OPTIMAL SOFTWARE OUTSOURCING CONTRACT UNDER ASYMMETRIC INFORMATION: BUNDLING DEVELOPMENT AND MAINTENANCE SUPPORT

Qiang Zeng, Business School, Shantou University, Guangdong, China, qzeng@stu.edu.cn
Shivendu Shivendu, Merage School of Business, University of California, Irvine, USA, sshivendu@uci.edu

Abstract

Recent research has called for studying optimal outsourcing contracts when some aspect of the quality of software is observed but is not verifiable. In this paper we model a software application outsourcing relationship wherein a client offers contracts for development of application software and providing maintenance support over a period of time. Quality of delivered software is impacted by intrinsic efficiency and effort, but is not contractible. We show that the client uses a development and support bundling contract as a screening device and second-best efficiency is achieved under certain conditions even in the presence of adverse selection. Counter to intuition, we find that the client may induce a shorter maintenance support period for a higher quality software application. Surprisingly, under information asymmetry and verifiable quality, unbundling contracts may enable the client to induce higher quality and longer support period.

Keywords: Outsourcing, adverse selection, bundling development and maintenance.
1 INTRODUCTION*

It is well known that firms outsource software applications development to external vendors to focus on their core competency and take advantage of the cost savings due to a vendor’s economies of scale and scope (Lichtenstein 2004; Wang et al. 1997; Dey et al. 2010). However, it is less clear what factors affect the decisions on software maintenance or support given that the cost of maintenance is usually substantial to the vendors (Lichtenstein 2004), and software quality is often non-verifiable (Gurbaxani et al 2011). Further, Lichtenstein (2004) found that firms make heterogeneous choices on system maintenance for the software applications developed externally and firms are more likely to maintain systems internally when the risks of being locked in a costly relationship are high. While transaction cost economics concludes that firms engage in longer contracts in the presence of relationship-specific investments, we are not able to fully explain how the length of the support period of a software application is determined.

When a government agency contracts a private vendor to develop an e-Government application, several models of partnership between the government agency and the vendor are feasible, depending on the nature of the particular project, including build, own and operate (BOO), build operate and transfer (BOT), build own operate and transfer (BOOT). When choosing a particular model of the so-called public-private partnership, the primary consideration is to combine public sector accountability with private sector efficiency, and to ensure risk sharing (Sharma 2007). What factors determine the form of the contracts between two firms when both software application development and maintenance support activities are involved?

While scholars (Whang 1992; Dey et al. 2010; Wang et al. 1997; Richmond & Seidman 1992; Gopal et al. 2003) have studied the dynamics of software application outsourcing, it is not clear why the duration of application maintenance is heterogeneous and what factors determine how long an externally-developed software application would be supported by the developer. In light of the key characteristics of a typical IT outsourcing relationship (Gurbaxani et al. 2011), further investigation is needed to shed light on this issue.

A client firm receives value from both the quality of the software application, which is often non-verifiable, and maintenance support offered by the external developer. The quality of the software application may be determined by the developer’s intrinsic efficiency since higher quality-improving effort can be exerted by an efficient developer during the project execution. Intuitively, a developer of high efficiency may find that (1) a longer period of support may signal his superior quality under information asymmetry and thus help to attract clients and/or to charge a premium for his products and (2) it is more profitable for him to offer a longer period of support due to the higher efficiency in providing such support. More interestingly, when we add quality and initial investment into the dynamics, how the length of support period is determined becomes more complicated. The client may prefer to induce the optimum level of the initial investments from the developer since the investments directly increase the quality of the software system. The investments may also decrease the developer's cost of providing support and therefore it may be less costly for the developer to offer a support of longer period. On the other hand, the client may find it feasible to utilize the length of support period as a strategic device to select a developer of high efficiency and thus offer a menu of contracts which may achieve self-selection, namely, a developer of certain level of efficiency may find it economically optimal to select the contract that is “meant” to him and by selecting the contract, the developer also reveals his true level of efficiency to the client.

In this paper, we develop a principal-agent model in which quality of software application is non-verifiable, there is ex ante uncertainty about value of software application to the client firm, and the developer’s unobservable effort or investment affects quality of the application software. We show that bundling development and maintenance support contracts can serve as a screening device for the client to select more efficient vendors. The optimal screening requires the client pays more for long period of maintenance through a fixed-price contract for the development phase of the relationship.

* This work was partially supported by China Guangdong Philosophy and Social Science Planning Grant GD12CGL11
which produces a high-quality software application. Counter-intuitively, the client may induce a shorter maintenance support period for a higher quality software application. We also find that bundling development and support service contracts mitigates the inefficiencies resulting from information asymmetry and non-verifiable quality while bundling achieves the first best efficiency when there is no information asymmetry and quality is verifiable. Under information asymmetry and verifiable quality, unbundling contracts may enable the client to induce higher quality and longer support period.

