives the client the power to approve contractor changes to specifications or contract price:

> Modifications. Any modifications to the present Agreement and its effect on the schedule, fees and other terms and conditions . . . shall be approved in writing by the Client prior to its inception and execution. [31, p. 289]

The following clause gives the contractor authority to approve changes in terms:

> If at any time Customer fails to fulfill Customer responsibilities in a timely and accurate manner, Softco reserves the right to stop work and renegotiate the price and/or terms of performance and, if no agreement is reached within a period of two (2) weeks to bill Customer, on a time and material basis, for custom software and training efforts to date. [37, Form 3.20-3]

And here is a clause giving change authority to both client and contractor:

> Any changes in the functional specification must be specifically approved in writing by both Customer's project manager and vendor's Vice President of programming. [37, pp. 4.1-6, 4.1-7]

Sometimes, authorization for change can be very detailed. For example, one recommended modifications clause runs into six paragraphs, totaling well over 350 words [30, pp. 983–984]. The clause required that the client submit all requests for additional services to the contractor in a formal "modification/change" request report. Upon receipt of the request, the contractor would evaluate each request and respond in no later than 10 working days. The written response was required to be detailed about the availability of contractor's personnel and resources to perform the modification, the completion date, and any changes in costs. With the written response, the client could elect to authorize the changes, in writing. If he or she did so, the changes were to be deemed incorporated into and part of the original agreement.

Other examples of authorizing elements include (a) giving the client the right to audit the work-in-progress (rather than just final performance), establishing a basis for some measure of supervisorial oversight:

> Right to Audit. The Client has the right to audit the work performed by ABC under this Agreement and to make recommendations for product and quality improvement [31, p. 289];

(b) specifying a specific person in the client organization to have authority over price adjustments on the project, opening a door for such price adjustments:

> Vendor acknowledges that vendor must complete the specified work for the stated price and no adjustment shall be made authorizing increased payments except in writing signed by a Vice President of Customer [37, p. 4.1-6];

(c) permitting the client to choose and change personnel from the contractor firm, a privilege generally restricted to hierarchical governance:

> Project Leader. Vendor agrees to provide the services of _____ to personally supervise all of Vendor's work on the System and to prepare the Design Document. All other personnel assigned by Vendor to work on the System shall be pre-approved by User and shall have all necessary and application skills and experience and education and shall be full-time employees of Vendor. Vendor shall replace any personnel User finds to be unfit or unsatisfactory for any reason [38, p. 174];

(d) giving clients the right not only to make discretionary decisions but also to cancel the project at specified points in the development, perhaps to avoid additional unproductive investments:

> The user may, at its option, elect to cancel the contract at any time, by notice to vendor, upon completion of any stage described in Schedule A. . . . In such event the user will pay to the vendor the amount due by virtue of completion of the products therefore delivered [33, p. 379];

and (e) giving the client the right to approach contractor staff in the event of business termination on the part of the contractor. Clauses such as this one may

be appropriate when the client has made a significant investment in contractor knowledge about the client's problem domain or solution:

> In the event the vendor shall cease conducting business, the user shall have the right to offer employment, on a permanent or part time basis, to all employees of the vendor assigned to the performance of this contract, notwithstanding anything to the contrary provided elsewhere herein. [33, p. 382]

Finally, authorizing clauses sometimes give authority to parties beyond the immediate contracting parties. For instance, in the following agreement between GTC (the contractor) and HCMC (the client), HCMC's information services group is given the authority to approve the purchase of hardware equipment required of the software that is being developed:

> i) GTC [contractor] must request in writing HCMC's [client's] authorization to purchase the equipment, and must include in the request GTC's [contractor's] justification for the purchase, ii) HCMC Information Services must authorize the purchase in writing, and iii) the equipment must conform to HCMC Information Services' current computer equipment and software standards. [30, p. 984]

Authorizing clauses substitute for predefinition of contingencies and contingent action. Rather than anticipating the unanticipatable, they establish a process for dealing with it.

4.2 Rule-based Incentive Systems

An incentive system is a system of rewards and punishments tied to behavior or outcomes. Stinchcombe [4] defines a hierarchical incentive system as one that allocates "differential compensations based on the level of performance, without further recourse directly to the market. A [rule-based] incentive system then is an enclave in the market within which special rewards and punishment apply" (p. 159). In contrast, a market incentive system would reflect differences in marginal revenue productivity. Ouchi [40] notes that in hierarchies compensation is often used as an inducement for future performance rather than a reward for past performance. As a result the association between compensation and past performance in hierarchies can be rather weak.

A market incentive system works well only under conditions of certainty or complete contracting where all performance contingencies are considered *ex ante*. Rule-based incentive systems serve to dissociate compensation and past performance. They reflect locally determined, as opposed to market determined, inducements for desirable future performance.

For example, if timely delivery is vital in software development, penalties for delays beyond an agreed completion date and rewards or bonus for early completion may be incorporated into the contract [33, p. 323]. The penalty rates are usually determined locally, with little reliance on market equivalents. Here are two provisions displaying rule-based incentive structures for penalizing late delivery on the part of the contractor:

> . . . should Vendor fail to complete the entire project by the final installation date specified in paragraph three above due to circumstances within his control, Vendor hereby agrees to reduce the purchase price of the system at the rate of one hundred dollars ($100) per week delay in exchange for Vendee's mutual promise to forfeit any legal right it may have to sue for any other damages direct, consequential, or otherwise. [29, p. 198]

> In the event of a delay in delivery, . . ., the vendor shall pay to the user the sum of $_____ for each day of delay in delivery as agreed liquidated damages. [33, p. 323]

As systems evolve, sometimes it is difficult, if not impossible, to decide which objective (cost containment, timely delivery, meeting user requirements, adherence to MIS standards, etc.) is the most critical. A strongly hierarchical incentive system would include contractual terms in which the client reserved the right to change the incentive structure as performance objectives changed or as more information about the project's costs or benefits became available. These clauses have the effect of postponing compensation and creating compensation (or punishment) schemes according to the business partners' own rules rather than resorting to market discipline.

4.3 Standard Operating Procedures

Standard operating procedures are routines established by the client referencing or describing specific, well-understood actions to be followed by the contractor or features to be included in the product. They are a basis for establishing behavior or outcome control measures [41,42]. Typically, hierarchies make greater use of behavioral control mechanisms, whereas market governance utilizes outcome control mechanisms, by prespecifying desired product or service outcomes. Hence, we classify standard operating procedures that specify behaviors to be exhibited by the contractors as hierarchical control mechanisms (e.g., the software will be developed using certain procedures), and those that describe characteristics of the software or system product (e.g., the software itself will conform to established government standards) as market control mechanisms. Behavior standards facilitate monitoring and reduce uncertainties arising from performance unobservability. They can constrain opportunistic behavior on the part of the contractor and give the client confidence that the contractor is acting in the client's interests.

