Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.031103

ARTICLE

Strategic Contracting for Software Upgrade Outsourcing in Industry 4.0

Cheng Wang[1,2,*] **and Zhuowei Zheng**[1]

[1]College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou, 310014, China

[2]Taizhou Research Institute, Zhejiang University of Technology, Taizhou, 318001, China

*Corresponding Author: Cheng Wang. Email: cwang@zjut.edu.cn

Received: 14 May 2023 Accepted: 06 July 2023 Published: 17 November 2023

ABSTRACT

The advent of Industry 4.0 has compelled businesses to adopt digital approaches that combine software to enhance production efficiency. In this rapidly evolving market, software development is an ongoing process that must be tailored to meet the dynamic needs of enterprises. However, internal research and development can be prohibitively expensive, driving many enterprises to outsource software development and upgrades to external service providers. This paper presents a software upgrade outsourcing model for enterprises and service providers that accounts for the impact of market fluctuations on software adaptability. To mitigate the risk of adverse selection due to asymmetric information about the service provider's cost and asymmetric information about the enterprise's revenues, we propose pay-per-time and revenue-sharing contracts in two distinct information asymmetry scenarios. These two contracts specify the time and transfer payments for software upgrades. Through a comparative analysis of the optimal solutions under the two contracts and centralized decision-making with full-information, we examine the characteristics of the solutions under two information asymmetry scenarios and analyze the incentive effects of the two contracts on the various stakeholders. Overall, our study offers valuable insights for firms seeking to optimize their outsourcing strategies and maximize their returns on investment in software upgrades.

KEYWORDS

Software upgrade outsourcing; the principal-agent; information asymmetry; reverse selection; contract design

1 Introduction

To lead the new industrial revolution, countries are reviving their manufacturing industry [1,2]. Germany, an old manufacturing powerhouse, stands out by proposing Industry 4.0. Industry 4.0 aims to transform machine-led manufacturing into digital manufacturing by utilizing advanced information technology [3].

In the battle for the digitalization of manufacturing, software as an intermediary link between virtual and real interaction, has injected new energy into R&D, simulation verification, manufacturing, management, sales, and service in the manufacturing industry [4]. With government support, some large manufacturing enterprises have taken the lead in becoming trial sites and integrating software into their manufacturing processes. Siemens, a world-class industrial giant with a presence in home

appliances, communications, medical, industrial, and other fields, has been a major beneficiary of software applications in its products and projects. Nowadays, more and more enterprise managers are increasingly aware of the benefits of software in improving efficiency. Moreover, the scope of software applications is no longer limited to internal enterprise operations. Instead, there is a growing interest in using software in the supply chain to achieve cooperation with upstream and downstream enterprises and overcome barriers across all links.

Software development is an ongoing process for enterprises. As market fluctuations occur and enterprise scales expand, software must be upgraded to meet the evolving needs of the business. Software upgrading involves updating the software from a lower version to a higher version. Market trends, industry transformations, or institutional replacements often require enterprises to upgrade or adjust their software. For instance, policies that promote "Internet + E-commerce" marketing models prompt enterprises to successively upgrade software to improve operating efficiency by focusing on procurement and distribution. Moreover, the complexity within the enterprise affects the scale of the enterprises, which is a dynamic process. As the number of employees increases, changes in management methods, technological progress, and adjustments to business chains make it difficult for closed software with single functions and outdated technology to respond quickly to enterprise needs. Upgrading the software to develop more functional services can help enterprises carry out various tasks smoothly and overcome inertia tendencies, thereby increasing enterprise profits [5].

Enterprises have two options for upgrading their software: self-upgrading and outsourcing. When enterprises opt to upgrade their software independently, they typically possess some in-house development capacity, and their internal IT department provides the software upgrade service. In this way, enterprise decision-makers can readily comprehend the software upgrade's direction, monitor development progress, and enjoy the convenience of the ensuing maintenance, upgrade, and optimization processes. Alternatively, outsourcing involves hiring professional service providers like IBM, Oracle, and Microsoft, who have emerged as software service providers with their professional technology and development capabilities and can provide one-stop software upgrade services to the enterprise.