In the IS literature, scholars have focused on the relative costs of developing software applications internally and externally and what factors alter the cost equation. Wang et al. (1997) show that when internal and external developers have identical cost functions, outsourcing yields the lower net value and that greater uncertainty about the development costs makes outsourcing less attractive. Lichtenstein (2004) found that firms have often chosen fixed price contracts in software application outsourcing. More interestingly, he also found that firms have heterogenous choices on system maintenance for the software applications developed externally and they are more likely to maintain systems internally when the risks of being locked in a costly relationship are high. Dey et al. (2010) studies how information asymmetry affects the relative efficiency of different types of contracts in software outsourcing. They find that fixed-price contracts are better for simple software projects while time-and-materials contracts are better for more complex projects. In the present paper, we focus on the factors determining the length of the support period for the software applications developed by external vendors. If the quality is verifiable, then as long as quality and effort are deterministic, the client can achieve first-best. When quality is non-verifiable then no incentive contract can be designed as there is only one contracting variable. Client fails to induce effort. We find that bundling maintenance with software development project, the client can achieve second-best under certain conditions.

2 A MODEL OF SOFTWARE OUTSOURCING

A firm considers outsourcing a software application development and maintenance project to an external developer. The value of the outsourced software application to the firm consists of quality of the software \( q \) and the length of free support offered by the developer \( t \). The external developer’s quality-improving effort \( x \) during the development phase directly contributes to quality \( q \). Without loss of generality, we set \( q = x \). The quality-improving effort exerted by the developer in the software development stage is unobservable. To simplify to model, we replace \( q \) with \( x \), and remove \( q \) from all expressions whenever it enhances clarity.

The total value of the software to the client is \( V = r(x+t) \); where \( r \in \{R,0\} \) such that \( \text{prob}[r = R] = \rho \). Binary random variable \( r \) captures the uncertainty faced by the firm: with probability \( 1 - \rho \) the software application may not be valuable to the firm at all after the delivery. The uncertainty about value of software application is resolved in the interim, after the development stage.

External developers are heterogeneous in their innate efficiency parameter \( \theta \) which affects the cost of application development and also cost of maintenance after the software is delivered. Lower \( \theta \) indicates lower cost parameter and therefore, the developer with lowest (highest) \( \theta \) is most efficient (inefficient). Under our formulation, the developer’s type \( \theta \) is his private information and without loss of generality, we assume that there are two types of developers in the population, \( \Theta = \{\theta_L, \theta_H\} \), where \( \theta_L < \theta_H \). The distribution of efficient type \( \theta_L \) and inefficient type \( \theta_H \) is common knowledge. The developer can be either efficient (\( \theta_L \)) or inefficient (\( \theta_H \)) with probabilities \( \nu \) and \( 1 - \nu \) respectively.

The developer’s cost of application development \( C_p = \psi(\theta, x) \) is increasing and convex in quality-improving effort \( x \), and in developer’s innate efficiency parameter \( \theta \). For tractability we assume \( C_p = \psi(\theta, x) = \theta x^2 \). Note that \( \psi(\theta, 0) = 0 \forall \theta \). The external developer’s cost of providing maintenance support for period \( t \) is \( C_s = \phi(\theta, x, t) \). If the quality of delivered software application is high, then the
cost of maintenance support is low, i.e., cost of maintenance support is decreasing in quality \( q \). If any developer exerts more quality improving effort at the developmental stage of the application, the quality of software application increases leading to decrease in maintenance cost. For tractability, following Dey et al (2010), we assume the support cost as \( C_s = \phi(\theta, x, t) = \theta t^2 - xt \).

We make following assumptions.
1. Cost of maintenance effort exerted by the developer is homogeneous and is normalized to zero.
2. Vendor firms have same innate efficiency parameter for development of software and for maintenance support.
3. The outsourcing firm and vendors are risk neutral.
4. The innate efficiency parameter of the most efficient type external developer is not too low, i.e., \( \theta_L > 1/2 \). This assumption excludes the trivial solution where software quality is infinite under certain conditions.

When client offers a development and support contract \( \{p(\theta), t(\theta)\} \), a developer of type \( \theta \) accepts the contract and exerts effort of \( x(\theta) \), the software quality is \( q(\theta) = x(\theta) \) and client’s payoff writes, 
\[ \Pi_C = rx(\theta) + rt(\theta) - p(\theta) \]

The developer’s expected payoff writes,
\[ U_D(\theta) = p(\theta) - \psi(\theta, x(\theta)) - \phi(\theta, x(\theta), t(\theta)) = p(\theta) - \theta x^2(\theta) - \theta t^2(\theta) + x(\theta)t(\theta) \]

The client offers a menu of contracts because type of developer is his private information and client does not know the type of the developer with whom she is going to contract with. The client has to design the menu of contracts to induce truthful self-selection by the developers. Since \( \Theta = \{\theta_L, \theta_H\} \), we denote the menu of contracts by a duplet \( \{(p_L, t_L), (p_H, t_H)\} \) and the client analyzes the payoff from any menu of contracts in expected terms. Therefore, the client’s optimization problem writes as:
\[ \max_{\{p_L, t_L\}} \{ v(rx_L + rt_L - p_L) + (1 - v)(rx_H + rt_H - p_H) \} \]

In the next section, we develop optimal contracts when there is no information asymmetry and software quality is verifiable.

3 OPTIMAL SOFTWARE OUTSOURCING CONTRACTS WITH VERIFIABLE QUALITY

First, we examine the contracts (unbundled and bundled) when the quality is verifiable and the developer’s type \( \theta \) is common knowledge, that is, there is no information asymmetry and hence no adverse selection issue. Note that under unbundled contract the client offers potentially only one or two sequential contracts. First, she offers the development phase contract and offers no maintenance support contract if \( r = 0 \). If realized \( r = R \), then she offers the maintenance support contract. We normalize \( R = 1 \) to focus on the impact of information asymmetry on contract bundling.