The archetype of a behavior standard is this provision requiring the contractor to produce formal progress reports for the client:

> Progress Reports and Meetings. Vendor shall submit monthly written progress reports to User on the first of each month. Each report shall describe the work performed since the preceding report, including, but not limited to, the development, testing, and installation status of each function of the System, the names of each employee then working on the project, and a description of the work performed by each listed employee during the report period. The report shall also describe the work expected to be performed during the month following the

> report. Any uncorrected delays or problems during the report period and their causes shall also be set forth. Progress meetings shall be held in person or by conference call on Monday of each week (unless rescheduled by agreement) and minutes thereof prepared by User and circulated to Vendor. [38, p. 174]

Armed with this information, the client might exercise behavioral control by invoking authorities provided elsewhere to speed up the project, adjust incentives, or change performance objectives.

Clauses outlining behavior standards can be very detailed. For example, one clause specifying the procedures for biweekly progress accountability ran for five detailed paragraphs, totaling more than 630 words [35, pp. 176–177]. In addition to detailing project status reports, the clauses required that both vendor and user project managers meet face-to-face to discuss the progress made by the vendor and the users in the performance of their respective obligations since their prior meeting. Problems and potential project delays were to be delineated explicitly and discussed. Alternative actions were then brainstormed, and the client was given the authority to direct the vendor to proceed with any of the alternative services or actions recommended by the vendor during the meeting. In ordinary contingent claims contracts, or market governance, progress reports such as these have little or no relevance, as there are no deferred decisions or opportunities to adjust terms during contract execution.

4.4 Nonmarket-based Pricing Systems

Software contract prices may incorporate prices based on cost-recovery, market prices established by bidding or benchmarking, or a combination of the two. Cost-recovery pricing is a feature of hierarchical governance [4]. In cost-recovery pricing, payment is based on the value of inputs. The norm in market prices, on the other hand, is that payments are in proportion to the value of outputs. Furthermore, market prices are disciplined by competitive forces. In cost-plus pricing, a combination of cost-recovery and market pricing, the reward for distinctive competence or efficiency of the contractor, the fee or profit, is separated from the costs incident upon the contractor due to changes in specification or other externalities.

When cost uncertainty is high, a cost-recovery pricing system may be adopted to remove the risks of uncertainty from the contractor. Software contracts based in some measure on cost recovery allow clients to modify requirements midstream without necessarily shifting the consequences of change, that is, the additional costs, onto the contractor. Clauses such as the following place all the risk of specification changes on the client:

> The price set forth on the face of this Agreement is the agreed upon price for only those Services described. Additional features or functions requested by the Customer may result in additional work to be performed by Softco, at the Customer's expense. All additional work performed for the Customer, at the Customer's request, shall be reimbursed by the Customer at Softco's standard hourly rates, or as provided in a change order accepted by Softco. Softco will on

request, provide estimates of the increased cost resulting from changes by the Customer in the requirements for any Services to be provided hereunder, or for additional features or functions requested by the Customer. [31, p. 295]

To mitigate price uncertainty, the client can insist on fixing part of the price for system delivery at the outset of the contract with incidental costs, such as additional contractor expenses, in the form of overtime, travel, reproduction, extra computer time, and so forth, to be approved as incurred. Such a combination of market and cost-recovery based system of contract pricing serves the dual role of reducing price and compensation uncertainties for both the client and the contractor:

The Client will pay ABC a fee as described in Schedule A (the "Fee"). . . . Expenses and costs incurred by ABC in the execution of this Agreement shall be preauthorized by the Client, invoiced at cost and supported by documentation. [31, p. 290]

We note that even where travel or other expense *rates* are subject to market discipline, the *quantity* of travel is unlimited by the preceding clauses. Therefore, a price ceiling or upper bound may be specified for fees and reimbursements. The primary benefit of a price ceiling or upper bounds is to lessen the contractor's opportunity to appropriate rents as the software development progresses. A sample clause reads:

(a) MMRF shall pay GTC fees in the amounts and at the dates set forth below:
. . .
(b) If the verifiable actual cost of developing the System exceeds $25,000, GTC shall invoice MMRF for half of such cost exceeding $25,000, up to a maximum of $18,000, upon MMRF's accepting of the System. . . . MMRF shall not be liable for such invoiced amount if it does not accept the system. [30, p. 982]

These clauses attempt to strike a reasonable balance between price risk for the client and compensation risk for the contractor, in a context of expected specification changes, to provide sufficient inducement for both the client and the contractor to enter into the contract.

4.5 Dispute Resolution Mechanisms

Conflicts may arise as a result of specification uncertainties, cost uncertainties, or performance unobservability, which the parties to a relational contract may wish to resolve in a manner that will permit the contract, and the business relationship, to continue. Whereas contingent claims contracts rely on legal and market sanctions for dispute resolution, hierarchical governance relies on resolution through hierarchical referral systems. In relational contracts, the emphasis is on surviving disputes, not just resolving them. This is accomplished by articulating dispute resolution procedures in some detail.

For example, a contract may specify that disagreements over the interpretation of terms be resolved by a sequence of private grievance procedures, then third-party arbitration, and concluding, as a last resort, with legal remedies. A sample provision reads:

> Dispute Resolution. In the event of any disagreement with respect to performance under this Agreement, the parties agree to first discuss the dispute informally at the Vendor Project Manager-User Data Processing Manager level. In the event that a resolution is not achieved at that level, the parties shall each designate one member of senior management to negotiate the dispute directly. In the event that such negotiation is not successful in achieving the resolution of the dispute, the parties agree to submit the dispute to nonbinding mediation by a mutually agreed-upon computer professional. The cost of retaining such professional shall be borne equally by the parties and shall be paid in advance. In the event that the recommendations of the professional are not accepted by both parties,the parties shall be free to pursue the remedies available to them, including, but not limited to, specific performance, it being the intent, however, that the foregoing procedures be used prior to the time that any litigation is commenced by either party against the other except when exigent circumstances exist. [38, pp. 179–180]

The contract stipulates a series of escalating opportunities during which disputes might be resolved informally. Then, by bringing up "specific performance," the contract reminds the parties that the consequence of failing to agree may be the termination of the business relationship.

4.6 Summary of Hierarchical Elements in Contracts

The premise of economic theory that firms coordinate internal behavior perfectly and engage only in discrete, anonymous contracts with external parties leads us to ignore the overlaps and similarities between internal and external coordination [43, p. 428, footnote 9]. The preceding analysis shows that certain contractual provisions can reproduce the effects of hierarchical control to mitigate contracting problems arising from uncertainties and specific assets. Hierarchical elements permit clients to issue orders and expect obedience from the contractor. Hierarchical elements also support incentive systems based on rules rather than market forces; they establish behavior standards to facilitate a regular flow of information between contractor and client showing that the contractor's actions, if not objectives, are congruent with those of the client; they protect contract prices from competitive pressures; and they establish local hierarchical referral systems for conflict resolution within the course of the contract.