The distinguishing factor in the context of Industry 4.0 is the emphasis on integration and digitization. Industry 4.0 implementation involves three key dimensions: vertical integration, horizontal integration, and end-to-end integration [6]. Vertical integration focuses on achieving intelligent interconnection and digitization within the production system, primarily through production networking. Horizontal integration aims to achieve intelligent interconnection and digitization between organizations across the entire value network [7]. Lastly, end-to-end integration seeks to achieve intelligent interconnection and digitization throughout all stages of the lifecycle. In short, vertical integration primarily addresses digital integration within an organization. On the other hand, horizontal and end-to-end integration focus on establishing cross-organizational digital networks involving various entities. The method of enterprise software upgrade plays a crucial role in this context. When an enterprise chooses to independently upgrade its software, the upgrade service is typically provided by the enterprise's IT department, falling under the category of vertical integration. Conversely, when an enterprise outsources the software upgrade to an external service provider, it involves both the enterprise and the service provider, constituting horizontal integration.

The rise of software outsourcing has brought attention to the principal-agent relationship between enterprises and service providers. The principal-agent theory highlights the risk of reverse selection when there is ex-ante information asymmetry and inconsistent interests between parties [8,9]. In the context of software outsourcing, the service provider typically possesses more information about

the software upgrade's capabilities, qualifications, quality, and cost, while the enterprise has greater knowledge of the software's business value, industry characteristics, and business process. Such information asymmetry can lead to conflicts and damaged interests of one party over the other.

An effective mechanism or contract can address the reverse selection problem that arises due to inconsistent interests and asymmetric information of both parties [10]. In the context of Industry 4.0, the traditional contracts used in software outsourcing may need to be adapted to accommodate the specific characteristics and requirements of digital manufacturing. Unlike traditional manufacturing processes, Industry 4.0 involves a higher level of automation, connectivity, and data exchange, requiring software to be adaptable, scalable, and capable of integrating with various digital technologies and devices. Therefore, the contracts for software upgrade outsourcing in the context of Industry 4.0 should consider these unique aspects and ensure the software's compatibility and interoperability within the digital manufacturing ecosystem.

The motivating context for this problem is the design of contracts for customized software upgrade supply chain in the Industry 4.0 era, consisting of an enterprise and a service provider. The service provider offers tailored software upgrade services to the enterprise, which faces a diminishing utility of its current software version due to market fluctuations. Given the existence of asymmetric information in software upgrade outsourcing, we adopt a principal-agent model. Within this framework, the agent possesses superior information, while the principal lacks crucial knowledge and assumes the responsibility of contract design. In order to motivate agents to engage in software upgrades, the principal must devise a contract menu that offers multiple options, tailored to each agent type. During the contract design phase, agents reveal their information type by selecting the contract aligned with their characteristics. In the contract execution phase, the service provider delivers the upgrade service to the enterprise at the appropriate time, and in return, the enterprise compensates the provider with the corresponding fee. By solving the equilibrium solutions for different scenarios, we derive software upgrade time and generalize the characteristics of the solutions. Ultimately, this research provides managerial implications for enhancing the efficiency and effectiveness of software upgrade outsourcing contract.

Our paper makes the following contributions: Firstly, we investigate the optimal software upgrade timing decision in response to market fluctuations when a centralized decision is made with full-information. Secondly, we use contracts to address the information asymmetry between the service provider and the enterprise in two scenarios where the former has cost information and the latter has revenue information. Thirdly, we explore the impact of different performance-based contracts on software upgrade timing decisions and profits under information asymmetry. Finally, we compare the equilibrium solutions of using a pay-per-time contract and a revenue-sharing contract under cost and revenue information asymmetry scenarios, respectively, to determine their incentive effects on the parties involved.

This research holds significant importance for both enterprises and service providers operating in the software outsourcing market. By introducing a novel software upgrade outsourcing model that incorporates the influence of market fluctuations on software adaptability, this study presents a valuable decision framework that differs from traditional research perspectives. The proposed pay-per-time and revenue-sharing contracts address the challenges of adverse selection caused by asymmetric information. Through a comparative analysis, the research highlights the characteristics and incentive effects of these contracts on various stakeholders. The findings not only provide a theoretical basis for future research and further exploration of optimal contract design in software outsourcing but also offer practical insights for firms in making informed outsourcing decisions and maximizing their

returns on investment. The innovative nature of this research lies in its ability to merge the principles of Industry 4.0 with software upgrade outsourcing, effectively empowering enterprises to optimize their software systems and adapt to the demands of the digital era.

