Are Software Contracts Effective?

The Impacts of Contract Type and Repeated Contracting on Software Development Outsourcing Outcomes

Sandra A. Slaughter**
Scheller College of Business
Georgia Institute of Technology
Atlanta, GA 30308
ph: (404) 385-3115
fx: (404) 894-6030
sandra.slaughter@scheller.gatech.edu

Donald E. Harter
Syracuse University
721 University Avenue
Syracuse, NY 13244
dharter@syr.edu

Soon Ang
Nanyang Business School
Nanyang Technological University
Nanyang Avenue 639798
Singapore
asang@ntu.edu.sg

Jonathan Whitaker
Robins School of Business
28 Westhampton Way
University of Richmond
Richmond, Virginia 23173
jwhitaker@richmond.edu

October 2012

** Contact author. Do not cite, quote or distribute without permission of the authors.

The authors thank Allison Davis-Blake, Cynthia Beath, George Huber, and participants at seminars given at the University of Texas at Austin, the University of Southern California, the University of Calgary, and the University of California at Irvine for their valuable comments on drafts of this paper. Earlier versions of this paper were presented at meetings of the Academy of Management and INFORMS.
Are Software Contracts Effective?

The Impacts of Contract Type and Repeated Contracting on Software Development Outsourcing Outcomes

Abstract

Departing from prior research that focuses on the *ex ante* conditions predicting contract type, we investigate the *consequences* of contract type on a wide array of performance outcomes associated with software development outsourcing. Drawing on agency theory and institutional economics, we hypothesize that contract type (fixed price-FP, Hybrid, time and materials-T&M) with its varying inherent incentive properties will engender differential software development performance outcomes in terms of technical design verification quality, cost and schedule performance, and client validation quality. We further posit that repeated contracting should improve software development performance outcomes as repeated contractual experiences serve to mitigate issues of adverse selection and moral hazard.

We tested our hypotheses using feasible generalized least squares and spline regression to analyze longitudinal archival data on software development contracts and performance outcomes drawn from a large software vendor for a major client. Results show that, consistent with our contract type hypotheses, FP contracts produce the best - and Hybrid contracts produce the worst - technical outcomes of design verification quality, cost performance and schedule performance. Further both T&M and Hybrid contracts yield higher client validation quality than FP contracts. Contrary to our repeated contracting hypotheses, we found that repeated contracting does not always improve performance outcomes. Instead, repeated contracting serves more to ‘reinforce’ the incentives associated with a particular contract type. These results have important implications for research on software development contracts and for managerial practice.

**Key Words:** Outsourcing; Repeated Contracting; Contract Type; Software Development; Software Quality; Software Cost Performance; Software Schedule Performance.
Are Software Contracts Effective?

The Impacts of Contract Type and Repeated Contracting on Software Development Outsourcing Outcomes

INTRODUCTION

Computer software is critical to the functioning of almost every aspect of organizations. Managers and decision makers envision software developed with high quality, within budget, and without delays, but the reality can be quite different. In recent years, many organizations have turned to software development outsourcing in an effort to improve software performance outcomes. Some of the potential benefits of outsourcing include the client firm’s potential to obtain higher quality by tapping into the broader and deeper information technology (IT) skill base of a software vendor, to achieve greater cost efficiencies through the vendor’s economies of scale in serving multiple firms, and to leverage an increased ability to focus on core competencies (Levina and Ross 2003). However, given the complexities and uncertainties of software development, these performance benefits are not guaranteed.

Research on IT outsourcing suggests the importance of the relationships between clients and IT vendors (Gefen, Wyss and Lichtenstein, 2008; Poppo and Zenger 2002). Contracts form the nexus of mechanisms governing inter-firm relationships in outsourcing (Ang and Beath 1993; Banerjee and Duflo 2000; Crocker and Reynolds 1993; Jensen and Meckling 1976; Williamson 1979). To date, much research has focused on the determinants of contract type (Bajari and Tadelis 2001; Gopal and Sivaramakrishnan 2008; Kalnins and Mayer 2004; Whang 1992). Risk issues such as specification uncertainty, and agency issues such as the inability to ascertain vendor quality have been shown to influence contract type (Banerjee and Duflo 2000; Richmond, et al. 1992). An implicit assumption of this research is that when risk, agency and contractual relationship issues are considered in determining contract type, the chosen type of contract will
serve as an effective governance mechanism for engendering positive contractual outcomes (Grossman and Hart 1983; Holmstrom 1979; Milgrom and Roberts 1992).

In addition to contract type, Argyres, et al. (2007) have observed that firms engaging in outsourcing, particularly in IT, do not regard contracting as a ‘once-and-for-all activity” (p. 4). Instead, clients and vendors form long term contractual relationships and engage in multiple repeated contracts. As is well known, any form of contracting suffers from the challenges of incomplete contracting arising from asymmetric information (Rothschild and Stiglitz, 1976). Milgrom and Roberts (1992) and Williamson (1991) have identified repeated contracting as a means for both client and vendor to gain information advantages and reduce the asymmetry of information between parties through repeat interactions.

Generally, there is little empirical evidence on the performance consequences of contract type in software development outsourcing. One exception in the offshoring context is a study of software development offshoring by Gopal, et al. (2003; 2008) that examined the effects of contract type on vendor profits. However, Argyres et al. (2007) recently observed that there is no empirical evidence on the effectiveness of contract type in the context of repeated contracting in assuring the primary development outcomes of quality, cost and schedule. Given the increasing use of outsourcing in software development, and the use of repeated contracting in the IT context, it is vital to understand how such contracting decisions influence software development outcomes (Argyres et al. 2007; In’t Veld and Peeters 1989).

This paper extends prior research in tracing the effects of contract type and repeated contracting on quality, cost and schedule performance in software development outsourcing. The paper analyzes empirical evidence from a valuable longitudinal archival set of contracts and contract performance data drawn from a large software vendor for a major client. The next section describes the range of performance outcomes associated with software development. The
following section identifies contract types in software development outsourcing, and then presents our theory and hypotheses. The empirical methodology is then presented, followed by the results. Finally, the results are discussed and the paper concludes by identifying the strengths and limitations of the study.

**PERFORMANCE OUTCOMES IN SOFTWARE DEVELOPMENT OUTSOURCING**

Ensuring high levels of performance outcomes is particularly challenging in software development. Software systems are quite complex in that they are composed of a large number of parts with numerous interactions (Simon 1999). In contrast to many physical products, software is not easily visualized, does not conform to the laws governing physical products, and is almost infinitely adaptable (Brooks 1995). Software is an “experience” good, and its quality can often be ascertained only after it has been created. It is not surprising, therefore, that software development processes are exploratory and more akin to research and development, than to manufacturing (Slaughter et al. 2006). These inherent properties of software can create performance problems in outsourcing arrangements by making it difficult to specify and verify the desired range of behaviors of software, and to observe, control, and monitor the efficacy of development processes (Ang and Beath 1993; Kirsch 1996).

Three critical dimensions of software development projects include quality, project costs and schedule performance (Brooks 1995; Boehm 1981; Dey, Fan and Zhang, 2010). We consider each in turn. In the software engineering literature, two fundamental aspects of software quality have been identified: verification quality and validation quality (Fisher 2007; Katasonov and Sakkinen 2006; Boehm 1984). Figure 1 shows the differences between these dimensions of quality in the software development process. As depicted in Figure 1, the client has actual needs for the software. These actual needs are documented in the functional requirements. The vendor
then transforms the functional requirements into a formal design specification and builds the software to satisfy the design specification.

**Figure 1. Software Quality Dimensions**

![Software Quality Dimensions Diagram]

*Design verification quality* measures the extent to which the software has been developed correctly and performs correctly with respect to its design specification, *i.e.*, whether the developer has “built the software right”. In a typical software development outsourcing arrangement, the vendor conducts design verification tests before the software is put into production. Design verification defects capture deviations from the design specification discovered by the vendor in unit, module, integration or system testing of the software. An example of a design verification defect for an inventory system (such as in our study) is an incorrect computation of the economic order quantity (EOQ), with respect to the EOQ formula described in the design specification.