Figure 1 elaborates the relationships between contingent claims contracts, relational contracts, employment contracts, contracts between legal firms, market governance, bilateral governance [15], and hierarchical or unified governance, in terms of the five hierarchical elements described earlier. It shows that relational contracts are a form of contingent claims contract [12], and they may be drawn between firms or within firms, where they are employment contracts

	Contingent Claims Contracts		
		Relational Contracts	
	Contracts Between Firms		Employment Contracts
	Market Governance	Bilateral Governance	Hierarchical Governance
Authority Relations		X	X
Incentive Systems:			
Fixed price	X	X	
Rule based		X	X
Standard Operating Procedures:			
Outcome standards	X	X	
Behavior standards		X	X
Pricing Systems:			
Market based	X	X	
Not market based		X	X
Dispute Resolution:			
Legal appeal	X	X	
Informal		X	X

Figure 1. Contracts and governance.

[27]. Although it has been said that a hierarchy can be viewed as a nexus of contracts [44], it might be more precise to say that a firm is a nexus of relational contracts. In employment contracts, supervisors implicitly hold the rights to oversee the work of employees, determine the basis on which they will be paid, change that basis periodically, change the person who does the overseeing, and own products and services produced by the employee. These supervisorial rights are not inherent in a client–contractor relationship but, rather, must be explicitly expressed in relational contracts, if they are desired.

5. DETERMINANTS OF CONTRACT VARIETY: PROPOSITIONS AND EMPIRICAL SUPPORT

5.1 Propositions

Contractual variety is the source of numerous puzzles in the study of economic institutions of capitalism [15, p. 68]. Transaction cost economists maintain that contractual variety is explained mainly by underlying differences in the attri-

butes of transactions. According to transaction cost analysis, efficiency purposes are served by matching governance structures to the attributes of transactions in a discriminating way. In this section,we contend that varying types and levels of uncertainties and asset specificities in software development projects also explain contract variation.

1) Software Development Uncertainties: Software development projects vary significantly in nature and characteristics. Projects can be differentiated by the degree of specification uncertainty, cost uncertainty, or performance measurement uncertainty inherent in them. Uncertainty complicates contingent claims contracting because it is impossible to write all contingencies into a contract, and "presentiation" is thus violated. However, flexibility can be accommodated in contingent claims contracts by using hierarchical elements. This leads to our first proposition:

Proposition 1

> The greater the level of uncertainty inherent in a software development project, the more hierarchical elements will be included in the contractual agreement between client and contractor.

Corollaries to this basic proposition may be derived if we consider, in more microanalysis, the relationship between *types* of uncertainties (specification requirements, cost uncertainties, performance unobservability) and *types* of hierarchical elements (authority systems, incentive systems, standard operating procedures, pricing system, and dispute resolution mechanisms). Certain types of hierarchical elements seem more suited to combat certain types of uncertainties. For example, with more uncertainty in specification requirements, we anticipate contracts will contain more elements describing authority systems and informal dispute processes. Moreover, since cost uncertainties are mitigated via the price system, we should expect that if cost uncertainty is very high, there will be a preponderance of rule-based incentive and price elements, rather than market-based ones. With respect to performance unobservability, we would expect performance uncertainty to be associated with the use of behavior standards and rule-based pricing elements in contracts.

2) Asset Specificities in Software Development: Software development projects also vary in the level of relationship-specific investments. Such investments may include investments in idiosyncratic learning, organization redesigns, or even specialized technology. Since these investments isolate the parties from other potential business partners and thereby raise the likelihood that requests for adaptations will result in costly opportunistic haggling, we would expect the following:

Proposition 2

> The greater the level of relationship-specific investments in a software development project,the more hierarchical elements will be designed into the contractual agreement between a client and the contractor.

Corollaries to this basic proposition may also be derived by differentiating specific assets associated with the software development stage of the life cycle from those associated with the operations and maintenance stage of the life cycle. During the development stage, both parties desire clauses that let them manage their exposure as decisions and choices unfold. A clause authorizing either party to initiate an orderly termination of the contract, without breach, reduces the probability of their being held up by a contract with escalating costs and commitments beyond the original agreement. Price ceilings or limitations on specification changes also limit opportunism. Clauses that make performance more observable may be tied to a desire to curb moral hazard.

During the maintenance and operations stage, however, the picture changes. The client has a significant sunk cost in an inoperable or inadequate system and may be dependent on the contractor for repair services. Clauses stipulating behavior standards would be very important in reassuring such a client that his or her interests were being protected by the contractor. Contractors, on the other hand, face salvage value problems. Whereas there might be learning benefits or reusable code salvaged from a development contract that went sour, the salvage value of enhancements is likely to be much lower for the contractor. Hence, for enhancement contracts, contractors will seek compensation protection.

We will use case studies of six software contracts to provide preliminary empirical validation of these propositions.

5.2 Research Design and Methods

The object of the empirical study is to challenge the existing assumption that software contracts are contracts that are discrete and presentiated in nature. From a content analysis of actual software contracts, we will show that outsourcing contracts for software development include flexible, open-ended, hierarchical clauses. Multiple cases are used here to demonstrate literal replication, that is, different cases and contracts provide similar results or evidence [45, p. 53]. A single-source content analysis design [46] was adopted with the selection of six contracts from two clients of a single software developing contractor. Thus, contracts are the sampling unit; they are analyzed at three levels: contract, business application (some contracts are for a single application), and firm.

1) Sampling of Contracts: Contracts were carefully selected to ensure variability in the antecedents (i.e., variability in software development uncertainties and transaction-specific assets) as well as variability in the consequences (i.e., the type and extent of hierarchical elements embedded in the software contracts). Six contracts were selected, two between the contractor and client Alpha, and four with client Beta. Figure 2 summarizes the differences among the client firms, the business applications, and the contracts.

2) Method of Data Gathering: Multiple sources of evidence were used for each case (contract) to provide greater confidence in the interpretations of the facts of each case. The written contracts themselves form the primary sources of evidence for the analysis. Contracts are frequently drafted by the contractor and

Figure 2. Data and results for the six contracts.

Firm	Alpha		Beta			
Business Application	Sales Forecasting		Sales Forecasting		Cost Analysis	Accounting
Name of Contract/Project	Gold	Opal	Silver	Ruby	Maple	Oak
Size of Contract (in units)*	224	5	105	25	41	375
Contract Length (word count)	3100	600	6000	2000	4000	4000
Uncertainties**						
Specification Uncertainty	H	L	M	M	L	L
Cost Uncertainty	M	M	M	M	M	M
Performance Unobservability	H	H	H	H	H	H
Asset Specificity***						
Client	H	M	H	M	M	M
Contractor	M	L	M	L	H	H
Hierarchical Elements						
Authority Relations	X		X	X		
Rule-based Incentive System		X				
Behavior Standards	X	X	X	X		
Non-market Pricing System	X	X	X	X	X	X
Informal Dispute Resolution			X		X	X

*Size of the contract is a linear transformation of the dollar value of the contract. Contract length is a rough approximation of the number of words in the contract, to the nearest 100 words.

**Assessment of the level of uncertainty for each of the three uncertainty components is based on phone interviews with the clients and the contractor. Questions included: (1) To what extent were both parties able to specify the system requirements at the outset of the contract? (2) To what extent were both parties able to estimate accurately the time and cost of development at the outset of the contract? (3) To what extent was the client able to observe the performance of the contractor during system development? Based on a comparison of the responses across the different projects to these questions, a level of uncertainty was assigned to each project. "H" represents high levels of uncertainty, "M" represents moderate levels, and "L" represents low levels.