The framework of this paper is as follows: The literature review in Section 2 provides an overview of relevant research. Section 3 outlines the software upgrade outsourcing model between the enterprise and the service provider. In Section 4, we solve for the optimal software upgrade timing strategy under complete information centralized decision-making. Section 5 investigates the impact of two performance-based contracts on optimal software upgrade timing decision-making under incomplete information decentralized decision-making. Finally, Section 6 summarizes our work and findings. The proof process for the Propositions in this paper is detailed in Appendixes A and B.

2 Literature Review

2.1 Software Outsourcing

This section will elaborate on the characteristics of software outsourcing from three perspectives: enterprise software development strategy, software outsourcing objects, and asymmetric information.

Software development strategy in an enterprise can be classified into two categories based on the mode of development: internal development and outsourcing [11]. Enterprises that opt for in-house development typically have a certain level of R&D capability, which enables them to maintain strict control over the direction and progress of software development. In contrast, software outsourcing involves subcontracting part or all of an enterprise's software project to an external service provider. Software outsourcing can be further divided into full outsourcing and partial outsourcing based on the degree of outsourcing [12,13]. Most enterprises choose software outsourcing to reduce costs, access advanced resources, and focus on their core business. However, software outsourcing also comes with risks, such as loss of control over information systems, hidden costs, and loss of innovation [14].

Research on the economic aspects of enterprise software development strategies dates back to Richmond et al. [15], who explored the impact of different revenue rules on enterprises' decisions to choose between in-house development or outsourcing, based on a two-stage software development model. Another study by Wang et al. [16] compared the value brought to the company by in-house development and software outsourcing, and determined that in-house R&D would generate more net profit. Early literature analyzed the software development strategies of enterprises, internal development or software outsourcing, mainly from an economic perspective. In contrast, this paper focuses solely on enterprise software outsourcing and aims to achieve the same level of results as in-house R&D through effective contract design.

When enterprises choose to outsource their software, outsourcing can be classified into two types: software development outsourcing and software upgrade outsourcing, based on the specific outsourcing object. Software development outsourcing involves the outsourcing of software development processes to external service providers, who help the enterprise to create software from scratch. Casado-Lumbreras et al. [17] have identified various challenges and opportunities associated with enterprise software development. To address the potential conflicts between the host firm and the outsourcing firm, Bandyopadhyay et al. [18] developed a knowledge-sharing model for software outsourcing teams.

A software upgrade is a newer version of the software that has been developed. Enterprise software upgrades are usually pursued for various reasons, such as reducing operational costs, pursuing new business opportunities, or breaking free from inertia tendencies. When an enterprise decides

to outsource software upgrades, the service provider is often the same one that provides software development services. Despite the widespread practice of outsourcing software upgrades, there is relatively little literature on this topic. Pricing is one of the major concerns of enterprises. To price their products, software providers need to consider the attractiveness of both current and future versions. Kornish [19] developed a two-stage game model that demonstrates the existence of an equilibrium pricing strategy if special pricing is applied to consumers who purchase earlier versions. Meanwhile, Bala et al. [20] studied the optimal pricing of new versions by examining the relationship between the magnitude of software product improvements and the structure of equilibrium pricing. Instead of focusing on pricing strategies for new software versions, we explore the adaptability of software and the optimal software upgrade timing strategy.

When enterprises outsource software development, they often face information asymmetry with the external service provider. Existing research categorizes information asymmetry into three types: (1) ex-ante and ex-post information asymmetry based on when it occurs, (2) information type and behavioral information asymmetry based on the content of information, and (3) information asymmetry of the service provider and the outsourcer based on the subject of the information. These three categories of information asymmetry are common in studies related to outsourcing. In the software outsourcing literature, studies typically focus on software value, cost, quality, capability, effort, and other types of information [21–23].

2.2 Adverse Selection Problem and Its Solution

The software upgrade outsourcing relationship between the service provider and the enterprise can be seen as a principal-agent relationship. However, this relationship is characterized by inconsistent interests and information asymmetry, leading to adverse selection [24,25]. Adverse selection occurs when one party in the market uses more information than the other party (such as software value, cost, and agent ability information) to benefit themselves and harm the other party's interests before the contract is signed [26]. In different fields, research has provided insights to address the adverse selection problem for ex-ante information asymmetry in various contexts.