In contrast, *client validation quality* measures whether the correct software has been developed, *i.e.*, whether the developer has “built the right software”, and is assessed by the client who evaluates the implemented software relative to the client’s needs. Client validation quality is determined in acceptance and production tests. In outsourcing, acceptance tests are performed by the clients to ensure that the software meets their needs. There are two parts of acceptance
testing: functional requirements testing and stress testing. The functional requirements test compares the software to the requirements definition using a partial database. The stress test then uses a full production database with extremely high transaction levels to stress the system. Both acceptance tests are performed by the client or the client’s testing contractors and focus on performance with respect to the requirements definition. Acceptance defects capture deviations of the software from the requirements definition. An example of an acceptance defect is an excessive length of time needed to compute the EOQ for all inventory items in the database with respect to the time limits noted in the requirements definition.

After the software has been implemented and put into production, it is possible to observe the actual incidence of software defects during the operation of the software. Production defects, therefore, assess the extent to which the delivered software performs in a production environment according to actual client needs. Production tests are conducted by the client after the software is implemented and used in a production environment. Production defects are discovered by the client and capture discrepancies between actual needs and the delivered software. An example of a production defect is the realization by the client that the particular EOQ formula specified in the design specification and implemented in the software is not really the right formula that the client should be using.

Cost and schedule performance are the other two critical outcomes of software development projects (Fleming and Koppelman 2000; Deephouse, et al. 1995). A primary consideration for project cost performance is whether the development work performed costs what it was planned to cost. Cost performance is therefore a function of whether the actual project costs are over, equal to, or under the budgeted project costs, given the work performed on the project. Similarly, for project schedule, it is important to assess whether the development effort is on schedule and meets the estimated schedule dates. Schedule performance is determined by whether the actual
amount of work completed in a particular time period takes shorter or longer than planned or whether it is done according to the schedule.

**CONTRACT TYPE VARIATIONS IN SOFTWARE DEVELOPMENT OUTSOURCING**

The literature suggests the importance of contract types in development settings to provide incentives for performance (Goodhue, *et al.* 2003; Grossman and Hart 1983; Lewis and Sappington 1991). Reyniers and Tapiero (1995), for example, determine analytically that contract parameters can influence the propensity of vendors to develop high quality products and the willingness of clients to inspect the products. Contract theory and the agency literature assert that the type of a suitable contract is crucial because an appropriate type can minimize adverse selection and moral hazard, and can provide incentives to ensure that the agent’s goals are aligned with those of the principal (Crocker and Reynolds 1993; Grossman and Hart 1983; Milgrom and Roberts 1992).

Traditionally, time and materials (T&M) and fixed price (FP) contracts are the primary alternative contractual arrangements. These contracts reflect the tradeoffs between incentives and contracting costs (Baron and Besanko 1987). T&M contracts place technical and financial risks primarily on the client. This type of contract is less costly to negotiate but it may not provide sufficient financial incentives for the vendor to act in the client’s interests. On the other end of the spectrum, FP contracts place technical and financial risks primarily on the vendor and therefore provide stronger incentives for the vendor to reveal the vendor’s hidden information. This type of contract is costly to negotiate because it requires both parties to provide detailed and complete design specifications *ex ante*.

A hybrid or incentive-based contract offers a third type of contractual arrangement used in software development outsourcing (Kalnins and Mayer 2004). This is a risk sharing type of contract between the vendor and the client (Bajari and Tadelis 2001). Hybrid contracts are...
designed to balance incentives for efficiency and quality (Berends 2000) by avoiding or mitigating the tradeoffs between incentives and contracting costs. For example, a Hybrid contract can specify a negotiated cost, plus an incentive fee that is awarded to the vendor upon completion of the contract. The client bears the financial risk for costs (because the client pays all costs even if there is an overrun). However, the vendor’s profitability can vary dramatically based on the award portion of fee. In a Hybrid contract, the client can control the final profit using a fee mechanism that will provide a fair and reasonable incentive for the vendor to assume an appropriate share of the risk, to control costs and to deliver a high quality product.

**THEORY AND HYPOTHESES**

Given the complexities and uncertainties of software development processes, the implementation of the incentives embedded in different contract types may be difficult, thereby limiting their ability to appropriately influence post-contract behavior. Thus, it is important to not only consider whether an appropriate contract type is selected based on *ex ante* contractual conditions, but also how the incentives embedded in the contract will affect *ex post* software development outcomes.

After a contract has been negotiated, there is a possibility of *ex post* opportunism on the part of the contracting parties, *i.e.*, they can act opportunistically by seeking to exploit a situation to their own advantage (Williamson 1979). With respect to software development outcomes, *ex post* opportunism creates a traditional moral hazard problem where the vendor could seek to exploit information asymmetries and cut project costs and schedule time by delivering a lower level of quality than was contracted (Holstrom 1979).

**Contract Type and Software Quality Outcomes**

For design verification quality (which assesses the extent to which the software conforms to its design specifications), the vendor is the party with the most information about how to build
the software according to the design specifications since the vendor created the design
specifications and knows how to implement them. The vendor also knows whether the software
has been built correctly as the vendor is responsible for conducting design verification tests.
Given this information asymmetry, the vendor’s behavior will most influence design verification
quality.

Different contract types provide different financial incentives for rework cost avoidance that
can promote or discourage opportunistic behavior by the vendor in this situation (and thus, the
level of design verification quality delivered). Of the three contract types, we posit that FP
contracts would motivate the vendor to correctly produce the software. FP contracts are often
regarded as offering the highest powered incentive scheme (Prendergast 2002). FP contracts
offer cost certainty for the client. Under FP contracts, the vendor would have to pay for rework
and fix any design verification defects. An FP contract specifies the design requirements for the
software, and the vendor bears the costs of rework if the software developed does not match its
design specifications. This provides a financial motivation for the vendor to act strategically and
avoid the costs of rework by developing software that matches the design specifications,
ensuring higher design verification quality. Thus, the vendor’s financial incentives are to develop
the software “right” the first time according to the requirements specified in the contract so that
the contract is complete, and the vendor is paid the consideration as stipulated in the contract.

By contrast, Hybrid or T&M contracts are regarded as low powered incentive schemes. The
client pays for rework and the vendor has less incentive to reduce costs incurred by the client.
Therefore, the financial incentives for rework cost avoidance associated with these contracts are
not as stringent on the vendor as with FP contracts. In fact, design verification quality may be
lower in T&M and Hybrid contracts than in FP contracts. The lower level of design verification
quality could result from the vendor’s incentive to “create opportunities” for rework because the
vendor can pass along the rework costs to the client under these contract types. Another reason for lower design verification quality may be the vendor’s incentive to accommodate design changes; the literature on software engineering has documented challenges with achieving high levels of technical design quality when requirements are frequently changing (Katasonov and Sakkinen, 2006). Accordingly, we hypothesize that:

**Hypothesis 1a:** *FP contracts have higher design verification quality than T&M or Hybrid contracts.*

While FP contracts with their inherent financial incentives favor higher design verification quality, client validation quality may be lower. Client validation quality assesses the extent to which the software meets the actual needs of the client. The client is the party that possesses information about what the client needs. Given the ‘experience’ nature of software, clients may change their requirements or articulate new requirements as they discover what they need during the development process. However, in an FP contract, the vendor has a financial incentive to “freeze” the design specifications since the costs of requirements changes are borne by the vendor. Thus, the software developed, while adhering to the initial design specifications, may not satisfy the client’s real needs, leading to lower client validation quality.

Of the three contract types, T&M contracts should induce the greatest financial discipline on the client to participate actively in the development process to discover and articulate evolving needs. Under FP and Hybrid contracts, the client does not have as strong an incentive to participate as intensely because under FP contracts, the vendor bears the technical risks of defects in development, while under Hybrid contracts, the technical risks of development defects are shared by the vendor and client. By contrast, in T&M contracting, the vendor profits from the effort expended to develop, enhance or fix each software feature. The vendor is not under as much time pressure or financial pressure as in an FP contract to complete the software on time.
and under budget. Thus, the vendor can take care in designing, programming and testing the software and is likely to ultimately deliver higher quality software. Since the client bears the costs of requirements changes in a T&M contract, the vendor will be willing to accommodate design changes, and the resultant software is more likely to meet the client’s needs, increasing client validation quality.