***As with uncertainty, assessments of the level of asset specificity for client and contractor are based on phone interviews with the clients and the contractor. Questions asked of the client were: (1) How much effort and time did you invest in verbally conveying or clarifying systems requirements with the contractor? (2) How much time and effort did you invest in learning the unique systems development methodology used by the contractor? The greater the time and effort invested in the client in these activities, the greater the possibility of specific asset investment made by the client. The primary question on specific investments asked of the contractor was: To what extent can you transfer the knowledge learned from this project to projects with new clients? The lower the level of transferable knowledge, the greater the possibility of specific investment by the contractor. Based on a comparison of responses across different projects, a level of asset specificity was assigned. "H" represents high levels of asset specificity, "M" represents moderate levels, and "L" represents low levels.

then revised by the client. This procedure was followed with Alpha. However, Beta drafted their own initial contracts, after which the contractor requested modification.

In addition to the contracts, information was also gathered from several key informants. At the contractor firm, our informant was a project manager on the software development teams. This person was interviewed face-to-face and by phone in several interviews over a five-month period, lasting in total about 15 hours. At the client firms, two informants, end-user support managers, were interviewed twice by phone. The interviews averaged about 45 minutes. The contractor and client informants provided additional information about the drafting the contracts and interpretations of various clauses in the contract.

A third source of evidence included an internal memorandum and a formal letter of communication between Beta and the contractor. This documentary evidence was particularly useful for understanding the formation of the contracts with Beta as well as the evolution of the relationship between the contractor and the client. Figure 2 summarizes the data on transaction uncertainties and asset specificities as described by the key informants.

3) The Contractor: The contractor was founded in December 1988 with the goal of developing and implementing decision support application software and providing related consulting services to manufacturers and retailers. The business currently runs at about $4 million revenue with a 20% profitability rate. The major clients of the contractor are primarily Fortune 100 companies. The decision support systems developed by the contractor integrate market, financial, and internal company data from a variety of different sources to facilitate sales and marketing decisions. The contractor sells expertise in data analysis and applications development. Over time, the contractor has developed a set of standard applications products that form the foundational components on top of which custom development work is layered for customer-specific applications. At present the foundation accounts for about 60–80% of the code required for application contracts. Most system development work is performed at the contractor site, rather than on the clients' premises.

5.3 Alpha Contracts

Two contracts were drawn between Alpha and the contractor. The first, for the Gold project, was agreed upon in January 1991, and the second, for the Opal project, in August 1991. The Gold contract was for developing and implementing a sales forecasting system using the contractor's foundational software. The Opal contract was for developing a report module to enhance the sales forecasting system implemented under the Gold contract. The size of the Gold contract was about 224 units; the Opal contract was much smaller, about 5 units.

1) The Gold Contract: The Gold contract was the first business transaction between Alpha and the contractor. Specification uncertainties were high for this contract: Alpha had never had a forecasting decision support system before, and the system was one of the first decision support systems developed by the relatively new contractor.

Investments in specific assets in the Gold project by Alpha were relatively high. Alpha held lengthy discussions with the contractor to explain the forecasting models, statistical analysis, and complex mathematical algorithms they wanted incorporated into the forecasting system. These discussions began three months before the contract was started and continued sporadically during the six-month course of systems development. If the contract had not been completed successfully, Alpha would have had to start these discussions completely over with a new contractor. Alpha was also unfamiliar with the contractor's systems development methodology, which was based on an object-oriented approach. Consequently, in addition to educating the contractor on their forecasting practices, Alpha also had to make nontransferable investments in learning and adapting to the contractor's unique development approach.

From the contractor's point of view, knowledge gleaned from developing this system would be partially applicable to forecasting system projects for future clients. The contractor thus could have underbid this contract (although we do not know that they did this), investing in domain knowledge or the construction of reusable code. These investments would not be specific to client Alpha, however.

The Gold contract is relatively elaborate and embeds many hierarchical elements to cope with uncertainties and problems arising from asset specificities. For example, detailed roles and authority responsibilities involving the contractor, the Alpha end-user client, and Alpha information systems (IS) personnel were extensively delineated:

> The [Gold] project will involve several organizations including [contractor] Staff, [end-user] Staff, and [Alpha] IS staff. It is critical that all parties understand their duties in the project. Responsibilities of the group are described below:
> Responsibilities of [Contractor] Staff
> . . .
> Review and approve final [DB2 database schema]
> Review and approve [a capsule] created by [Alpha] IS Group
> . . .
> Responsibilities of [End-User] Client Staff
> . . .
> Review and approve functional specification for Phase I of [Gold] project.
> Review and approve detailed design specification for each module in Phase I.
> . . .
> Responsibilities of [Alpha] IS Staff
> Review and approve functional specification for Phase I.
> Review and approve detailed requirements for Phase I
> . . . (Document A.1, pp. 6–8)[1]

Because this was the first project undertaken by the contractor for Alpha, the contractor was not familiar with Alpha's existing technology. As a conse-

[1]Contracts are referenced by the following scheme: "Document A.1" means contract 1 for client Alpha, "Document B.2" means contract 2 for client Beta, etc.

quence of uncertainty about the technical feasibility of the system, the contract included clauses giving the contractor the authority to adjust schedule and price:

> Phase I of the [Gold] project rests on the following assumptions: The application will have access to the following SQL commands within DB2: . . . . If the above SQL commands are not made available for use on this project, [Contractor] reserves the right to adjust all development schedules and corresponding project costs.
>
> . . .
>
> [Contractor] will provide an initial database schema, and will negotiate a final database design with [Alpha] IS staff. If an agreed design cannot be jointly reached between [Contractor] and [Alpha] IS, [Contractor] reserves right to adjust all development schedules and corresponding project costs. (Document A.1, p. 8)

In addition, the implementation of the system depended on the successful acquisition of a software component by Alpha from another vendor. Uncertainty regarding this market acquisition led to a clause giving the contractor the right to modify the schedule:

> [Alpha] will purchase and install [SAS module] for DB2 option for use in this project. Specific installation dates will be negotiated . . . . (Document A.1, p. 9)

Because it was unclear how much training would be required, the contract limited the amount of training to be covered by the fee, and provided for training on a cost-recovery basis. Cost-recovery clauses such as these permitted Alpha to change and modify requirements midstream without necessarily shifting additional costs of change onto the contractor.

> At time of installation, [Contractor] will provide up to four additional days of on-site support to train [Gold] project administrator on use and administration of each module. [Contractor] will provide additional training as needed, on a per diem basis. (Document A.1, p. 9)

To cope with problems of performance unobservability (since the work was conducted for the most part at the contractor's offices) and to protect Alpha's unrecoverable investments in the software development project, clauses describing behavior standards such as the provision of progress reports and conducting of design reviews were included:

> [Contractor] will provide [Alpha] with weekly status reports outlining accomplishments, problems/issues, upcoming tasks and project resource requirements.
>
> . . .
>
> [Contractor] will conduct detailed design review meetings with [Alpha] representatives to obtain sign-off prior to development of each module of Gold project. . . . Specific schedules for these meetings will be agreed to by [Alpha] and [Contractor], as appropriate. (Document A.1, p. 9)

2) Gold Contract: Summary of the Evidence: Overall, the Gold contract contains many clauses that cope with high levels of specification uncertainty by identifying decisions to be made during the course of the contract and designating who would have the authority to make such decisions. To reduce the exposure created by Alpha's transaction-specific investments, the contract also gives Alpha many opportunities to monitor the project's progress. To compensate the contractor for the cost uncertainty this raises, the fee is partly set on a cost-recovery basis. (See also Figure 2.)