In their study of carbon capture and storage systems, Cai et al. [27] developed a principal-agent model that addresses the adverse selection problem caused by asymmetric ex-ante information between contracting parties. An interesting aspect of their work is the consideration of a heterogeneous distribution of agent demand with varying levels under each distribution. Many studies focus on the asymmetric information in only one scenario, our paper investigates the adverse selection problem in two scenarios: the service provider's cost information asymmetry and the enterprise's revenue information asymmetry.

An effective contract design can solve the adverse selection problem between the contracting parties. Considering that outsourcing investments are irreversible and costs are uncertain, Yao et al. [28] analyzed the issues of when to outsource, how to choose a contract, and how to negotiate an outsourcing contract by building a model.

In the software outsourcing domain, contracts are often incomplete due to unpredictable events in an incomplete contract environment [29]. Time-and-materials (T&M) contracts and fixed-price (FP) contracts are two common types used [30]. T&M contracts have some risk protection and consist of two components: the project development cost and the service provider's profit. In contrast, fixed-price (FP) contracts have a price cap and are not adjusted once set within the agreed risk range. FP contracts are less efficient for software projects with high uncertainty in software development costs, and are suitable for software projects with low development difficulty and short development time.

T&M contracts can be used for relatively difficult software projects, but need to be coupled with project supervision and require enterprises to have relevant software development knowledge. According to Gopal et al. [31], FP contracts make providers more efficient in the software development process and develop higher-quality software.

As software development becomes more complex, contract designers are increasingly considering performance-based contracts (PB) to align the interests of both parties. PB contracts tie the service provider's revenue to the quality and performance of the software. Revenue-sharing (RS) contracts are the most common type of PB contract, where the revenue generated by the software is shared between the service provider and the enterprise. Contract designers need to select and design contracts based on the specific requirements of software outsourcing projects and incentive goals to achieve the optimal outcome. To address the challenge of monitoring service provider efforts during software development, Huang et al. [32] explored the effect of monitoring on the choice between a time-and-material contract and a revenue-sharing contract. In this paper, we investigate the impact of two performance-based contracts, pay-per-time and revenue-sharing contracts on the optimal software upgrade time.

3 The Model

This section presents the software upgrade outsourcing model between an enterprise and a service provider. The enterprise relies on customized software to generate revenue, but over time, the adaptability of the software decreases due to market fluctuations. The service provider offers software upgrades to the enterprise, aiming to enhance its software's adaptability. Both parties must make strategic decisions regarding the optimal timing of software upgrades. This section outlines the key elements of the software upgrade outsourcing model.

Software Adaptability Deterioration: The software's adaptability gradually diminishes as it operates in a changing market environment. Initially, when the software is fully adapted to the current environment, it generates significant revenue per unit time denoted as R, which varies based on the scale of the enterprise. The larger the enterprise, the higher the revenue per unit time that can be generated for the enterprise when the software is fully adapted to the current environment.

However, due to market fluctuations represented by σ, a gap emerges between the existing software version and the version that is adaptable to the current environment. This gap is quantified as the degree of software deviation $D(t)$, which increases with the length of time t the software is in use and is influenced by market fluctuations σ, as shown in Fig. 1. For convenience, we normalize $D(t)$ to a range between 0 and 1. The parameter σ determines the speed at which the software deviates from its fully adaptable state. We assume that the degree of software deviation follows the functional relationship [33]:

$$D(t) = 1 - e^{-\sigma t} \tag{1}$$

where $D(t)$ ranges from 0 to 1. In the context of our study, t has two interpretations: (1) the time of software upgrade and (2) the running time of existing software version.

Enterprise Perspective: The existence of a gap between the existing software version and the version adapted to the current environment hinders enterprises from fully realizing their value, which can result in substantial costs. For instance, in the "Internet + E-commerce" marketing mode, many enterprises have upgraded their software to remain competitive. Failure to take action may lead to the erosion of the enterprise's original market share by competitors that have adopted "Internet + E-commerce"

models, resulting in financial losses. Therefore, the actual revenue P of the enterprise comprises two components: revenue received when the software is fully adapted to the current environment and lost due to the gap between the existing software version and the software version adapted to the current environment. Mathematically, this can be expressed as:

$$P(t) = R\int_0^t [1 - D(x)]\,\mathrm{d}x \tag{2}$$

The service provider offers software upgrades to the enterprise, and in return, the enterprise pays the service provider a fee, denoted by T. We assume that the software upgrade provided by the service provider will be successful.