3.1 Unbundling contracts

In the application development contracting stage, the client’s objective is to induce the efficient level of effort (quality) from the developer given that the development cost is \( C_D = \psi(\theta, x) = \theta x^2 \). Her optimization problem simply writes \( \max_x (\rho x - \theta x^2) \) since \( q = x \). Let the optimal quality contingent development contract specify \( q = q^* \). The principal would continue to proceed with the maintenance phase contracting only when \( r = 1 \). In the application maintenance contracting stage, the client’s objective is to set the optimal length of support period. Her optimization problem writes as \( \max_t (t - \theta t^2 + q^* t) \). Lemma 1 specifies the optimal development phase and support phase contracts.

**LEMMA 1:** With unbundling contracts, under verifiable quality and no adverse selection, the first-best contract is characterized by the following properties:(i) the optimal quality of software is:
One might think that since the value uncertainty of outsourced software application is resolved in the interim before the contract for the support phase, probability \( \rho \) will play no role in optimal support period in *unbundling contracts*. But counter to intuition, in Lemma 1, we find that optimal support period, \( t^{*} \), is increasing in \( \rho \). The impact of \( \rho \) on \( t^{*} \) is indirect and it is through the optimal quality \( q^{*} \). As \( \rho \) increases, optimal quality \( q^{*} \) increases, this leads to decrease in the marginal cost of support duration. But the marginal benefit of support duration remains the same, leading to increase in optimal support duration. Similarly, profit of the client firm and the price paid to the firm providing software development services and to the firm providing maintenance support services is increasing in \( \rho \), and decreasing in efficiency parameter \( \theta \).

### 3.2 Bundling contracts

When quality is verifiable, the client offers a *bundling contract* consisting of a triplet \( \{q^{B}, p^{B}, t^{B}\} \). Under no information asymmetry, client observes \( \theta \) and induces quality-improving effort \( x \) via a quality-contingent contract. The client always chooses the most efficient vendor (lowest \( \theta \)) amongst all those who participate. The client’s optimization problem writes as \( \max_{x, p, t} (\rho(x + t) - p) \). Since there is no information asymmetry and \( q = x \), the client offers always offers price such that vendor gets his outside option which is assumed to be zero, that is, \( p = \theta x^{2} + \theta t^{2} - xt \).

**LEMMA 2:** Under verifiable quality and no adverse selection, when the client firm offers bundling contracts, the first-best contract is characterized by the following properties: (i) the optimal quality of software is: \( q^{B*} = \frac{\rho}{2\theta - 1} \); (ii) the optimal support duration is: \( t^{B*} = \frac{\rho^{2}}{2\theta - 1} \); (iii) The vendor receives zero payoff: \( U_{D} = 0 \); and (iv) The client’s expected payoff is: \( \Pi_{C}^{B*} = \frac{\rho^{2}}{2\theta - 1} \).

Similar to the case with unbundling contracts, the optimal quality, the optimal support duration, and the client’s expected payoff are increasing in value of the software application \( \theta \) and measure of value uncertainty \( \rho \), and decreasing in efficiency parameter \( \theta \).

### 3.3 Comparing unbundling and bundling contracts under benchmark case

**PROPOSITION 1:** When quality is verifiable and there is no adverse selection, (i) the optimal quality-improving effort exerted by the vendor in the development phase and the optimal quality of software is always higher with bundling contract; (ii) the optimal length of support period is higher
with bundling contract when $\rho > \frac{4\theta^2 - 2\theta}{4\theta^2 - 2\theta + 1}$; and (iii) the client is better off with bundling contract when $(2\theta - 1)\rho^2 - 4\theta(2\theta^2 - \theta + 1)\rho + 4\theta^2(2\theta - 1) < 0$.

The downside of the unbundling contract is that the client is not able to offer a contract in the development phase that takes into consideration the potential benefits of higher quality in the development phase in terms of lower cost in the maintenance support phase. On the other hand, the upside of the unbundling contract is that the client saves cost of useless maintenance support during the support phase if the realized value of random variable $r = 0$ in the interim. The upside of bundling contract is that the client is able to fine tune the optimal quality of the software by being forward looking and economizing on support cost, and the downside is that the cost of support as well as the cost of developing higher quality is wasteful when $r = 0$. Since the potential benefits and costs of bundling contract compared to unbundling contract are determined by the efficiency parameter $\theta$ and measure of value uncertainty $\rho$, results on Proposition 1 are driven by these tradeoffs.

Proposition 1 (i) implies that the client always contracts for higher quality software under bundling contract compared to unbundling contract. The observation regarding quality-improving effort directly follows from our conceptualization that vendor needs to exert higher effort to deliver higher quality software ($q = x$). The optimal support period under unbundling and bundling contract is determined by the efficiency parameter $\theta$ as well as by the measure of value uncertainty $\rho$. Under unbundling contract, the client offers the maintenance contract only when realized value of $r = 1$. The condition in Proposition 1(ii) implies that the marginal benefit of support under bundling contract is higher than under unbundling contract, which leads to higher support duration under bundling contract. The locus of point where optimal support period under unbundling contract and bundling contract is the same, that is, $t^{UB^*} = t^{B^*}$, is shown in the $\rho - \theta$ space in Figure 1a.