3) The Opal Contract: The Opal contract was for the development of a report module enhancement to the Gold project system. As with the Gold project, knowledge gained by the contractor from developing the system was partially transferable to other clients, but was of lesser value. A relatively small project (about one-twentieth the size of the Gold project in terms of monetary value), its requirements were relatively well-specified prior to the formation of the contract. However, despite a set of detailed functional specifications, the contract was negotiated on a "time and material" basis, with an agreed upon upper dollar limit:

> Services are provided on a time and material basis. Time: [Person X] @ [$Y] per hour, Material: as required and approved by [Alpha]. Travel: As required
> . . .
> Total fees for services, travel and lodging to be paid hereunder are estimated not to exceed [$K]. (Document A.2, p. 2)

Our interpretation is that the contractor was able to transfer some price risk to Alpha, in essence holding Alpha hostage [47] via the installed Gold system. The setting of an upper limit on costs (a rule-based incentive system feature) protects Alpha somewhat, as does the inclusion of requirements for product reviews and status reports:

> [Contractor] will develop, verify and submit for review and approval each item listed in Attachment A for the [Alpha]. [Contractor] will provide [Alpha] with weekly status reports outlining accomplishments, problems/issues, upcoming tasks and project resource requirements. (Document A.2, p. 1)

4) Opal Contract: Summary of the Evidence: Overall, the Opal contract exhibits the use of hierarchical elements, despite low uncertainty, occasioned by the threat (and possibly the reality) of opportunistic behavior. A combination of elements—nonmarket pricing, rule-based incentives, and behavior standards—is used to foster a stable, contractual business relationship.

5.4 Beta Contracts

As noted earlier, the Beta contracts were initially drafted by Beta, and then negotiated with the contractor. Beta also had a very powerful centralized IS department that was actively involved in the negotiation of external software development contracts for the firm. Four contracts were written between the

contractor and Beta. The first contract, Silver, was for a sales forecasting application, very similar to the Gold application at Alpha, with a size of about 105 units. The second contract, Ruby, was for a small, specialized module to be added to the Silver application. It was about 25 units in size. The third and fourth contracts were for two financial reporting applications, the Maple project with size of about 41 units, and the Oak project, the largest project, about 375 units in size. The Oak and Maple projects replaced existing applications.

1) The Silver Contract: From Beta's perspective, specification uncertainties on the Silver project were high as they had never installed a forecasting decision support system. Conversely, by the time these contracts were signed, the contractor had gained some experience from developing similar systems for other clients with similar needs. Thus, the process of discovery was less uncertain for the contractor than for Beta.

Beta invested considerable effort (during the four months prior to the start of the contract) interacting with the contractor, imparting forecasting requirements to the contractor. This nontransferable investment in time spent with the contractor continued intermittently throughout the system development process, which lasted about one year. Similar to Alpha, Beta was also unfamiliar with object-oriented systems analysis and design methodology. Considerable time was devoted, therefore, to learning the methodology to understand the development process as well as to be able to monitor the contractor's behavior. Conversely, on the contractor's side, knowledge gleaned from developing the system was expected to be partially transferable to forecasting system projects required of other clients, and, therefore, specific asset investments were more modest.

The Silver contract included a generalized, open-ended, "agree to amend" clause, indicating that the parties had the intention to create a relational contract:

> It is anticipated that the Proposal, from time to time, will be amended by mutual written agreement of the parties. (Document B.1, p. 11)

An example of a specific anticipated amendment to the contract was the decision on whether or not the contractor would be involved in training or what the compensation for that service would be:

> Should [Beta] choose to have [Contractor] be responsible for System training, compensation for said service shall be agreed on at that time. (Document B.1, p. 22)

We classify this clause as a nonmarket-based pricing element, as it contains not the slightest indication that market prices for training will be consulted in the setting of future compensation.

Certain personnel ("Key Persons") at the contractor were assigned to and held responsible for the project. Authority structures were established giving Beta the right to change or approve changes in such key persons.

> [Beta] shall have the right to interview and reject any person assigned by [Contractor] to replace a Key Person and shall have the right at any time to remove any person furnished by [Contractor] if in [Beta]'s sole judgment such person does not satisfy [Beta]'s requirements. (Document B.1, p. 10)

The Silver contract also delineated behavior standards with respect to design, development, and installation routines (Document B.1, pp. 7–8). Unique among these was a performance study, a series of test demonstrations conducted jointly by the contractor and Beta to evaluate the progress of the project. The performance study was required in lieu of written progress reports as Beta preferred a hands-on prototyping evaluation of the evolving system on-site to documented progress reports.

> A performance study will be conducted in accordance with the Proposal by [Contractor] and [Beta], at no additional cost to [Beta], in order to evaluate [the System]'s response time and impact of processing on [Beta]'s resources. (Document B.1, pp. 8–9)

> The performance study will also evaluate the operation of [the System] in accordance with the criteria set forth in the Proposal and the Performance Study will be conducted at [Beta]'s headquarters. (Document B.1, pp. 21–22)

Like the Gold contract for Alpha, the Silver contract adopted a combination pricing system with a fixed fee for the products and services outlined in the Silver contract, and a more open-ended, cost-recovery system for out-of-pocket travel and lodging expenses:

> In consideration for the design, development, and documentation referred to above, [Beta] agrees to pay [Contractor] a fee of [$K] . . .
>
> [Beta] will also reimburse [Contractor] for reasonable and necessary out-of-pocket travel and lodging expenses incurred by [Contractor] employees in performance of [Contractor]'s obligations under this Agreement, subject to [Beta]'s prior written approval. (Document B.1, p. 12)

Because the contractor was a relatively new firm at the time of the contract, Beta was particularly concerned with the viability of the contractor as a going concern. Their concern was in part related to their desire to protect specific investments consequent to system development. As a result, elaborate clauses were spelled out for a smooth termination of the contract without formal litigation should financial complications arise. The clauses for termination of agreement ran for five paragraphs, consisting of well over 350 words. The clauses detailed specific conditions for which the agreement would be terminated. Among others, the conditions included circumstances where the contractor was unable to deliver the system within a certain period of time, or when the contractor became encumbered with financial difficulties or legal disputes with third parties (Document B.1, pp. 13–14).