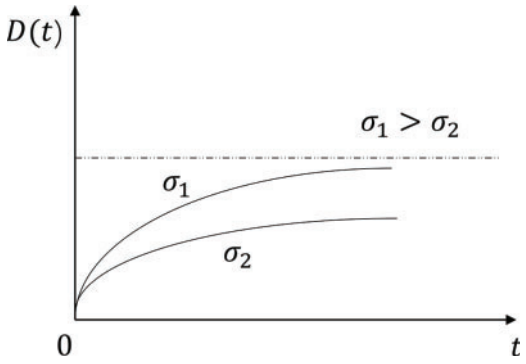

Figure 1: The relationship between the degree of software deviation and upgrade practices and market volatility

The Service Provider Perspective: The service provider incurs costs associated with software upgrades, comprising fixed costs and variable costs. The fixed costs K represent the resources invested by the service provider, including human resources, for each software upgrade. The variable costs $C(D)$ depend on the degree of software deviation at the time of upgrade. As the deviation increases, the service provider incurs higher variable costs, exhibiting a marginal increasing trend. Previous literature has provided insights into the variable cost functions, and we adopt the following form [27]:

$$C(D) = \frac{1}{2}kD^2 \tag{3}$$

where k denotes the variable cost coefficient for the service provider to upgrade the software.

Performance-Based Contracts: In the realm of service contracts, performance-based (PB) contracts are commonly employed to align the agent's interests with those of the principal. These contracts establish a direct link between the agent's profits and their performance, thereby incentivizing actions that benefit the principal. Our study focuses on two specific types of PB contracts: pay-per-time contracts and revenue-sharing contracts.

The pay-per-time contract encompasses two key elements: the timing of software upgrades and transfer payments. This contract structure outlines when the software upgrades should take place and determines the corresponding transfer payments to the service provider. On the other hand, the revenue-sharing contract entails a collaborative approach between the enterprise and the service provider, where they jointly partake in the revenue generated by the software. Similar to the pay-per-time contract, this arrangement incorporates considerations regarding the optimal timing of software

upgrades. Additionally, it outlines the proportion of revenue that the service provider will receive as part of the transfer.

The existence of unobservable information (the service provider's cost information/the enterprise's revenue information) gives rise to the challenge of adverse selection in software upgrade outsourcing contracts. This information asymmetry poses a significant hurdle prior to contract signing. However, despite the inability to directly observe the asymmetric information between the service provider and the enterprise, the party at the information disadvantage in our model can strategically design the contract to gain insights into the information advantage held by the other party.

Assumptions: To simplify the calculation process, our model does not consider the time required for the service provider to develop the new version, the time needed for the enterprise to install the new version, and the impact of market fluctuations on the existing software version during the service provider's research and development phase.

For a comprehensive understanding of the model, we provide a list of parameters involved and their descriptions in Table 1.

Table 1: Model parameters and their descriptions

Symbols	Descriptions
$D(t)$	The degree of software deviation
t	The time of software upgrade
σ	Market fluctuations
$P(t)$	The actuall revenue of the enterprise
R	Revenue generated for the enterprise when the software is fully adapted to the current environment for a defined enterprise scale
T	Fees paid by the enterprise to the service provider for upgrading the software
K	Fixed costs for the service provider to upgrade software
$C(D)$	Variable costs for the service provider to upgrade software
k	Variable cost coefficient for software upgrades by the service provider

Next, we explore the properties of optimal software upgrade time for outsourcing in two different settings: one with full-information and the other with incomplete information.

4 Benchmark: Centralized Decision with Full-Information

To establish a benchmark, we first analyze the optimal software upgrade timing decision in a centralized decision-making case with full-information. In this scenario, the service provider is treated as a department of the enterprise, and the payment from the enterprise to the service provider is purely monetary. Hence, the problem is to determine the optimal software upgrade time t to maximize the system's expected profits. The optimal expected profits of the system are expressed below:

$$\max_{t} W(t) = P(t) - C[D(t)] - K = R\int_0^t [1 - D(x)]\mathrm{d}x - \frac{1}{2}kD^2(t) - K \tag{4}$$

subject to

$$t \geq 0 \tag{5}$$

Proposition 4.1 characterizes the optimal software decision for centralized decision-making with full-information, which is proved in Appendix A.