Note that for any given measure of value uncertainty, $\rho$, as $\theta$ increases, that is, the vendor is less efficient, the loss due to wasteful support cost (when realized $r = 0$) outweighs the saving from lower support cost due to higher quality under bundling contract and the optimal support period under bundling is lower compared to unbundling contract. Similarly, for any given efficiency parameter of the vendor, as measure of value of uncertainty decreases, that is the probability $\rho$ increases, the expected loss due to wasteful support in the maintenance support phase decreases and the client’s optimal support period under bundling is longer than that under unbundling contract.

The vendor’s investment in terms of quality-improving effort has two effects: 1) directly increasing the value that the client receives from the software; 2) decreasing the vendor’s cost of providing support and thus indirectly increasing the value resulting from the length of application support (with scalar $r$). A bundling contract enables the client to consider both of the effects simultaneously and that may be beneficial to her. But this upside of bidding needs to be balanced against the potential downside where the cost of maintenance support is wasteful when realized $r = 0$. Therefore, when choosing between unbundling contracts and bundling contracts, the trade-offs involve realizing the savings on support costs, when realized $r = 0$, with unbundling contracts and getting the efficiency gains due to higher quality with bundling contracts.

For any given measure of value uncertainty $\rho$, as efficiency parameter $\theta$ increases, the savings in wasteful support costs with unbundling contracts outweighs the efficiency gains due to higher quality with bundling contracts and the client is better off with unbundling contracts. Similarly, for any given efficiency parameter $\theta$ as value uncertainty decreases ($\rho$ increases), potential gain from higher quality outweighs potential loss due to wasteful maintenance support and the client is better off with bundling contract. The intuition is that the forward-looking feature of bundling contacts that generates substantial efficiency outweighs cost savings with unbundling contracts as the probability that the software application will be valuable to the client increases. Figure 3b shows the region in $\rho - \theta$ space where bundling contract dominates unbundling contract.
When there is no uncertainty bundling contract is always better for the client due to (i) higher quality is induced to take advantages of more cost savings in the support phase, and (ii) the downside of bundling in terms of potential wasteful maintenance support when \( r = 0 \) vanishes. More generally, the investment in development (thus quality) is higher with bundling contracts, reflecting the fact that the effect of quality on efficiency (via savings on support costs) is internalized in the principal’s optimization while such effect is not considered with unbundling contracts.

Interestingly, we also find that when value uncertainty and vendor’s efficiency are medium, the client is better off with bundling contract where the support period is lower compared to unbundling contract (Figure 1 c). The economic intuition for this surprising result that when both \( \theta \) and \( \rho \) are medium, the efficiency gains with bundled contract outweigh the savings in support costs with unbundled contract even though the support period is longer with unbundling contracts.

Figure 1. Comparison of bundled and unbundled contracts under the benchmark case

In the next section, we consider a more general case where the vendor-specific efficiency parameter \( \theta \) is vendor’s private knowledge, and therefore the client must design optimal contracts taking adverse selection into consideration.

### 4 Optimal Contracts in Presence of Adverse Selection

When there is asymmetric information, the vendor knows his efficiency parameter \( \theta \), but the client knows only the distribution. Intrinsic efficiency parameter directly impacts the cost of development of software of any given quality and maintenance support for any given period and it indirectly impacts the quality of software because as \( \theta \) increases marginal cost of producing quality increases. The client always prefers to contract with an efficient type developer.

#### 4.1 Unbundling contracts

When the vendor’s efficiency parameter (type) is vendor’s private knowledge and the client firm only knows the distribution of efficiency parameter \( \theta \), the client offers a menu of self-selection contracts. These contracts must ensure that vendors’ truthfully choose the contract designed for their type and the contract should induce efficient quality-improving effort by the vendor for the development task. The client needs to pay information rent to the efficient type vendor to incentivize him to reveal his true efficiency parameter \( \theta \) and in order to economize on the information rent, the client firm may lower the quality of software for the inefficient vendor. The optimal second-best contract is designed by the client by making tradeoff between information rent and efficiency.

**Lemma 3:** Under information asymmetry and verifiable quality, (i) in the development phase the optimal software quality pair is: \( \begin{align*}
q^{SB}_L &= \frac{\rho}{2\theta_L}, \\
q^{SB}_H &= \left(1 - \nu\right)\frac{\rho}{2(\theta_H - \nu\theta_L)}
\end{align*} \); (ii) The client’s total payoff from software development and maintenance support is:

\[
\Pi^{SB}_C = \frac{\rho}{4} \left(\nu \left(1 + \frac{\rho}{2\theta_L}\right)^2 + (1 - \nu) \left(1 + \frac{(1 - \nu)\rho}{2(\theta_H - \nu\theta_L)}\right)^2 + \rho \left(\nu \frac{\nu}{\theta_L} + \left(1 - \nu\right)^2\right)\right).
\]