Finally, in Hybrid contracts, financial risks are shared by the client and vendor. Most financial risk is on the client for costs, but the vendor’s profit margin depends on the award portion of the fee (which is determined by the client). Thus, there is less incentive for the vendor to undersupply or oversupply software quality, as the client can penalize the vendor for opportunistic behavior. This suggests that the level of client validation quality will be intermediate for Hybrid contracts, relative to the levels achieved under the more extreme contract structures of FP and T&M. Therefore, given the differential financial incentives embodied in the alternative contracting arrangements, and controlling for the \textit{ex ante} uncertainties of software design specifications and vendor quality, we would expect that:

\textbf{Hypothesis 1b:

T&M contracts have higher client validation quality than FP or Hybrid contracts.}

\textbf{Contract Type and Cost and Schedule Performance Outcomes}

In software development projects, cost and schedule performance are important performance outcomes (Deephouse, \textit{et al.} 1995). In contrast to software quality which is determined \textit{ex post}, it is possible for both the client and vendor to obtain intermediate information about cost and schedule performance \textit{during the conduct of the project}. Project management systems can provide periodic information to both client and vendor about actual cost and schedule progress versus targeted performance (Fleming and Koppelman 2000). Thus, there is no information asymmetry between the client and vendor with respect to project cost and schedule performance.
Since cost and schedule information is available to both parties, performance will be driven by the incentives for monitoring this information and taking action based upon it.

As we had described earlier, different contract types provide different financial incentives that have a direct impact on project cost and schedule performance. Of the three contract types, we hypothesize that FP contracts would motivate the vendor to produce the software according to its cost and schedule targets. Under FP contracts, the vendor bears the primary financial consequences for poor project cost and schedule performance – if there are schedule or cost overruns, the vendor will bear the responsibility of the overages. Thus, the vendor has a particularly strong financial incentive in FP contracts to monitor and react to project cost and schedule performance. Similarly, under T&M the client bears the primary financial consequences for poor project performance – if there are schedule overruns or cost overruns, the client bears the consequences of the overages. Thus, the client has a strong financial incentive in T&M contracts to monitor project cost and schedule performance and to ensure that the project stays on track.

In contrast, in Hybrid contracts, the risks and financial consequences of poor performance are shared. Based on the theory of diffused responsibilities and social loafing (Darley and Latane, 1968; Karau and Williams, 1993), we would expect that once responsibilities are shared between the client and the vendor, there is a common assumption that the other party is going to intervene when a problem arises. Thus, each party refrains from intervening, leading to lowered vigilance in monitoring project performance. This suggests that, under Hybrid contracts, periodic information about project cost and schedule performance is not as closely monitored as under the other contract types and that neither party has a strong enough incentive to take action sooner rather than later if problems occur. Thus, the projects completed under Hybrid contracts are more likely to get out of control and go over budget and schedule. Accordingly:
Hypothesis 2: FP and T&M contracts have better cost performance (H2a) and better schedule performance (H2b) than Hybrid contracts.

Repeated Contracting and Software Development Outcomes

An inability of the client to ascertain the quality of the vendor can create problems of adverse selection and moral hazard (Artz and Norman 2002; Stump and Heide 1996). For example, a vendor who does not possess the capability to provide certain levels of software quality might misrepresent its abilities by making false quality claims, and the client may be deceived by these claims if it is difficult to determine the vendor’s true capabilities.

Theory and empirical research in institutional economics has found that repeated interaction and establishing long-term relationships are critical for mitigating adverse selection and moral hazard (Corts and Singh 2004; Williamson 1979). Specifically, repeated contractual experiences between the vendor and client improve the client’s ability to determine the vendor’s true capabilities. With repeated interaction, the costs of contracting should be lower as the vendor and client learn how to negotiate and work together effectively.

In addition, repeated transactions are likely to provide incentives that decrease the likelihood of opportunism. When transactions are carried out frequently, the vendor and client will value repeat business, and neither will want to tarnish their respective reputations by acting opportunistically (Gefen, et al. 2008; Mayer 2006; Poppo and Zenger 2002). In contrast, with once-off or infrequent transactions, the incentive to act opportunistically and exploit information asymmetry may be higher. Thus, one could argue that increased information, learning, and possibly trust through repeated interaction could motivate both parties to optimize performance, and we would therefore expect software development performance outcomes to improve with repeated contracting. Accordingly, this suggests that:
Hypothesis 3: Repeated contracting is positively associated with software development performance outcomes such that repeated contracting will lead to higher design verification quality (H3a); higher client validation quality (H3b); better cost performance (H3c); and better schedule performance (H3d).

METHOD

Research Setting

We evaluated our hypotheses by analyzing detailed longitudinal, archival data collected in a field study of contracts completed over twenty years during a major software development outsourcing engagement between a large software vendor and client. To our knowledge no other study has examined contracting choices and outcomes over a time period of this length. Contracts were negotiated for individual software modules and components\textsuperscript{1} of a procurement and inventory management system to help manage the client’s spare parts acquisition. The procurement and inventory management system was extremely large, and was divided into various sub-systems such as inventory control, purchasing and spare parts maintenance. Different vendors were competitively selected to work on the different sub-systems. In addition, the bidding process for each contract within a sub-system was competitive, and a new bidding and contracting process was initiated for each module or component in a sub-system. The software vendor in this study was competitively selected to work on the spare parts acquisition piece of the inventory management system and was awarded the majority of the contracts in this sub-system.\textsuperscript{2} The modules completed for this sub-system by the vendor contained in total more than 2,000 software components and four million lines of code.

\textsuperscript{1} Sometimes modules were split into “components” or versions in which each component represented different subsets of the functionality (e.g., different types of economic order quantity calculations) or a lifecycle phase of the development process (e.g., design vs. coding). A separate contract was negotiated for each component.

\textsuperscript{2} Although different vendors were awarded work on different sub-systems, contracts within a sub-system were also competitively bid, and sometimes a vendor would not get all of the work in the sub-system. The software vendor in this study completed most
One hundred and fourteen contracts were negotiated for the software modules. Of these contracts, eleven were for feasibility studies rather than for software development, and were not included in our analysis, leaving a total of 103 contracts, including T&M contracts (n=33), FP contracts (n=45) and Hybrid contracts (n=25). The T&M contracts specified negotiated hourly rates for vendor employee time and prices for materials. The client paid based on the time charged by the vendor at these rates. The total price in these contracts was not predetermined so the vendor could pass on costs resulting from client-requested changes. As the client bore the costs in these contracts, the risks and responsibility for control in these contracts were primarily placed on the client. In contrast, the FP contracts specified a set price for the contract in exchange for completion of software development services as specified in the contract. All cost overruns were borne by the vendor, placing all risk and responsibility for control on the vendor. Hybrid contracts combine the features of T&M and FP contracts and are used to balance incentives for cost efficiency and quality (Kalnins and Mayer 2004; Berends 2000). The Hybrid contracts in this study included an agreed upon cost target with a fee awarded by the client upon completion of the contract. The client paid all costs, even if in excess of the cost target. Although much of the financial risk was still with the client, the award fee provided a powerful incentive mechanism for the vendor because the vendor’s profit margin was based on the award portion of the fee.

The contracts selected for the engagement were organized temporally into three separate contracting regimes: the Hybrid contracts were used in the early part of the engagement, followed by the use of FP contracts in the next regime, and T&M contracts were used in the third and final part of the engagement. At the beginning of the relationship between the client and

---

of the contracts on the spare parts acquisition sub-system, but occasionally lost a contract to a competitor. By bidding out contracts within sub-system, the client could maintain a competitive environment for contract work even within a sub-system.
vendor (Regime 1), the two parties had no experience in working together on software
development projects. With no prior software development experience with the client, the vendor
was not willing to assume all financial risk with a FP contract. With no prior experience with the
vendor, the client was not willing to assume all financial risk with a T&M contract. The
compromise (as suggested by Kalnins and Mayer 2004) was a Hybrid contract. After three years,
the client requested a shift from Hybrid to FP contracts with the intent of moving performance
risk to the vendor. The vendor had three years of working with the client and had a better
understanding of the client’s requirements and was therefore willing to shift to the use of FP
contracts in Regime 2. After twelve years of software development (3 years in Hybrid, 9 years in
FP), the client had gained confidence in the vendor’s ability to successfully develop systems, and
the vendor had evolved a more refined understanding of the client. Thus, the parties agreed to
perform all remaining work on the project as T&M. The choice of T&M contracts for the last
eight years of the engagement reduced the client’s costs of writing a very detailed contract such
as that required for a fixed price environment.