Proposition 4.1. For centralized decision with full-information, when $R \geq k\sigma$, t^* takes infinity, indicating that the software does not need to be upgraded during its use. When $0 \leq R < k\sigma$, the optimal software upgrade time and the optimal expected profits of the system are

$$t^* = \frac{\ln\left(1 - \dfrac{R}{k\sigma}\right)}{-\sigma},\ W(t^*) = \frac{R^2}{2k\sigma^2} - K \tag{6}$$

Moreover, t^* and $W(t^*)$ are increasing in R and decreasing in σ.

The optimal software upgrade timing decision for centralized decision with full-information can be divided into two cases. In the first case, where $R \geq k\sigma$, t^* is infinite, indicating that the enterprise need not consider software upgrades after obtaining the existing software version. Moreover, the expected profits of the system will continue to increase as long as the software is in use. This is because any losses resulting from software deviations caused by market fluctuations are insignificant when compared to the value that the existing software provides to the enterprise. As a result, the enterprise is not compelled to upgrade the software.

In the second case, where $0 \leq R < k\sigma$, the optimal software upgrade time t^* exists. The expected profits of the system increase initially, then decrease over time, and reach their maximum value at t^*. This indicates that prior to t^*, the utility derived from the existing software version is positive. However, after t^*, enterprises are forced to enhance the value of their software through upgrades due to the reduced overall profits.

Furthermore, for centralized decision-making with full-information, the optimal software upgrade time is delayed as the enterprise scale expands and advanced as market fluctuations increase. Moreover, the expected profits of the system increase with the expansion of the enterprise scale, but decrease with the increase of market fluctuations.

5 Contract Design in the Case of Decentralized Decision with Incomplete Information

In a software upgrade outsourcing relationship, the service provider and enterprise often keep their cost and revenue information private. This lack of transparency can result in adverse selection if the enterprise does not have accurate access to the service provider's cost information. The service provider may inflate the cost of the software upgrade to secure a higher profit margin before signing the contract. Conversely, if the service provider does not know the true revenue information of the enterprise, the enterprise may under-report its software revenue to gain excess revenue, leading to further adverse selection problems. To address these issues, this section proposes the use of incentive contracts that screen the true information of service providers and enterprises in situations where they have private information. The goal is to achieve the optimal solution in a centralized decision-making environment with full-information.

In this section, we establish the software upgrade outsourcing relationship between the enterprise and the service provider as a principal-agent model, based on the widely-used principal-agent theory. In principal-agent theory, the party possessing private information is referred to as the agent, while the party lacking information is known as the principal.

5.1 Contract Design for the Service Provider with Private Cost Information

As mentioned earlier, the service provider's costs consist of fixed and variable costs. Fixed costs refer to the human resource costs invested by the service provider, which can be estimated by the enterprise based on the average salary of software developers in the market. Hence, we consider fixed costs as common knowledge between the parties to the contract. On the other hand, variable costs are influenced by the software deviation degree D and the cost coefficient k, which is determined by the service provider's business capability. A stronger business capability results in a smaller cost coefficient, while a weaker business capability leads to a greater cost coefficient. Therefore, information asymmetry between the parties mainly exists in the cost coefficient k.

Assuming that service providers in the market can be categorized into high-cost and low-cost types based on their cost coefficient, denoted by k_h and k_l, respectively, where $k_h > k_l$. The enterprise has no prior knowledge of the cost coefficient type of the service provider but holds prior beliefs about its distribution. Let π_{1h} be the probability that the service provider is of the high-cost type, π_{1l} be the probability that it is a low-cost type, and $\pi_{1h} + \pi_{1l} = 1$.

5.1.1 Pay-Per-Time Contract Offered by the Enterprise

We consider a scenario where the service provider possesses private cost information. In this case, the enterprise offers a pay-per-time contract (t, T), where the service provider upgrades the software for the enterprise at time t, and in return, the enterprise compensates the service provider with payment T for the software upgrade. The time line between two parties can be divided into two stages: contract design and contract execution, as illustrated in Fig. 2.