To further protect investments that were not transferable to another contractor should this one become uncooperative, Beta inserted authority elements in the contract, in the form of an explicit escrow assignment of source code in the contract:

> [Contractor] shall deposit a copy of the source code for [System] with [ZZZ Trust] (the Escrow Agent) upon Acceptance of the [System]. The source code will be updated with each Release which pertains to or would affect in any manner [Contractor]'s software and shall be deposited with the Escrow Agent. Such copies of the source code will be held in escrow in the event of a filing by [Contractor], or on [Contractor]'s behalf, of a bankruptcy petition. [Beta] will, upon payment of duplication costs and other handling charges of the escrow agent, be entitled to obtain a copy of the source code from the escrow agent. (Document B.1, pp. 15–16)

Informal dispute resolution mechanisms included in the Silver contract included the explicit delineation of procedures to terminate the contract, without legal recourse, if the application system was not performing to the satisfaction of Beta. This ensured that neither Beta nor the contractor could be held hostage for a software system with little or no salvage value.

> [Beta] and [Contractor] will review the Performance Study's results and shall have thirty days from the completion of the Performance Study to report to each other their findings. [Contractor] shall have 90 days to resolve any failure raised by either party if System does not meet the criteria set forth in the Proposal. If the failure is not resolved within 60 days, [Contractor] shall report to [Beta], in writing, what remains to be performed in order to correct the failure. After review of the Performance Study's results, and [Contractor]'s correction of any failure, client shall inform [Contractor] of its desire to continue work on the System or give written notice of its desire to terminate this Agreement.
>
> In the event [Beta] terminates this Agreement after its review of the Performance Study, [Contractor] shall be paid for the time it expended developing System to the date of the termination. Said amount shall not exceed [$M] plus any reasonable, documented travel, lodging and meal expenses incurred to date of termination. (Document B.1, p. 9)

2) Silver Contract: Summary of Evidence: Overall the Silver contract, similar to the Gold contract, makes liberal use of hierarchical elements to provide flexibility within a context reasonably safe from threats of opportunism, beginning with a general agreement to amend the contract. To cope with high levels of uncertainty, key personnel and detailed behavior standards are embedded in the contract. To protect specific assets accumulated prior to and during system development, the contract includes orderly termination and escrow clauses.

3) The Ruby Contract: The Ruby contract was for a specialized module developed to be used as part of the Silver application; it adds simulation and future event planning routines. The contract was drawn up about 21 months after the Silver project concluded and was about one-fourth the size of the Silver contract.

The Ruby contract was less detailed than the Silver contract, with more than half the contractual clauses pertaining to functional specifications of the module.

As with the Gold project, uncertainty about the market acquisition of a vital component of the system, as well as uncertainty of the availability of the data in certain format, triggered the embedding of authority structures in the contract giving the contractor the right to make decisions on issues concerning the nature of data to be processed by the system, and to require changes in applications to be connected to the Ruby application. These authorizing elements were written into a lengthy clause detailing several project assumptions. There were altogether 14 assumptions running roughly two pages (Document B.2, pp. 7–9).[2] The contract also entitled the contractor to re-evaluate and adjust both the fixed fees and project timing in cases where there were significant changes or modifications of these assumptions (Document B.2, p. 7). This entitlement ensured that the contractor would not be held hostage to the promises laid out at the outset of the agreement:

> The . . . assumptions have been used to derive application fees and project timing that follows. Significant changes to or modification of these assumptions will require re-evaluation and potential adjustment to both the application fees and project timing as set forth herein. (Document B.2, p. 7)

To cope with performance unobservability, a two-page addendum (Document B.2, Addendum) was included in the Ruby contract, listing in great detail the behavior standards or routines to be followed by the contractor in the course of developing the system. The procedures included on-site visits of the contractor for meetings and interviews with primary users of the system, discussions with Beta's data management personnel, presentation of progress reports, and demonstration of a prototype before final delivery of the system.

Similar to the Silver contract, the Ruby contract adopted a combination pricing system with a fixed fee for developing the application and an open-ended, cost-recovery system for out-of-pocket expenses (which did not have to be approved in advance by Beta) and other additional enhancements:

> The following table depicts the fees associated with implementing [Ruby].
> . . .
> Software Customizing fee [$K].
> . . .
>
> [Beta] agrees to pay all of [Contractor's] reasonable travel and out-of-pocket expenses associated with execution of this project.
> . . .
>
> [Mr. A] is the author and designer for [Ruby System]. He is available for consulting on system related issues at the rate of [$J] per day. (Document B.2, p. 10)

[2]Specific details of the project assumptions contain confidential details about the Silver application. For this reason, they are not included in this article.

4) Ruby Contract: Summary of Evidence: Overall, the Ruby contract, similar to the Opal contract with Alpha, seeks to manage the threat of opportunism. Some authority relations are established, but they involve both the contractor and Beta. Nonmarket pricing prevents the contractor from being held hostage to the failure of assumptions on which the project is based. In addition, extensive behavior standards are included to assure Beta that all is well.

5) The Maple and Oak Contracts: Following the completion of the Silver and Ruby projects, contracts were negotiated for two financial reporting applications, Maple and Oak, both replacements for existing automated systems. The Maple contract was for a managerial cost analysis application, and the Oak contract was for a financial accounting application. The process of discovery for these two applications was thus less uncertain for both Beta and the contractor, since both applications had many standardized or well-understood functions, and because both Beta and the contractor had had some experience with existing systems. The contractor, however, saw the systems as highly specific to Beta's needs and business operations, and therefore less likely to generate knowledge or code that could be leveraged with other clients. The salvage value of the systems, therefore, if the contract was not successful, would be very low.

The Maple and Oak contracts were similar in form and content to the Silver contract. Both adopted a combination pricing policy with a fixed fee for the system and incidental expenses reimbursed as they were incurred. Informal resolution mechanisms including termination clauses and escrow assignments of the source codes were retained as Beta was still concerned about the viability of the contractor as a going concern.

Conspicuously missing from the Maple and Oak contracts were authority structures for rights to approve changes in contractor personnel and behavior standards detailing progress reports, performance studies or a detailed design, development or installation requirements or procedures. Two explanations for this development are offered. First, with regard to the authority structures, it seems likely that because the systems were much more readily specified, less flexibility was needed in the contract. However, since some uncertainty around performance observability remains (the work was still performed at the contractor's office), and since some price exposure exists for Beta due to nonmarket pricing, we initially found it very curious that the project review clauses had been omitted. Based on our interviews with Beta and contractor informants, however, we suggest that a working relationship characterized by mutual learning, understanding, and trust had evolved between Beta and the contractor by the time these contracts were written. The evolution of informal, socialized governance seems to have substituted, in these instances, for formal hierarchical routines. Thus, the omission of authority structures and behavioral control elements in the contract may have been occasioned by the presence of governance mechanisms based on social agreement and a common world view. We will elaborate the impact of this overlay of social relations of interorganizational transactions on contract variety in the conclusion of this article.

6. CONCLUSION AND SUGGESTIONS FOR FUTURE RESEARCH

We have argued that software contracts should not always be assumed to be conditioned by market forces. Drawing on notions about contractual relations, we have identified and illustrated five types of hierarchical elements that could be included in contingent claims contracts to facilitate software development outsourcing by enhancing information flow, postponing decision making, and motivating and shaping contractor behavior. With illustrations of hierarchical clauses in Section 4 and case evidence presented in Section 5, the article highlights the important role hierarchical elements play in facilitating the governance of complex activities across organizational boundaries.