From the perspective of research design, this setting is ideal for a study of performance
outcomes in a repeated contracting setting. First, our longitudinal design allows us to relate the
performance effects (that emerge over time) to different contract types. Second, focusing on the
repeated contracting between one client-vendor dyad over a long period of time provides a
natural control for client and vendor characteristics that enables us to identify how the increasing
experience of the client and vendor in working together affects contract outcomes. Finally, the
three distinct contracting regimes in this vendor-client engagement offer a unique opportunity to
compare the effects on contract outcomes of the experience due to repeated contracting within a
contract type. We can, therefore, evaluate whether repeated contracting reinforces or mitigates
the incentives embedded in the different contract types.
Data and Measures

Data were collected on the 103 contracts completed during this engagement starting from 1987 (when the first contract on the project was negotiated) to 2006 (when the last contract on the project was completed). All data were obtained from archival sources at the software vendor including archived contract status reports and process maturity assessment reports, project management systems, software estimation tool archives, configuration management systems, and software defect databases. The data and measures are described in more detail in the following sections.

Software Development Outcomes: In software development, three fundamental outcomes are quality, cost and schedule (Boehm 1981), and we examine each of these outcomes. We further distinguish the two types of software quality identified earlier: design verification quality and client validation quality.

Design verification quality is measured as the number of verification defects recorded for the software divided by the size of the software in thousands of lines of code (KLOC). Design verification defects were identified during the vendor’s verification test for each software module. An independent test team within the vendor (not the development teams) tested the functionality of the software module based on its design specifications. A defect was identified when the software module did not function according to its defined specifications. The test team recorded all design verification defects on software problem reports. Both the vendor and the client reviewed these reports to ensure agreement in terms of whether the problem constituted a design verification defect in the software, and the vendor’s configuration management unit then recorded the characteristics of each defect in a database. We extracted data on defects from this database for our measure of design verification quality. To ease interpretation of this variable as a measure of “quality”, we multiplied it by -1.
Consistent with our earlier definition, client validation quality is measured based upon the extent to which the software delivers what the user actually needs as represented in the requirements definition (acceptance defects) and as discovered in the production environment (production defects). Specifically, client validation quality is the sum of acceptance defects and production defects recorded for the software module divided by the size of the software in thousands of lines of code (KLOC). Upon completion of development testing and resolution of all design verification defects found, the software module was tested by the client’s acceptance test team. This team included an independent validation and design verification contractor hired by the client as well as selected users from the client’s user community. The client’s test team assessed the functionality of the software in two stages. In the first stage, sample databases of about 10% the size of actual production databases were used; in the second stage, full production databases were used, and stress tests were performed to evaluate response times and the ability of the software to handle large numbers of users simultaneously. The goals of the client’s team test were to evaluate whether the functionality implemented in the software matched their requirements and whether the software performed accordingly in a production environment. Each defect recorded by the client was reviewed and audited by the vendor’s quality assurance unit and an independent client audit team to ensure accuracy, agreement and completeness and then was entered into the database from which we extracted acceptance defects. Production defects were found and recorded by the vendor’s data center operators or by the client’s users during the first twelve months of their actual day-to-day use of the software in a production environment. All defects were audited by the client and by the vendor’s configuration management and quality assurance units against the requirements definition to ensure that the production "bug" was really a production defect and not a request for a new feature or an enhancement. Production defects were entered into the database from which we extracted the
data to measure client validation quality. To ease interpretation of this variable as a measure of “quality”, we multiplied it by -1.

Cost Performance is evaluated using numbers from earned value analysis (Fleming and Koppelman 2000). Earned value analysis is a project management technique that incrementally measures progress on a project based on technical, schedule and cost performance. It ascribes a dollar value to planned versus actual project costs for work performed. Cost performance is measured by comparing the budgeted cost of work performed (earned value) to the actual cost of work performed (actual cost). A positive or favorable cost variance reflects better cost performance and means that the work performed on the project costs less than anticipated, i.e., the project has earned more value than planned. Conversely, worse cost performance is reflected in a negative or unfavorable cost variance which means that the work performed costs more than planned. In our study, the earned value analysis was performed monthly, and we obtained the cost variance amount (positive or negative and in thousands of dollars) for each month for each software module from the financial archives of the vendor’s project management office.

Schedule Performance is also evaluated using earned value analysis. Schedule performance is measured by comparing the budgeted cost of work performed (earned value) to the budgeted cost of work originally scheduled for the same timeframe (planned value). A favorable or positive schedule variance means that more work is being accomplished than originally planned in a given timeframe, suggesting that the project is ahead of schedule. An unfavorable or negative schedule variance means that less work is being accomplished than originally planned for a period of time, suggesting that the project is behind schedule. We obtained the schedule variance amount (positive or negative and in thousands of dollars) for each month for each software module from the scheduling archives of the vendor’s project management office.
**Contract Type:** We identified the three types of contracts and operationalized contract type as a series of binary variables where \( TM = (1 \text{ if T&M contract, 0 otherwise}) \), \( Hybrid = (1 \text{ if Hybrid contract, 0 otherwise}) \) and FP contracts are indicated when \( TM \) and \( Hybrid \) are both zero. Data on contract type as well as contract start date and completion date were extracted from archived Contract Status Reports generated by the vendor for the client.

**Repeated Contracting:** Repeated contracting between the client and vendor was assessed by counting the number of contracts completed on the project before the current contract started. We extracted the start and completion dates for each contract from the archived contract status reports. Using this information, we constructed a timeline so that we could count the contracts completed prior to each contract’s start date. With each contract completed, the client gains experience in contracting and in the software being developed; similarly, the vendor also gains experience in working with the client and in understanding the client’s requirements. Thus, the number of contracts completed before work starts on the current software module provides a measure of the parties’ repeated contracting together.

**Controls:** In our analysis of contract performance, we controlled for factors that could be correlated with outcomes. First we include *ex ante* uncertainty factors as possible correlates of outcomes. According to contract and agency theories, adverse selection problems can occur when information is hidden prior to the negotiation of an exchange transaction, and moral hazard problems can arise from the information asymmetry between the client and vendor. These problems could potentially influence software development performance in addition to the particular contract type (Milgrom and Roberts 1992). In the context of software development, the two primary sources of *ex ante* uncertainty are: uncertainty in design specifications and uncertainty about the vendor’s capabilities (Artz and Norman 2002; Stump and Heide 1996; Kalnins and Mayer 2004). When the design specifications and the vendor’s capabilities are not
known *ex ante*, the task of writing complete contracts is often not possible. Software development performance could suffer as a result.

Uncertainty of design specifications was measured using two dimensions: specification uncertainty and design complexity. Measures of *specification uncertainty* and *design complexity* were obtained from the archives of a software estimation tool used by the vendor’s engineering department to prepare development schedules and costs for the software. Specification uncertainty refers to the extent to which the specifications for the software are unclear or incomplete (Boehm 1981). The level of specification uncertainty for each software module was measured in the estimation tool archives using a scale of 1 (low) to 5 (high) that captured the extent of ambiguity in the client’s requirements. The design complexity of the software module refers to the level of consistency and coherence of the functionality to be implemented in the software (Banker and Slaughter 2000). Design complexity was measured in the estimation tool archives using a scale of 1 (low) to 5 (high) that evaluated the level of coherence in the module’s design. We used principal components analysis to reduce the archival measures to two independent factors, one for specification uncertainty and the other, for design complexity. We then operationalized specification uncertainty and design complexity for each software module by computing scores for each factor using the regression method.

The *vendor’s software process maturity* was measured in terms of vendor quality certification. Vendor quality certification levels were assigned to individual software modules and components based on the vendor’s level of software development process maturity (ranging

---

3 Two factors were extracted from principal components analysis of the archival measures using the eigenvalue > 1 criterion and a varimax rotation. The two factors extracted explained 79% of the variation in the data. The loadings of the archival measures on their respective factors range from 0.79 to 0.97, supporting convergent validity of the factors. Specifically, items and loadings for the specification uncertainty factor are: *ambiguous or incomplete user requirements* (0.84) and *many complex modules and paths* (0.79). For the design complexity factor, the loadings are: *informal or hasty design with no design automation* (0.97). Cross loadings of the measures are all below 0.19, supporting discriminant validity.
from level 1 – initial to level 5 - optimizing) as of the start date of each module’s contract.\textsuperscript{4} Auditors in external divisions of the vendor and governmental agencies provided independent assessments of the vendor’s software process maturity. The auditors used the Software Engineering Institute’s SW-CMM (Paulk, \textit{et al.} 1995) to evaluate the level of maturity of software development processes at the vendor at seven different times over the length of the engagement with the client. After each evaluation, the vendor received a report documenting the date of the assessment, the maturity rating, a list of strengths based on key process areas in the SW-CMM, and a list of areas for improvement to reach the next maturity level. These reports were archived, and provided the source of the process maturity ratings used in the study. To facilitate interpretation and comparison of the estimated coefficients we standardized this variable to its Z-score.