Note that the second-best contract for the development phase requires the efficient vendor (\( \theta_L \) type) to deliver the same quality as in Lemma 1 \( (q^*_L = q^{SB}_L) \) though the quality prescribed for the inefficient
vendor (\( \theta_H \) type) is strictly lower than that in Lemma 1 (\( q_{_{HB}}^* < q_{_{HB}}^* \)). As the proportion of efficient vendors in the population increases, the quality distortion for the inefficient vendor (\( q_{_{HB}}^* - q_{_{HB}}^* \)) increases to economize on information rent. Similarly, as the inefficient vendor becomes more inefficient, that is \( \theta_H \) increases for any given \( \theta_L \), quality offered to inefficient type vendor is more distorted. On the other hand, as the probability of software application being valuable to the client \( (\rho) \) increases, the quality distortion also increases. Since the information rent paid to the efficient vendor in optimal contract increases as the quality offered to the inefficient vendor increases, the client sacrifices efficiency by lowering contract quality for inefficient vendor from efficient level, \( q_{_{HB}}^* \) in order to save the information rent to the efficient type vendor. For the software application development, the client offers a menu and when no efficient vendor participates in contracting then the software development is undertaken by the inefficient vendor. Therefore, now we design the optimal maintenance support contract keeping in view the possibility that the quality of software may be \( q_{_{BL}}^* \) if software development was undertaken by the efficient vendor, or it may be \( q_{_{HB}}^* \) if the software development was undertaken by the inefficient vendor.

When in the development phase the client contracts with an efficient vendor then the first-best quality \( (q_{_{BL}}^* = q_{_{BL}}^*) \) of software is delivered. Therefore, when the client contracts with an efficient vendor for maintenance support there is no distortion in the optimal maintenance support period for the efficient vendor, that is \( t_{_{BL}}^* = t_{_{BL}}^* \). On the other hand, even when the client contracts with an efficient vendor in the development phase and the first-best quality software is delivered, she offers lower maintenance support period than the first-best when she contracts with an inefficient vendor for maintenance support, that is \( t_{_{BL}}^* < t_{_{BL}}^* \). The client pays information rent to the efficient vendor which is increasing in maintenance support period offered to the inefficient vendor, \( t_{_{BL}}^* \) and the difference in the efficiency parameter of efficient and inefficient vendors, that is, \( \Delta \theta = \theta_H - \theta_L \).

When in the development phase, the client contracts with an inefficient vendor then quality of software is \( q_{_{BL}}^* \) and cost of maintenance support is higher for both type of vendors because \( \partial C_S / \partial q < 0 \). Therefore, the maintenance support period for the efficient vendor is distorted downwards by the client because quality of software is below the first-best level, \( q_{_{BL}}^* < q_{_{BL}}^* \). This leads to lower maintenance support period offered to the efficient vendor. In order to economize on information rent to be paid to the efficient vendor during the maintenance support phase, the client sacrifices efficiency and further lowers the maintenance support period for the inefficient vendor, that is \( t_{_{BL}}^* < t_{_{BL}}^* \).

**PROPOSITION 2:** The optimal maintenance support are ranked in the following order where support duration is longest when software development is undertaken by an efficient vendor and maintenance support is also undertaken by an efficient vendor:

\[
\begin{align*}
& t_{_{HL}}^* | q_{_{HB}}^* < t_{_{HL}}^* | q_{_{LB}}^* < t_{_{HL}}^* | q_{_{LB}}^* = t_{_{HL}}^*. 
\end{align*}
\]

Interestingly, we see that the support duration offered by an efficient vendor with a lower level of quality induced from an inefficient vendor is longer than the support duration offered by an inefficient vendor with a higher level of quality induced from an efficient vendor, namely, the direct effect of efficiency on maintenance is larger than the indirect effect of development efficiency on maintenance.

**PROPOSITION 3:** Under asymmetric information and verifiable quality, (i) there is no distortion in quality of software in development phase and maintenance support duration during the support phase for the efficient vendor; (ii) There is a downward distortion in quality of software in development phase and support duration during the maintenance support phase for the inefficient vendor; and (iii) when the development phase vendor is inefficient, then there is a further downward distortion in
support duration for the inefficient vendor compared to the case when the efficient vendor accepts the development phase contract.

The client makes the trade-off between inducing efficient level of quality and support duration from the efficient vendor with information rents and enduring downward distortions on quality and support duration for the inefficient vendor. Furthermore, a downwardly distorted quality in the development phase would cause a further distortion on support duration for the inefficient vendor.

4.2 Bundling contracts

Now we consider the case where the client offers one composite contract for software application development and subsequent maintenance support.

**Lemma 4:** Under asymmetric information and verifiable quality, the optimal software quality and support duration pair is:

\[
\left( q_L^B, t_L^B \right) = \frac{\rho}{2\theta_L - 1} + \frac{(1 - \nu)\rho}{2(\theta_H - \nu\theta_L) - 1 + \nu}, \quad \left( q_H^B, t_H^B \right) = \frac{(1 - \nu)\rho}{2(\theta_H - \nu\theta_L) - 1 + \nu};
\]

and the client’s expected payoff is

\[
\Pi_B = \rho^2 \left[ \frac{\nu}{2\theta_L - 1} + \frac{(1 - \nu)^2}{2(\theta_H - \nu\theta_L) - 1 + \nu} \right].
\]

**Proposition 4:** Under asymmetric information and verifiable quality when bundled contract is offered, (i) the bundling contract serves as a screening device; (ii) there is no quality or support duration distortion for the efficient type vendor compared to the first-best, \( q_L^B = q_L^* \) and \( t_L^B = t_L^* \); (iii) there is a downward distortion in quality and support duration for the inefficient type vendor compared to the first-best, \( q_H^B < q_H^* \), and \( t_H^B < t_H^* \); and (iv) the downward distortion in optimal quality and support duration increases when a random vendor is more likely to be efficient.