The article also identified two factors that affect varieties in software contracts: uncertainty and asset specificity. In addition to these factors, other factors may also affect contract variety. Future research should theorize more about some of these factors. For example, we have discussed how discrete contracting ignores past and future exchanges among contracting parties. In reality, contracts are often not discrete. As seen at Beta, contracts evolve as parties dynamically interact, commit to, and reinterpret their agreements with one another. Generally, mutual learning and adaptation occur through the social interaction that accompanies joint work activities. In fact, as social interaction intensifies, parties frequently take for granted the terms of the contract, and informally modify them as they learn to work together. This overlay of social relations on what may begin with a purely economic and instrumental transaction plays a crucial role in promoting informal adaptation in transaction structures and procedures. Thus, as contractual relationships evolve, one may expect corresponding changes in contract elements. Socially embedded business relationships may generate implicit standards of expected behavior, reliable information, and monitoring procedures that need not be incorporated explicitly into contracts. Ring and Van de Ven [48] argue that such norms of behavior are superior or at least equal to that of internal hierarchal relations in their ability to discourage malfeasance. Such socially embedded relationships, in fact, have the effect of producing stable relations of trust, obligation, and customs among independent firms [26,40,49]. And, in this sense, long-term relational transactions over time could combine the efficiency and flexibility of markets with the control and informational advantages of organization [50, p. 281]. Informal controls based on social agreements and a common world view [40] may, over time, replace formal hierarchical elements such as authority relations, standard operating procedures, and dispute resolution mechanisms, as the perceived interorganizational trust between the client and contractor develop.

Other environmental contexts in which contracting parties are embedded may have significant impacts on contract variety, particularly since contracts occur in a social context, not in a vacuum [51, p. 5]. For example, whether due to greater accountability or more bureaucracy, contracts with government agencies generally require more documentation than private contracts [52]. Moreover,

global software development outsourcing [53] may necessitate a much more complex set of hierarchical elements to facilitate communication among parties across different nation states, cultures, and languages.

By their very nature, tests of these propositions in future research will require examination of real software contractual documents. Content-analytic techniques and research designs [46,54] might be used to analyze these documents. Content-analytic research designs can be differentiated on the basis of data sources. We consider here single-source designs, in which contracts are derived from a single source; multiple-source designs, in which contracts are obtained from multiple clients or contractors; and standard or normative source designs.

A single-source design was used in this study. Several single- or common-source designs are possible. The investigator may compare contracts obtained either from a single client, from a single contractor, or from a single client–contractor pair. Single-source designs are conducive for making comparison of contracts over time. For instance, using trend analysis [55], the investigator may compare changes in hierarchical elements in contracts over a period of time. Patterns of governance sequences over time may also be traced using Abbott's [56] sequence methods of analysis. In addition to comparing contracts over time, the investigator may also compare contracts from a single source across differing situations. In cases where the client has both internal software development and external software development contracts, a comparison of the elements embedded in internal and external software contractual documents could yield insights into the differences between unilateral governance (internal hierarchies) and bilateral governance (external hierarchies).

Multiple-source designs would be useful for comparing contracts derived from clients or contractors who differ along a series of theoretically significant attributes. In standard or normative source designs real contracts could be compared with prescriptive software contracts to assess the extent to which real contracts deviate from prescription. Books and articles prescribing elements in software development contracts serve as prescriptive contracts (e.g., [38], [57], [58]). Since software contracting is multifaceted, with legal, managerial, and technological implications, whether the prescribed contracts are written by a lawyer, a manager, or a technician may also affect contractual variety. For example, contracts drawn by lawyers may focus more on elements protecting intellectual property; contracts drawn by managers may stress on authority structures, incentive systems, and procedures to align the contractor's goals with the client's; and contracts drawn by technicians may focus on product definition.

Based on the theories of transaction cost economics and relational contracting, we have created a conceptual framework of relational contracts for software development and provided preliminary empirical support for our expected determinants of contract variety. Clearly, there is more theoretical and empirical work to be done to understand better the process of systems development outsourcing. Further examination of software contracts can be expected to shed

more light on the intricacies of external software development relationships and how these relationships evolve over time.

REFERENCES

[1] O.E. Williamson, *Markets and Hierarchies: Analysis and Antitrust Implications*. New York: Free Press, 1975.

[2] W.S. Hesterly, J. Liebeskind, and T.R. Zenger, "Organizational Economics: An Impending Revolution in Organizational Theory?" *Academy of Management Review*, Vol. 15, No. 3, pp. 402–420, July 1990.

[3] A. Alchian and H. Demsetz, "Production, Information Costs, and Economic Organization," *American Economic Review*, Vol. 62, pp. 777–795, December 1972.

[4] A.L. Stinchcombe, "Contracts as Hierarchical Documents," in *Organizational Theory and Project Management*, A.L. Stinchcombe and C.A. Heimer, eds. Oslo, Norway: Norwegian University Press, 1985. (Also reprinted in A.L. Stinchcombe, *Information and Organizations*. Berkeley, CA: University of California Press, 1990, Chapter 6.)

[5] J.D. Thompson, *Organizations in Action*. New York: McGraw-Hill, 1967.

[6] G. Morgan, *Riding the Waves of Change: Developing Managerial Competences for a Turbulent World*. San Francisco: Jossey-Bass, 1988.

[7] Price Waterhouse, "Contracting out the Problem," *Price Waterhouse IT Review 1990–91*, United Kingdom: Price Waterhouse, 1990/91, pp. 8–11.

[8] J.R. Oltman, "21st Century Outsourcing," *Computerworld*, Vol. 24, No. 16, pp. 77–90, April 16, 1990.

[9] S.J. Whang, "Contracting for Software Development," *Management Science*, Vol. 38, No. 3, pp. 307–324, March 1992.

[10] W.B. Richmond, A. Seidmann, and A.B. Whinston, "Contract Theory and Information Technology Outsourcing," *Decision Support Systems*, Vol. 8, No. 5, pp. 459–477, September 1992.

[11] I.R. Macneil, "The Many Futures of Contracts," *Southern California Law Review*, Vol. 47, pp. 691–816, May 1974.

[12] I.R. Macneil, "Contracts: Adjustment of Long-Term Economic Relations under Classical, Neoclassical, and Relational Contract Law," *Northwestern University Law Review*, Vol. 72, pp. 854–905, 1978.

[13] V.P. Goldberg, "Toward an Expanded Economic Theory of Contracts," *Journal of Economic Issues*, Vol. 10, No. 1, pp. 45–61, March 1976.

[14] K.J. Crocker and S.E. Masten, "Pretia Ex Machina? Prices and Process in Long-Term Contracts," *Journal of Law and Economics*, Vol. 34, pp. 69–99, April 1991.

[15] O.E. Williamson, *The Economic Institutions of Capitalism*. New York: Free Press, 1985.

[16] J. Dearden, "The Withering Away of the IS Organization," *Sloan Management Review*, Vol. 28, pp. 87–91, Summer 1987.