Finally, we also controlled for software size. The software engineering research suggests that software size is an important correlate of software defects, costs and schedules (Boehm 1981). \textit{Software size} is assessed as the thousands of lines of code (KLOC) in each module. Since all modules were coded using a single programming language, the use of lines of code to measure software size is appropriate. Size data were obtained from the vendor’s configuration management tools that tracked the lines of source code in each software module. We transformed size using the natural logarithm to reduce skewness.

\textbf{ANALYSIS AND RESULTS}

Descriptive statistics and correlation matrices for the variables in our analysis are reported in Tables A1 and A2 in the Appendix. The following section describes the specifications, estimations and results for our hypotheses on the performance outcomes associated with contract

\textsuperscript{4} We used the process maturity level as of the start of each contract because the vendor did not change the processes used for a particular contract after work started on it.
type (H1a-H2b). We then describe our spline regression analysis which models the different relationships between repeated contracting and performance outcomes in the different contract regimes (H3a-H3d).

**Estimation of Performance Outcomes by Contract Type (H1a-H2b)**

We relate contract type to the different performance outcomes (design verification quality, client validation quality, cost performance, and schedule performance). The performance outcome equations are listed below. In these equations, \( i \) refers to the software module, \( j \) refers to the contract segment for the module, and \( t \) refers to the month. Since the data are nested panels (contract segment within module for the quality equations, and time period within contract segment for the cost and schedule equations) we estimated the equations using feasible generalized least squares (GLS) to correct for the possible non-independence of disturbances across different contract segments for the same module or across different time periods for the same contract segment (Greene 2003). We separately estimated the equations for each performance outcome.

1. **Design Verification Quality**
   
   \[
   y_{i} = \beta_{01} + \beta_{11} X_{1i} + \beta_{21} X_{2i} + \beta_{31} X_{3i} + \beta_{41} X_{4i} + \beta_{51} TM_{i} + \beta_{61} Hybrid_{i} + \epsilon_{i1}
   \]

2. **Client Validation Quality**
   
   \[
   y_{i} = \beta_{02} + \beta_{12} X_{1i} + \beta_{22} X_{2i} + \beta_{32} X_{3i} + \beta_{42} X_{4i} + \beta_{52} TM_{i} + \beta_{62} Hybrid_{i} + \epsilon_{i2}
   \]

3. **Cost Performance**
   
   \[
   y_{i} = \beta_{03} + \beta_{13} X_{1i} + \beta_{23} X_{2i} + \beta_{33} X_{3i} + \beta_{43} X_{4i} + \beta_{53} TM_{i} + \beta_{63} Hybrid_{i} + \epsilon_{i3}
   \]

4. **Schedule Performance**
   
   \[
   y_{i} = \beta_{04} + \beta_{14} X_{1i} + \beta_{24} X_{2i} + \beta_{34} X_{3i} + \beta_{44} X_{4i} + \beta_{54} TM_{i} + \beta_{64} Hybrid_{i} + \epsilon_{i4}
   \]

We conducted a variety of diagnostics. Using the criteria established by Belsley, et al. (1980) and Kennedy (2003) we found no evidence of collinearity problems in any of the models. As
reflected in the pair-wise correlation matrices (see Appendix), the inter-correlations between variables are generally modest, with the highest correlation of 0.53. Variance inflation factors and condition indices are well below the cutoff levels for all four of the models (highest average VIF = 1.590, highest condition index = 8.199). No outliers were detected in the design verification quality model, three outliers in the client validation quality model, one outlier in the cost performance model and four outliers in the schedule performance model. In all cases, after removing the outliers and re-running the analysis, the results are similar to those when the outliers are included; thus, we report our results with all observations included. We also evaluated whether the disturbances are autocorrelated using the test of Wooldridge (2001) for autocorrelation in panel data models and found evidence of serial correlation in the design verification quality and client validation quality models. Thus, we corrected for AR1 serial correlation in our estimation of these equations. We also corrected for panel-specific heteroskedasticity in the client validation quality model.

Columns 3 and 4 of Table 1 show the detailed results from the GLS estimation of each performance model. Column 3 reports the results when only control variables are in the equations, and Column 4 reports the results when contract types are added. We use the results reported in Column 4 to compute the marginal means of the outcomes associated with each contract type for each performance outcome. The marginal means represent the estimated means for design verification quality and client validation quality and for cost or schedule performance by type of contract, controlling for other factors in the model.

In Hypothesis 1a, we predicted that FP contracts would have higher design verification quality than T&M or Hybrid contracts. Our results support this hypothesis. The results in Table 1 show that FP contracts have higher design verification quality (1.392 design verification defects per KLOC) than both T&M contracts (1.428 design verification defects per KLOC) and Hybrid
contracts (2.025 design verification defects per KLOC), although only the difference in means between FP and Hybrid contracts is statistically significant (FP vs. T&M: \( \chi^2 = 0.02, p > 0.10 \); FP vs. Hybrid: \( \chi^2 = 4.51, p < 0.05 \)).

| Table 1 |
| Results from GLS Estimation of Contract Type and Performance Outcomes |

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dependent Variable</strong></td>
<td><strong>Independent Variable</strong></td>
<td><strong>GLS Estimate (Std Error)</strong></td>
<td><strong>GLS Estimate (Std Error)</strong></td>
</tr>
<tr>
<td><strong>Design Verification Quality</strong></td>
<td>Specification Uncertainty</td>
<td>0.064 (0.106)</td>
<td>0.084 (0.104)</td>
</tr>
<tr>
<td></td>
<td>Design Complexity</td>
<td>0.125 (0.112)</td>
<td>0.190 † (0.107)</td>
</tr>
<tr>
<td></td>
<td>Software Process Maturity</td>
<td>-0.060 (0.118)</td>
<td>-0.211 † (0.123)</td>
</tr>
<tr>
<td></td>
<td>Software Size</td>
<td>0.136 * (0.056)</td>
<td>0.128 * (0.054)</td>
</tr>
<tr>
<td></td>
<td>Contract Type = FP {or Intercept}</td>
<td>-1.610 *** (0.230)</td>
<td>-1.392 *** (0.280)</td>
</tr>
<tr>
<td></td>
<td>Contract Type = HYBRID {relative to FP}</td>
<td></td>
<td>-0.633 * (0.298)</td>
</tr>
<tr>
<td></td>
<td>Contract Type = T&amp;M {relative to FP}</td>
<td></td>
<td>-0.036 (0.270)</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Wald ( \chi^2 = 8.06, p &lt; 0.10 )</td>
<td>Wald ( \chi^2 = 14.05, p &lt; 0.05 )</td>
<td></td>
</tr>
<tr>
<td><strong>Client Validation Quality</strong></td>
<td>Specification Uncertainty</td>
<td>0.224 (0.189)</td>
<td>-0.086 (0.169)</td>
</tr>
<tr>
<td></td>
<td>Design Complexity</td>
<td>0.450 (0.233)</td>
<td>0.231 (0.175)</td>
</tr>
<tr>
<td></td>
<td>Software Process Maturity</td>
<td>0.350 * (0.168)</td>
<td>0.211 (0.135)</td>
</tr>
<tr>
<td></td>
<td>Software Size</td>
<td>0.671 *** (0.097)</td>
<td>0.762 *** (0.078)</td>
</tr>
<tr>
<td></td>
<td>Contract Type = FP {or Intercept}</td>
<td>-4.147 *** (0.463)</td>
<td>-5.118 *** (0.372)</td>
</tr>
<tr>
<td></td>
<td>Contract Type = HYBRID {relative to FP}</td>
<td></td>
<td>0.613 * (0.273)</td>
</tr>
<tr>
<td></td>
<td>Contract Type = T&amp;M {relative to FP}</td>
<td></td>
<td>2.088 *** (0.484)</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Wald ( \chi^2 = 49.35, p &lt; 0.001 )</td>
<td>Wald ( \chi^2 = 116.43, p &lt; 0.001 )</td>
<td></td>
</tr>
<tr>
<td>Cost Performance</td>
<td>Specification Uncertainty</td>
<td>11.443 (8.431)</td>
<td>7.429 (9.852)</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Design Complexity</td>
<td>-42.908 *** (11.478)</td>
<td>-2.496 (12.132)</td>
<td></td>
</tr>
<tr>
<td>Software Process Maturity</td>
<td>10.393 (12.042)</td>
<td>-36.875 ** (12.294)</td>
<td></td>
</tr>
<tr>
<td>Software Size</td>
<td>-19.448 *** (4.253)</td>
<td>-9.252 (5.669)</td>
<td></td>
</tr>
<tr>
<td>Contract Type = FP {or Intercept}</td>
<td>-9.001 (16.953)</td>
<td>25.250 (33.240)</td>
<td></td>
</tr>
<tr>
<td>Contract Type = HYBRID {relative to FP}</td>
<td>-313.950 *** (30.035)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contract Type = T&amp;M {relative to FP}</td>
<td>-23.937 (28.662)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>Wald $\chi^2 = 38.18$, $p &lt; 0.001$</td>
<td>Wald $\chi^2 = 160.02$, $p &lt; 0.001$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schedule Performance</th>
<th>Specification Uncertainty</th>
<th>3.746 (2.414)</th>
<th>0.346 (2.844)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Complexity</td>
<td>-7.556 * (3.292)</td>
<td>1.470 (3.506)</td>
<td></td>
</tr>
<tr>
<td>Software Process Maturity</td>
<td>5.769 † (3.463)</td>
<td>-5.946 † (3.561)</td>
<td></td>
</tr>
<tr>
<td>Software Size</td>
<td>-1.791 (1.220)</td>
<td>2.469 (1.637)</td>
<td></td>
</tr>
<tr>
<td>Contract Type = FP {or Intercept}</td>
<td>-16.054 *** (4.855)</td>
<td>-18.684 * (9.592)</td>
<td></td>
</tr>
<tr>
<td>Contract Type = HYBRID {relative to FP}</td>
<td>-79.531 *** (8.668)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contract Type = T&amp;M {relative to FP}</td>
<td>5.182 (8.271)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>Wald $\chi^2 = 12.90$, $p &lt; 0.05$</td>
<td>Wald $\chi^2 = 114.76$, $p &lt; 0.001$</td>
<td></td>
</tr>
</tbody>
</table>