The optimal solutions calls for no distortion for the first-best for the efficient type’s in terms of free maintenance support time \( t_L^B = t^*(\theta_L) \) and the efficient vendor would exert \( q_L^B = q^*(\theta_L) \) to get a positive rent. The second best solutions require distortion away from the first-best for the inefficient vendor. The quality-improving effort and the maintenance support period are distorted downwards for the inefficient vendor in order to economize on information rent.

4.3 Comparing unbundled and bundled contracts under asymmetric information

**Proposition 5:** Under asymmetric information and verifiable quality, (i), the expected optimal quality-improving effort exerted by the vendor in the development phase and the expected optimal quality of software is always higher with bundling contract; (ii) the support duration induced from an efficient vendor with bundling contracts is longer than that with unbundling contracts when \( \rho > \frac{4\theta_L^2 - 2\theta_L}{4\theta_L^2 - 2\theta_L + 1} \);(iii) when \( \rho > \frac{4(\theta_H - \nu\theta_L)^2 - 2(1 - \nu)(\theta_H - \nu\theta_L)}{4(\theta_H - \nu\theta_L)^2 - 2(1 - \nu)(\theta_H - \nu\theta_L) + (1 - \nu)^2} \), the support duration induced from an inefficient vendor with bundling contracts is longer than that with unbundling contracts ; and (iv) the client is better off with bundling contract when

\[
\nu \left[ 1 + \frac{\rho}{2\theta_L} \right] + (1 - \nu) \left[ 1 + \frac{(1 - \nu)\rho}{2(\theta_H - \nu\theta_L)} \right] + \frac{(1 - \nu)^2}{2(\theta_H - \nu\theta_L) - 1 + \nu} < \frac{4\nu\rho}{2\theta_L - 1} + \frac{4(1 - \nu)^2\rho}{2(\theta_H - \nu\theta_L) - 1 + \nu}.
\]

The condition in Proposition 5 partitions the two dimensional \( \rho - \nu \) and \( \rho - \theta \) spaces into two regions respectively (see figure 2 below). In the lower-left region of all the figures below, the principal is better off with unbundling contracts while she is better off with bundling contracts in the upper-right region.
The principal is better-off with unbundling contracts

2 2
\( UB_B(C_C_\nu \nu_1) = \Pi = \Pi = \nu = \nu_1 \)

The principal is better-off with bundling contracts

2 1
\( UB_B(C_C_\theta \theta_1) = \Pi = \Pi = \theta = \theta_1 \)

Figure 2. The impact of information asymmetry on client’s preference: a. 

As the probability that the developer is efficient increases, the region in which the principal is better off with bundling contracts becomes larger (Figure 2a), indicating that it becomes easier for her to take advantage of bundling contracts. The intuition is as follows. When the developer is more likely to be efficient, the downward distortions on the investment in quality and the length of support period are relatively small. And the inefficiency resulting from information asymmetry is also small. Thus the efficiency gains from bundling contracts outweigh those from unbundling contracts. This result is counter-intuitive since one might expect that using a bundling contract as a screening device achieves less efficiency (and thus makes the principal less likely be better off) when the agent is more likely to be efficient.

Similarly, when the efficient developer’s efficiency deferential is higher, the bundling region expands. When the efficiency gains from contracting the efficient developer are larger, the inefficiency resulting from bundling contracts (offering a support contract ex ante) can be easily offset and thus bundling contracts are more likely to produce higher payoffs for the client.

The client uses a bundling contract as a screening device to select more efficient vendors and that the optimal length of support period and the optimal investment can be induced from the efficient developer. In order to economize on information rent paid to the efficient developer, an allocative inefficiency in terms of downward distortions on investment and length of support by the inefficient developer has to be endured. Under the optimal contract, the client’s trade-off between rent extraction and efficiency arises in the way that optimal efficiency is preserved with the efficient developer and that zero rent is paid to the inefficient developer.

When a random vendor is more likely to be efficient, larger distortions on optimal length of support period and optimal investment to the inefficient vendor arise. The intuition is that, at the second-best optimum, the client equates the expected marginal efficiency gain and the expected marginal cost of the rent brought about by an infinitesimal increase of the inefficient developer’s investment and length of support. A change of the likelihood of a random vendor being efficient changes both the marginal gain and the marginal cost of rent paid and thus changes the balance between economizing on rent and allowing inefficiency. In the next section we extend the analysis into non-verifiable quality.