[17] J. Elam, P. Lash, and D. Robey, "Whither the IS Organization? Part 2: Cooperative Strategies for Delivering IT Products and Services," Working Paper, Department of Decision Sciences and Information Systems, Florida International University, Miami, FL, 1991.

[18] D.J. Teece, "Economic Analysis and Strategic Management," *California Management Review*, Vol. 26, No. 3, pp. 87–110, Spring 1984.

[19] C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Communications of the ACM*, Vol. 30, pp. 416–429, 1987.

[20] S.S. Vicinanza, T. Mukhopadhyay, and M.J. Prietula, "Software-Effort Estimation: An Exploratory Study of Expert Performance," *Information Systems Research*, Vol. 2, No. 4, pp. 287–314, December 1991.

[21] L.J. Kirsch, "Formal, Informal, and Self Controls: An Analysis of Working Relationships During Systems Development Projects," unpublished doctoral dissertation, University of Minnesota, Minneapolis, MN, 1992.

[22] P. Milgrom and J. Roberts, *Economics, Organization and Management*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[23] *Restatement (Second) of Contracts*, Section 1. New York: American Law Institute, 1979.
[24] S. Macaulay, "Non-Contractual Relations in Business: A Preliminary Study," *American Sociological Review*, Vol. 28, pp. 55–67, February 1963.
[25] S. Macaulay, "An Empirical View of Contract," *Wisconsin Law Review*, pp. 465–482, 1985.
[26] M. Granovetter, "Economic Action and Social Structure: The Problem of Embeddedness," *American Journal of Sociology*, Vol. 78, pp. 481–510, 1985.
[27] C.J. Goetz and R.E. Scott, "Principles of Relational Contracts," *Virginia Law Review* Vol. 67, No. 6, pp. 1089–1150, September 1981.
[28] I.R. Macneil, "Relational Contracts: What We Do and Do Not Know," *Wisconsin Law Review*, pp. 483–525, 1985.
[29] American Bar Association, *Software Contract Forms*. Chicago: American Bar Association, Section of Science and Technology, 1984.
[30] American Bar Association, *Software Contract Forms*. Chicago: American Bar Association, Section of Science and Technology, 1987.
[31] American Bar Association, *Software Contract Forms*. Chicago: American Bar Association, Section of Science and Technology, 1992.
[32] J. Auer and C.E. Harris, *Computer Contract Negotiations*. New York: Van Nostrand Reinhold, 1981.
[33] D.H. Brandon and S. Segelstein, *Data Processing Contracts: Structure, Contents, and Negotiation*. New York: Van Nostrand Reinhold, 1984.
[34] L.J. Davis, D.A. Allen, T. Bowman, and J. Armstrong, *A User's Guide to Computer Contracting: Forms, Techniques and Strategies*. New York: Law & Business Inc. (Harcourt Brace Jovanovich, Publishers), 1984.
[35] M.L. Gordon and S.B. Starr, "Software Development Contracts and Consulting Arrangements: A Structure for Enforceability and Practicality," in *Negotiating Computer Contracts*, L.J. Davis and C.B. Ortner, eds. New York: Law & Business Inc. (Harcourt Brace Jovanovich, Publishers), 1983, pp. 142–192.
[36] W. Gray, "Drafting the Arbitration Clause—Appendix," in *Computer Contracts: An In-Depth Study*. Boston: Massachusetts Continuing Legal Education, Inc., 1984, pp. 707–717.
[37] P.S. Hoffman, *The Software Legal Book*. Croton-on-Hudson, NY: Shafer Books, 1992.
[38] D.F. Simon, *Computer Law Handbook: Software Protection, Contracts, Litigation, Forms*. Philadelphia: American Law Institute, 1990.
[39] C.I. Barnard, "Functions and Pathology of Status Systems in Formal Organizations," *Industry and Society*, W.F. Whyte, ed. New York: McGraw-Hill, 1946.
[40] W.G. Ouchi, "Markets, Bureaucracies, and Clans," *Administrative Science Quarterly*, Vol. 25, No. 1, pp.129–141, March 1980.
[41] K.M. Eisenhardt, "Control: Organizational and Economic Approaches," *Management Science*, Vol. 31, No. 2. pp. 134–149, February 1985.
[42] K.M. Eisenhardt, "Agency Theory: An Assessment and Review," *Academy of Management Review*, Vol. 14, No. 1, pp. 57–74, January 1989.
[43] V.P. Goldberg, "Regulation and Administered Contracts," *The Bell Journal of Economics*, Vol. 7, No. 2, pp. 426–448, Autumn 1976.
[44] E.F. Fama and M.C. Jensen, "Separation of Ownership and Control," *Journal of Law and Economics*, Vol. 26, pp. 301–325, Autumn 1976.
[45] R.K. Yin, *Case Study Research: Design and Methods* (revised edition). Newbury Park, CA: Sage Publications, 1989.
[46] O.R. Holsti, "Content Analysis Research Designs," *Content Analysis for the Social Sciences and Humanities*. Reading, MA: Addison-Wesley, 1969, Chap. 2.
[47] O.E. Williamson, "Credible Commitments: Using Hostages to Support Exchange," *American Economic Review*, Vol. 83, pp. 519–540, 1983.
[48] P.S. Ring and A.H. Van de Ven, "Cooperative Relationships between Organizations," Discussion Paper 125, Strategic Management Research Center, University of Minnesota, Minneapolis, MN, November 1989.

[49] P. Bromiley and L.L. Cummings, "Transaction Costs in Organizations with Trust," Department of Strategic Management and Organization Working Paper, University of Minnesota, Minneapolis, MN, 1992.

[50] J.R. Lincoln, "Japanese Organization and Organization Theory," *Research in Organizational Behavior*, Vol. 12, L.L. Cummings and B.M. Staw, eds. Greenwich, CT: JAI Press, 1990, pp. 255–294.

[51] D.M. Rousseau and J.M. Parks, "The Contracts of Individuals and Organizations," in *Research in Organizational Behavior*, Vol. 15, L.L. Cummings and B.M. Staw, eds. Greenwich, CT: JAI Press, 1993, pp. 1–44.

[52] R.L. Bernacchi and G.H. Larsen, *Data Processing Contracts and the Law*. Boston, MA: Little, Brown and Company, 1974.

[53] U.M. Apte, "Global Outsourcing of Information Systems and Processing Services," *The Information Society*, Vol. 7, pp. 287–303, 1991.

[54] K. Krippendorff, *Content Analysis: An Introduction to its Methodology*. Sage CommText Series. Newbury Park, CA: Sage Publications, 1980.

[55] M. Wood, "Alternatives and Options in Computer Content Analysis," *Social Science Research*, Vol. 9, pp. 273–286, 1980.

[56] A. Abbott, "A Primer on Sequence Methods," *Organization Science*, Vol. 1, No. 4, pp. 375–392, November 1990.

[57] G. Brandon and J.K. Halvey, *Data Processing Contracts: Structure, Contents, and Negotiation*, 3rd ed. New York: Van Nostrand Reinhold, 1990.

[58] L.R. Schwartz, *Computer Law Forms Handbook, A Legal Guide to Drafting and Negotiating*. New York: Clark Boardman Co., 1990.