Notes: $n = 103$ in the Quality equations, $n = 939$ in the Cost and Schedule equations; † $p < 0.10$; * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. Estimates are reported after correcting for AR1 serial correlation in the design verification and client validation quality equations and panel-specific heteroskedasticity in the client validation equation.

In Hypothesis 1b, we posited that FP contracts would have lower client validation quality than T&M and Hybrid contracts. Our results support this hypothesis, as FP contracts have more client validation defects (5.118 client validation defects per KLOC) than both T&M (3.030 client validation defects per KLOC) and Hybrid contracts (4.505 client validation defects per KLOC),
and these differences are statistically significant (FP vs. T&M: $\chi^2 = 18.58$, $p < 0.001$; FP vs. Hybrid: $\chi^2 = 5.05$, $p < 0.05$).

In Hypothesis 2a, we predicted that T&M and FP contracts would have higher cost performance than Hybrid contracts. Our results support this hypothesis, as both T&M (1.313 favorable cost variance) and FP contracts (25.250 favorable cost variance), have significantly higher cost performance than Hybrid contracts (288.696 unfavorable cost variance) (T&M vs. Hybrid: $\chi^2 = 74.40$, $p < 0.001$; FP vs. Hybrid: $\chi^2 = 109.26$, $p < 0.001$).

Finally, in Hypothesis 2b, we predicted that T&M and FP contracts would have higher schedule performance than Hybrid contracts. Our results support this hypothesis, as both T&M (13.502 unfavorable schedule variance) and FP contracts (18.684 unfavorable schedule variance) have significantly higher schedule performance than Hybrid contracts (98.216 unfavorable schedule variance) (T&M vs. Hybrid: $\chi^2 = 76.24$, $p < 0.001$; FP vs. Hybrid: $\chi^2 = 84.20$, $p < 0.001$).

**Effects of Repeated Contracting on Performance Outcomes (H3a-H3d)**

We now turn our attention to testing hypotheses of repeated contracting on performance outcomes. Recall, we hypothesized in H3a-3d that repeated contracting is associated with higher design verification quality (H3a); higher client validation quality (H3b); better cost performance (H3c); and better schedule performance (H3d).

Given the temporal nature of the data such that contract type differs across the three temporally ordered contract regimes, we analyzed the effects of repeated contracting of each contract type in each regime using spline regression models (Marsh and Cormier 2001; Greene 2003; Pindyck and Rubinfeld 1998). Traditional regression analysis would estimate a coefficient for repeated contracting at the overall mean, but the overall mean of repeated contracting in our data is relevant for neither the Hybrid nor the TM contract regimes as the mean is outside the
range of repeated contracting in these regimes. In addition, regression analysis would assume that repeated contracting has exactly the same relationship to performance outcomes in each contract regime. It is possible that repeated contracting may have a different (e.g., stronger or weaker) relationship to performance in different regimes. Thus, we use spline regression models to allow the relationships between repeated contracting and performance outcomes to be calibrated to each regime and to change trajectory in response to a change in contracting regime.

Spline regression is a technique that fits and smooths the twists and turns of a time line or series of sequentially ordered events (Marsh and Cormier 2001). Spline regression models are operationalized using restricted dummy variables as with unrestricted dummy variables there is a different intercept and slope at each shift in the regression line. A spline regression model avoids this inappropriate “jump” or break in joining two different regression lines by using a spline knot to connect the lines together (Pindyck and Rubinfeld 1998). The regression line changes direction at these knots but does not “jump” at these points. The spline knots in our analysis occur at two known locations: 1) the shift from Hybrid contracts to FP contracts at the beginning of Regime 2; and 2) the shift from FP contracts to T&M contracts at the beginning of Regime 3.

To set up the spline model requires operationalizing spline adjustment variables. We used the procedure described in Marsh and Cormier (2001) to create spline adjustment variables for repeated contracting in the different regimes and added them to each performance outcome equation. The procedure required us to operationalize two additional spline adjustment variables for repeated contracting. We first created two dummy variables (D1 and D2). D1 takes on the value of 1 for the current contract if it is in the second regime (FP) or third regime (T&M) and 0 otherwise. D2 takes on the value of 1 for the current contract if it is in the third regime (T&M) and 0 otherwise. We then used these dummy variables to construct the two additional spline adjustment variables for repeated contracting: repeated contracting regime 2 = D1*(contracts
completed before the current contract starts – contracts completed in Regime 1 before the current contract starts), and repeated contracting regime 3 equals D2*(contracts completed before the current contract starts – contracts completed in Regimes 1 and 2 before the current contract starts). These spline variables will allow us to model different slopes for repeated contracting and performance outcomes in the different contract regimes. Thus, we added the following variables to each model: \( \beta_7 \times \text{Repeated-Contracting-Regime-1}_{ij} + \beta_8 \times \text{Repeated-Contracting-Regime-2}_{ij} + \beta_9 \times \text{Repeated-Contracting-Regime-3}_{ij} \) and used GLS to re-estimate each performance outcome equation. The results from this estimation are reported in column 3 of Table 2.