5 DISCUSSIONS

With a model of software outsourcing contracts under information asymmetry in which quality of the software affects maintenance cost, we found that the client uses bundling contracts as a screening device to select more efficient vendors. Counter-intuitively, an efficient vendor may be required to offer a shorter period of support for the software application he developed. With verifiable quality, unbundling contracts may enable the client to induce higher quality and longer support period. Further extensions of the current model include investigating the impacts of non-verifiable quality and a setting with multidimensional asymmetric information in which the vendor may exhibit different levels of efficiency for maintenance and development.
Appendix: Proofs of Lemmas and Propositions

**Proof of Lemma 1:** Directly follows from the FOCs. ■

**Proof of Lemma 2:** The client’s payoffs under first best writes
\[ \Pi = \rho x + \rho t - \theta x^2 - \theta t^2 + xt \]
The first order conditions write
\[ \frac{\partial \Pi}{\partial x} = \rho - 2\theta x + t = 0; \quad \frac{\partial \Pi}{\partial t} = \rho - 2\theta t + x = 0. \]
Solving this system of equations, we have
\[ x^* = \frac{\rho}{2\theta - 1}, \quad t^* = \frac{\rho}{2\theta - 1}. \]
Binding the vendor’s participation constraint yields
\[ p^* = \theta x^* - \theta t^* - x^* t^*; \]
Substituting \( t^*, x^* \) into \( \Pi \) yields
\[ \rho^2 \frac{\rho}{2\theta - 1}; \]
Substituting \( t^*, x^* \) and \( p^* \) into \( \Pi \) yields
\[ \rho^2 \frac{\rho}{2\theta - 1}. \]

**Proof of Proposition 1:** Comparing the optimal quality levels and the support periods under bundling and unbundling, we can see that \( x^B > x^U \) is always true and that \( t^B > t^U \) when \( \rho > \frac{4\theta^2 - 2\theta}{4\theta^2 - 2\theta + 1} \).
Comparing the client’s payoffs under bundling and unbundling, we can see that \( \Pi^B > \Pi^U \) when
\[ (2\theta - 1)\rho^2 - 4\theta(2\theta^2 - \theta + 1)\rho + 4\theta^2(2\theta - 1) < 0. \]

**Proof of Lemma 3, Corollary 2 and Proposition 2:** Under information asymmetry, for a menu to be accepted, it must yield to each type at least its outside opportunity level (assuming zero), the following two individual participation constraints must thus be satisfied:
\[ p_H - \theta_H x_H^2 \geq 0; \quad p_L - \theta_L x_L^2 \geq 0 \]
For the menu of contracts to be used as a screening tool, the following incentive compatibility constraints must be satisfied as well:
\[ p_H - \theta_H x_H^2 \geq p_L - \theta_L x_L^2; \quad p_L - \theta_L x_L^2 \geq p_H - \theta_H x_H^2. \]
We define \( U_H = p_H - \theta_H x_H^2, \quad U_L = p_L - \theta_L x_L^2 \). Then, the client’s optimization problem writes,
\[ \max_{\{U_L, U_H\}} \left( x_L^2, x_H^2 \right) \]
subject to,
\[ U_L \geq U_H + \Delta \theta x_H^2 \quad \text{(A 1)} \]
\[ U_H \geq U_L - \Delta \theta x_L^2 \quad \text{(A 2)} \]
\[ U_L \geq 0 \quad \text{(A 3)} \]
\[ U_H \geq 0 \quad \text{(A 4)} \]
We have \( U_L = \Delta \theta x_H^2 \) and \( U_H = 0 \), and the first order conditions write,
\[ \frac{\nu (px_L - \theta_L x_L^2)}{\nu (px_H - \theta_H x_H^2)} = \frac{(1 - \nu) \rho}{2(\theta_H - \nu \theta_L)}. \]
The second best solution:
\[ x_L^{SB} = x_L^* = \frac{\rho}{2\theta_L}, \quad x_H^{SB} = \frac{(1 - \nu) \rho}{4\theta_L - \nu \theta_L}. \]
The client’s expected payoff from development writes
\[ \frac{\nu \rho^2}{4\theta_L} \quad \text{at the support contracting stage, the client will offer a pair of contracts} \]
\[ \{(p_L^{SB}, t_L^{SB}, x_L^{SB}), (p_H^{SB}, t_H^{SB}, x_H^{SB})\} \]
if a type L developer accepts the development contract and offer
\[ \{(p_L, t_L, x_L^{SB}), (p_H, t_H, x_H^{SB})\} \]
if a type H developer accepts the development contract.

**A type L developer accepts the development contract**
Because verifiable quality, the vendor’s support cost function writes,
\[ C_s(\theta) = \theta^2 - \frac{\rho}{2\theta_L} t. \]
Then, the client’s optimization problem writes,
For the menu of contracts to be used as a screening tool, the following incentive compatibility constraints must be satisfied as well:

\[
p_H - \theta_H t_H^i + \frac{\rho}{2\theta_L} t_L \geq 0; \quad p_L - \theta_L t_L^i + \frac{\rho}{2\theta_L} t_L \geq 0
\]

The second best solution: \[ t_H^{SB} = \frac{1}{2\theta_L} + \frac{(1-\nu)\rho}{2(\theta_H - \nu \theta_L)} \], \[ t_L^{SB} = \frac{1}{2\theta_L} + \frac{(1-\nu)\rho}{2(\theta_H - \nu \theta_L)} \] .