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>Independent Variable</th>
<th>GLS Estimate (Std Error)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Verification Quality</strong> (-1 * Defects/KLOC)</td>
<td>Specification Uncertainty</td>
<td>-0.082 (0.101)</td>
</tr>
<tr>
<td></td>
<td>Design Complexity</td>
<td>0.204 † (0.116)</td>
</tr>
<tr>
<td></td>
<td>Software Process Maturity</td>
<td>-0.167 (0.135)</td>
</tr>
<tr>
<td></td>
<td>Software Size</td>
<td>0.116 † (0.068)</td>
</tr>
<tr>
<td></td>
<td>Repeated contracting {Regime 1}</td>
<td>-0.044 † (0.029)</td>
</tr>
<tr>
<td></td>
<td>Repeated contracting {Regime 2 relative to Regime 1}</td>
<td>0.067 * (0.031)</td>
</tr>
<tr>
<td></td>
<td>Repeated contracting {Regime 3 relative to Regimes 1 &amp; 2}</td>
<td>-0.062 * (0.027)</td>
</tr>
<tr>
<td></td>
<td>Intercept</td>
<td>-1.484 ** (0.508)</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>Wald ( \chi^2 = 25.05, \ p &lt; 0.001 )</td>
</tr>
<tr>
<td></td>
<td>Specification Uncertainty</td>
<td>0.285 * (0.122)</td>
</tr>
<tr>
<td></td>
<td>Design Complexity</td>
<td>-0.184 (0.165)</td>
</tr>
<tr>
<td></td>
<td>Software Process Maturity</td>
<td>0.149 (0.137)</td>
</tr>
<tr>
<td>Client Validation Quality (-1 * Defects/KLOC)</td>
<td>Software Size</td>
<td>0.941 *** (0.070)</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>---------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Repeated contracting {Regime 1}</td>
<td>0.069 ***     (0.012)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 2 relative to Regime 1}</td>
<td>-0.090 *** (0.015)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 3 relative to Regimes 1 &amp; 2}</td>
<td>0.202 *** (0.034)</td>
<td></td>
</tr>
<tr>
<td>Intercept</td>
<td>-6.637 ***    (0.391)</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>Wald $\chi^2 = 267.27$, $p &lt; 0.001$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cost Performance</th>
<th>Specification Uncertainty</th>
<th>-32.780 *** (9.092)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Complexity</td>
<td>-72.864 *** (13.891)</td>
<td></td>
</tr>
<tr>
<td>Software Process Maturity</td>
<td>26.101 * (13.545)</td>
<td></td>
</tr>
<tr>
<td>Software Size</td>
<td>11.967 (8.045)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 1}</td>
<td>-23.676 *** (3.132)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 2 relative to Regime 1}</td>
<td>28.638 *** (3.360)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 3 relative to Regimes 1 &amp; 2}</td>
<td>-3.089 (2.952)</td>
<td></td>
</tr>
<tr>
<td>Intercept</td>
<td>41.410 (58.854)</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>Wald $\chi^2 = 163.27$, $p &lt; 0.001$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schedule Performance</th>
<th>Specification Uncertainty</th>
<th>-5.668 * (2.690)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Complexity</td>
<td>-10.483 * (4.115)</td>
<td></td>
</tr>
<tr>
<td>Software Process Maturity</td>
<td>1.576 (4.023)</td>
<td></td>
</tr>
<tr>
<td>Software Size</td>
<td>7.085 ** (2.381)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 1}</td>
<td>-1.818 * (0.926)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 2 relative to Regime 1}</td>
<td>2.798 ** (0.994)</td>
<td></td>
</tr>
<tr>
<td>Repeated contracting {Regime 3 relative to Regimes 1 &amp; 2}</td>
<td>-0.017 (0.823)</td>
<td></td>
</tr>
</tbody>
</table>
We had predicted in Hypothesis 3 that repeated contracting is positively associated with higher design verification quality (H3a); higher client validation quality (H3b); better cost performance (H3c); and better schedule performance (H3d). However, our results do not unilaterally support this hypothesis. As is suggested by the numbers reported in column 3 of Table 2 there are some fascinating and differential patterns of relationships between repeated contracting and performance outcomes in the three contract regimes. Repeated contracting does not appear to improve performance in all contract regimes.

To investigate these differences further, we used the estimated slope coefficients for repeated contracting to compute the slopes in the different contracting regimes. The slope of repeated contracting is computed for Regime 1 = $\beta_7$, the slope for Regime 2 = $\beta_7 + \beta_8$, and the slope for Regime 3 = $\beta_7 + \beta_8 + \beta_9$. Each regime also has a different intercept or constant term which must be adjusted at each spline knot to accommodate the change in the slope. This keeps the regression line continuous even when the regression line changes direction at each knot. The intercept term for contract Regime 1 = the estimated intercept in the model. The intercept term for the contract Regime 2 = the estimated model intercept – (#contracts in Regime 1 * estimated slope coefficient of the increment for repeated contracting in Regime 2). The intercept term relevant for contract Regime 3 = estimated model intercept – (#contracts in Regime 1 * estimated slope coefficient of the increment for repeated contracting in Regime 2) – (#contracts in Regimes 1 and 2 * estimated slope coefficient of the increment for repeated contracting in
Regime 3). The computed slopes and intercepts for each contracting regime were used to generate the graphs shown in Figures 2a-2d.

**Figure 2a. Design Verification Quality Spline**

![Design Verification Quality Spline](image)

**Figure 2b. Client Validation Quality Spline**

![Client Validation Quality Spline](image)

**Figure 2c. Cost Performance Spline**

![Cost Performance Spline](image)

**Figure 2d. Schedule Performance Spline**

![Schedule Performance Spline](image)

As can be seen in Figures 2a-2d, performance erodes in Regime 1 (Hybrid contracts) with repeated contracting for all outcomes except for client validation quality in which performance improves. Regime 2 (FP contracts) has the opposite pattern in which performance improves with repeated contracting for all outcomes except for client validation quality in which performance erodes. Finally, for Regime 3 (T&M contracts) performance improves with repeated contracting for all outcomes except for design verification quality in which performance erodes. These differences in repeated contracting slopes across regimes are significant for each of the performance outcomes (Slope Regime 1 vs. Slope Regime 2 vs. Slope Regime 3: for Design
Verification Quality, $\chi^2 = 7.93$, $p < 0.05$; for Client Validation Quality, $\chi^2 = 57.04$, $p < 0.001$; for Cost Performance, $\chi^2 = 72.82$, $p < 0.001$; for Schedule Performance, $\chi^2 = 8.14$, $p < 0.05$). We interpret and discuss our results in the Discussion Section.

**DISCUSSION**

In this study, we hypothesized contract type and repeated contracting as predictors of key performance outcomes in software development outsourcing. Leveraging our access to a detailed, longitudinal archive of contracts and performance data spanning 1987-2006 between a major client and a major vendor, we investigate the effects of three forms of contract types - FP, Hybrid and T&M - on quality, cost and schedule performance outcomes. We also used spline regression analyses to diagnose the effects of repeated contracting within each contract type regime on quality, cost and schedule performance. A number of critical findings can be gleaned from the findings of this study to help us understand the consequences of governance decisions associated with software development outsourcing. Table 3 summarizes the findings.

**Table 3. Summary of Results**

<table>
<thead>
<tr>
<th>Outcomes for Contract Type (marginal means)</th>
<th>FP</th>
<th>Hybrid</th>
<th>TM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Verification Quality (Defects per KLOC)</td>
<td>H1a 1.392</td>
<td>2.025</td>
<td>1.428</td>
</tr>
<tr>
<td>Client Validation Quality (Defects per KLOC)</td>
<td>H1b 5.118</td>
<td>4.505</td>
<td>3.030</td>
</tr>
<tr>
<td>Cost Performance (Value of Variance)</td>
<td>H2a 25.250</td>
<td>-288.700</td>
<td>1.313</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>With Repeated Contracting (estimated slopes)</th>
<th>H3a 0.023</th>
<th>H3b -0.021</th>
<th>H3c 4.962</th>
<th>H3d 0.980</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Verification Quality</td>
<td>Improved</td>
<td>Eroded</td>
<td>Improved</td>
<td>Improved</td>
</tr>
<tr>
<td>Client Validation Quality</td>
<td>Eroded</td>
<td>Improved</td>
<td>Improved</td>
<td>Improved</td>
</tr>
<tr>
<td>Cost Performance</td>
<td>Improved</td>
<td>Eroded</td>
<td>Improved</td>
<td>Improved</td>
</tr>
<tr>
<td>Schedule Performance</td>
<td>Improved</td>
<td>Eroded</td>
<td>Improved</td>
<td>Improved</td>
</tr>
</tbody>
</table>
First, our findings suggest that consistent with our hypotheses, none of the three contract types actually led to optimal performance on all dimensions. FP contracts have the best- while Hybrid contracts yield the worst- outcomes on the technical dimensions of software development, i.e., design verification quality, cost performance and schedule performance. By contrast, T&M contracts and Hybrid contracts produce better outcomes on client validation quality compared to FP contracts.