The client’s expected payoff in this case writes \[ \frac{1}{4} \left[ \frac{\nu}{\theta_L} + \frac{(1-\nu)^2}{2(\theta_H - \nu \theta_L)} \left( 1 + \frac{\rho}{2\theta_L} \right)^2 \right] . \]

A type H developer accepts the development contract

Because verifiable quality, the vendor’s support cost function writes,

\[
C_S(\theta) = \theta H^2 - \frac{(1-\nu)\rho}{2(\theta_H - \nu \theta_L)} \]

Then, the client’s optimization problem writes,

\[
\max_{\{(L,S)\mid (U_H, U_L)\}} \nu \left( t_L - \theta_L t_L^i + \frac{\rho}{2\theta_L} t_L \right) + (1-\nu) \left( t_H - \theta_H t_H^i + \frac{\rho}{2\theta_L} t_H \right) - (\nu U_L + (1-\nu) U_H)
\]

\[
p_H - \theta_H t_H^i + \frac{\rho}{2\theta_L} t_H \geq 0; \quad p_L - \theta_L t_L^i + \frac{\rho}{2\theta_L} t_L \geq 0
\]

The second best solution: \[ t_H^{SB} = \frac{1}{2\theta_L} + \frac{(1-\nu)\rho}{2(\theta_H - \nu \theta_L)} \], \[ t_L^{SB} = \frac{1}{2\theta_L} + \frac{(1-\nu)\rho}{2(\theta_H - \nu \theta_L)} \] .

The client’s expected payoff in this case writes \[ \frac{1}{4} \left[ \frac{\nu}{\theta_L} + \frac{(1-\nu)^2}{2(\theta_H - \nu \theta_L)} \left( 1 + \frac{\rho}{2\theta_L} \right)^2 \right] . \]

The client’s total expected payoff from development and support writes

\[
\Pi_C^{SB} = \frac{\rho}{4} \left[ \nu \left( 1 + \frac{\rho}{2\theta_L} \right)^2 + (1-\nu) \left( 1 + \frac{(1-\nu)\rho}{2(\theta_H - \nu \theta_L)} \right)^2 + \rho \left( \frac{\nu}{\theta_L} + \frac{(1-\nu)^2}{2(\theta_H - \nu \theta_L)} \right) \right].
\]

Proof of Lemma 4:

Under information asymmetry, for a menu to be accepted, it must yield to each type at least its outside opportunity level (assuming zero), the following two individual participation constraints must thus be satisfied: \[ p_H - \theta_H x_H^i - \theta_H t_H^i + x_H t_H \geq 0; \quad p_L - \theta_L x_L^i - \theta_L t_L^i + x_L t_L \geq 0 \]

For the menu of contracts to be used as a screening tool, the following incentive compatibility constraints must be satisfied as well:

\[
p_H - \theta_H x_H^i - \theta_H t_H^i + x_H t_H \geq 0; \quad p_L - \theta_L x_L^i - \theta_L t_L^i + x_L t_L \geq 0
\]

We define \[ U_H = p_H - \theta_H x_H^i - \theta_H t_H^i + x_H t_H \], \[ U_L = p_L - \theta_L x_L^i - \theta_L t_L^i + x_L t_L \]

Then, the client’s optimization problem writes,

\[
\max_{\{(L,S)\mid (U_H, U_L)\}} \nu \left( \rho x_L + \rho t_L - \theta_L x_L^i - \theta_L t_L^i + x_L t_L \right) + (1-\nu) \left( \rho x_H + \rho t_H - \theta_H x_H^i - \theta_H t_H^i + x_H t_H \right) - (\nu U_L + (1-\nu) U_H) \]

subject to,

\[ U_L \geq U_H + \Delta \theta \left( t_H^i + x_H^i \right) \] (A5)
The ability of the efficient agent to mimic the inefficient agent implies that the efficient agent’s participation constraint (A 7) is always strictly satisfied. If a menu of contracts enables an inefficient agent to reach his status quo utility level, it will be also the case for an efficient agent who can offer the same length of support at a lower cost.

Therefore, we must have,

\[ U_L = \Delta \theta \left( t_H^2 + x_H^2 \right), \quad U_H = 0 \]

The second best solution:

\[ t_{SB}^L = t_L^* = \frac{\rho}{2\theta - 1} \quad \text{and} \quad t_{SB}^H = \frac{(1 - \nu)\rho}{2(\theta - \nu \theta_L) - 1 + \nu} \]

Note that, under the second best case,

\[ x_{SB}^L = x_L^* = \frac{\rho}{2\theta - 1} \quad \text{and} \quad x_{SB}^H = \frac{(1 - \nu)\rho}{2(\theta - \nu \theta_L) - 1 + \nu} \]

\[ p_{SB}^L = \frac{2\Delta \theta (1 - \nu)^2 \rho^2}{(2(\theta - \nu \theta_L) - 1 + \nu)^2} + \frac{\rho^2}{2\theta - 1} \quad \text{and} \quad t_{SB}^H = \frac{(1 - \nu)\rho}{2(\theta - \nu \theta_L) - 1 + \nu} \]

The client’s expected payoff writes \[ \Pi_{SB}^L = \rho^2 \left[ \frac{\nu}{2\theta - 1} + \frac{(1 - \nu)^2}{2(\theta - \nu \theta_L) - 1 + \nu} \right] \]

Comparing Lemma 4 with Lemma 2, Lemma 4 with Lemma 3, we can see the results in Proposition 4 and 5.

References