These patterns of results can be explained by considering who bears the financial risks for rework and requirements changes. In T&M contracts, the client bears all financial risks, while the vendor bears the financial risks in FP contracts. Thus, in an FP contract, vendors are under greater financial discipline to avoid re-work, and this leads to higher design verification quality. However, because of the “experience” nature of software, a software module that is designed and developed to the initial or original design specifications may not be as high in client acceptance as one that evolves or emerges as the software develops. In contrast, for T&M contracts, vendors are not under as great of a financial discipline to avoid re-work and requirements changes since the client is paying. We observe that even after controlling for specification uncertainty and design complexity, client validation quality is higher in software created under T&M contracts. Finally, although Hybrid contracts have intermediate levels of client validation quality, Hybrid contracts have the lowest levels of design verification quality, and cost and schedule performance. Since neither the vendor nor the client bears all financial risks for technical problems in Hybrid contracts, it appears as though neither party has a strong enough incentive to closely monitor the process and take corrective actions when projects go awry. This can be especially problematic if even small deviations from target escalate and become difficult to correct.
Second, our findings show that contrary to our expectations, repeated contracting between client and vendor does not necessarily improve all performance outcomes. In fact, the patterns of findings clearly suggest that repeated contracting serves more to "reinforce" the incentives associated with a particular contract type. As suggested in Figures 2a-2d, if a contract type provides an incentive scheme that supports a specific set of behaviors, then repeated contracting serves to reinforce only the same set of the specific behaviors. On the other hand, if a contract type does not support another set of behaviors, the lack of reinforcement or incentives sends a powerful message that poor performance on this particular dimension would not be penalized, and therefore no corrective action is necessary. Specifically, for high powered incentive contracts such as FP contracts, the client enjoys cost certainty and could guard against misappropriation by the vendor. Since the vendor in FP contracts is the residual claimant who enjoys surplus from high design verification quality and cost savings effort, repeated contracting then serves to strengthen the same incentives for the vendor to exert greater effort to improve on the cost and schedule performance, and on design verification quality. However, we would not expect vendors to improve on client validation quality in FP contracts given that the costs and risks of additional rework are borne solely by the vendor. In contrast to FP contracts, both Hybrid and T&M contracts are regarded as low-powered incentive contracts. In low-powered incentive contracts, the vendor has less incentive to reduce costs incurred to clients as the contract risks are borne by the client. Since costs of rework are borne by the client, client validation quality tends to improve in low powered Hybrid and T&M contracts.

**CONCLUSIONS**

This study has several strengths. First, this paper presents empirical evidence on the consequences of contract type on a wide range of performance outcomes. While prior research on contract choice has made the assumption that observed contract designs represent
optimal/equilibrium outcomes of competition, this study actually offers concrete evidence of outcomes. Second, this paper used a valuable and rich data set on contracts and longitudinal performance data which enabled us to directly measure objective performance outcomes of quality, cost and schedule performance. Third, from the perspective of research design, this setting is ideal for a study of performance outcomes in a repeated contracting setting. By adopting a rigorous longitudinal design and considering the repeated interaction between one client and one vendor, we were able to trace the complete chain of effects of contract types on performance outcomes in repeated contracting. This analysis revealed that while contract types may be efficiently chosen with respect to *ex ante* uncertainties, these same types may not necessarily lead to optimal performance outcomes. Further, our results suggest the power and limitations of repeated contracting as a mechanism to influence contracting behaviors as repeated contracting reinforces the incentives already inherent in a contract type.

As no study is definitive, future studies can extend our research. Our findings derive from the particular context examined. Although it would certainly be instructive for future research to consider other contexts, we actually consider our research design to be a key strength of the paper. Our use of archival measures collected from different sources and audited for validity and reliability by both client and vendor increases the internal validity of our findings. As such, the data are robust and not subject to self-report or social desirability bias symptomatic of typical survey studies of performance. By studying the repeated contracting between one vendor and client in one software development setting we are able to control naturally for client, vendor and software characteristics (*e.g.*, vendor size, relative bargaining power, application domain, *etc.*) that would otherwise have to be controlled for statistically and incorporated into the models. In addition, studying the repeated contracting between a vendor and client over time enabled us to discern the complete chain of effects from contracting decisions to performance outcomes.
Nevertheless, if data were drawn from multiple clients and multiple vendors, our model could be extended by considering client and vendor characteristics as potential main effects or moderators to contract types and the relationship between types and performance outcomes.

The findings from our study have important implications for research and practice in software development. The foremost implication concerns the critical insight that choice of contracting type (FP, Hybrid or T&M) should not only consider *ex ante* uncertainty factors, but also should embrace the likely software development outcomes of contracting types. As we found, none of the three contract types of FP, Hybrid or T&M led to optimal outcomes on all performance dimensions. While we had speculated *a priori* that repeated contracting could mitigate some of the risks associated with contract type, the evidence from this study shows that repeated contracting serves only to reinforce the same incentive patterns inherent in each contract type. A key implication of this finding is that the client and vendor need to be cognizant of the tradeoffs between performance outcomes inherently embedded in the financial incentives of different contracts at the point of contract negotiation. It can be difficult for decision makers in software development contexts to discern the downstream performance implications of earlier development decisions (Sterman et al. 1997). Due diligence, therefore, is necessary to prioritize performance outcomes in the negotiation stage so that contractual parties are not surprised by the path dependency of consequential performance outcomes arising from different contract types.

**REFERENCES**


## APPENDIX

Table A1. Descriptive Statistics and Correlation Matrix for Quality Panel (n=103)  
(Mean with std. dev. in parentheses; Pearson correlation coefficients with p-values in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Mean (s.d.)</th>
<th>1.</th>
<th>2.</th>
<th>3.</th>
<th>4.</th>
<th>5.</th>
<th>6.</th>
<th>7.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.Specification Uncertainty</td>
<td>3.15 (.49)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.Design Complexity</td>
<td>2.46 (.62)</td>
<td>-.030 (.762)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.Contracting Experience</td>
<td>53.30 (34.23)</td>
<td>.199 (.044)</td>
<td>.060 (.547)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.Process Maturity</td>
<td>2.49 (.84)</td>
<td>-.041 (.679)</td>
<td>.173 (.080)</td>
<td>.530 (.000)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.Software Size</td>
<td>154.09 (262.88)</td>
<td>.096 (.337)</td>
<td>.027 (.785)</td>
<td>.530 (.654)</td>
<td>.004 (.971)</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.Verification Quality</td>
<td>1.11 (1.15)</td>
<td>.058 (.568)</td>
<td>.047 (.643)</td>
<td>.091 (.367)</td>
<td>-.053 (.598)</td>
<td>.221 (.027)</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>7.Validation Quality</td>
<td>1.67 (2.65)</td>
<td>.077 (.442)</td>
<td>-.017 (.870)</td>
<td>-.011 (.915)</td>
<td>.017 (.864)</td>
<td>.221 (.026)</td>
<td>.066 (.518)</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table A2. Descriptive Statistics and Correlation Matrix for Cost/Schedule Panel (n=939)  
(Mean with std. dev. in parentheses; Pearson correlation coefficients with p-values in parentheses)

<table>
<thead>
<tr>
<th></th>
<th>Mean (s.d.)</th>
<th>1.</th>
<th>2.</th>
<th>3.</th>
<th>4.</th>
<th>5.</th>
<th>6.</th>
<th>7.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.Specification Uncertainty</td>
<td>3.16 (.49)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.Design Complexity</td>
<td>2.28 (.53)</td>
<td>-.250 (.000)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.Contracting Experience</td>
<td>64.19 (32.97)</td>
<td>.280 (.000)</td>
<td>.021 (.520)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.Process Maturity</td>
<td>2.65 (.69)</td>
<td>-.035 (.288)</td>
<td>.317 (.000)</td>
<td>.318 (.000)</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.Software Size</td>
<td>124.32 (221.58)</td>
<td>-.016 (.622)</td>
<td>.220 (.000)</td>
<td>-.173 (.000)</td>
<td>.014 (.677)</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.Cost Performance</td>
<td>-60.42 (282.79)</td>
<td>.068 (.036)</td>
<td>-.120 (.000)</td>
<td>.248 (.000)</td>
<td>.002 (.946)</td>
<td>-.089 (.006)</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>7.Schedule Performance</td>
<td>-19.65 (79.89)</td>
<td>.068 (.038)</td>
<td>-.072 (.027)</td>
<td>.200 (.000)</td>
<td>.035 (.285)</td>
<td>.013 (.688)</td>
<td>.480 (.000)</td>
<td>1.000</td>
</tr>
</tbody>
</table>