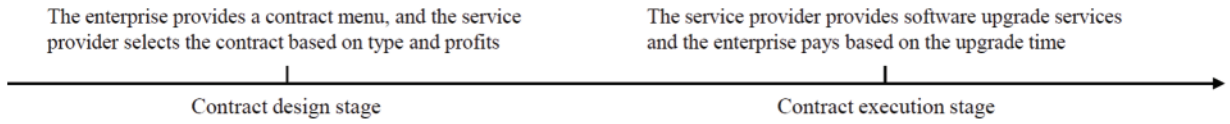

Figure 2: The time line using pay-per-time contracts where the service provider has privare cost information

In the contract design stage, the service provider knows its cost type, but the enterprise only has prior beliefs about the likelihood of the service provider's variable cost coefficient type. The service provider is presented with a contract menu, $\left\{(t_l^{pe}, T_l^{pe}), (t_h^{pe}, T_h^{pe})\right\}$, from which to choose. The superscript pe indicates that the pay-per-time contract is provided by the enterprise. Under this contract, the low-cost type service provider provides software upgrade services at moment t_l^{pe} and receives the corresponding remuneration T_l^{pe}, while the high-cost type service provider provides software upgrade services at moment t_h^{pe} and receives the corresponding payment T_h^{pe}. The service provider can choose to accept or refuse the contract menu provided by the enterprise. If accepted, the service provider will select the contract that best suits it based on its profits and cost type.

In the contract execution stage, the enterprise identifies the cost type of the service provider based on the chosen contract. Once the contract is signed, the service provider delivers the software upgrade

service at the agreed-upon time, and the enterprise makes the corresponding payment based on the upgrade time.

In outsourcing agreements, the enterprise's contract aims to maximize its profits while also satisfying the service provider's individual rational (*IR*) and incentive compatibility (*IC*) constraints. To achieve this, an optimal contract can be formulated as follows:

$$\max_{\{t_n^{pe},T_n^{pe}\}_{n=l,h}} \quad U^{pe} = \pi_{1l}[P(t_l^{pe}) - T_l^{pe}] + \pi_{1h}[P(t_h^{pe}) - T_h^{pe}]$$

subject to

$$\begin{aligned}
&T_l^{pe} - \frac{1}{2}k_l[D(t_l^{pe})]^2 - K \geq 0 && (IR_l^{pe})\\
&T_h^{pe} - \frac{1}{2}k_h[D(t_h^{pe})]^2 - K \geq 0 && (IR_h^{pe})\\
&T_l^{pe} - \frac{1}{2}k_l[D(t_l^{pe})]^2 - K \geq T_h^{pe} - \frac{1}{2}k_l[D(t_h^{pe})]^2 - K && (IC_{lh}^{pe})\\
&T_h^{pe} - \frac{1}{2}k_h[D(t_h^{pe})]^2 - K \geq T_l^{pe} - \frac{1}{2}k_h[D(t_l^{pe})]^2 - K && (IC_{hl}^{pe})\\
&t_n^{pe},\ T_n^{pe} \geq 0 \quad \forall n \in \{l,h\} && (NN_n^{pe})
\end{aligned} \tag{7}$$

The individual rationality (*IR*) constraints, denoted by IR_l^{pe} and IR_h^{pe} set a minimum level of acceptance for the service provider. Additionally, incentive compatibility (*IC*) constraints, denoted by IC_{lh}^{pe} and IC_{hl}^{pe}, encourage the service provider to disclose its true costs by allowing it to select a contract that aligns with its actual cost type, resulting in profits equal to or greater than it would receive with a contract that aligns with a different cost type.

Proposition 5.1 describes the optimal menu of a pay-per-time contract provided by the enterprise in the context of asymmetric cost information, as proved in Appendix B.

Proposition 5.1. Under the condition $0 \leq R < k_l\sigma$, using pay-per-time contract in the case where the service provider has private cost information, the optimal contract menu designed by the enterprise is $\left\{(t_l^{pe}, T_l^{pe}), (t_h^{pe}, T_h^{pe})\right\}$, where

$$t_l^{pe} = t_l^* = \frac{\ln\left(1 - \dfrac{R}{k_l\sigma}\right)}{-\sigma} \tag{8}$$

$$T_l^{pe} = \frac{R^2(k_h - k_l)(1-\pi_{1l})^2}{2\sigma^2(k_h - \pi_{1l}k_l)^2} + \frac{R^2}{2k_l\sigma^2} + K \tag{9}$$

$$t_h^{pe} = \frac{\ln\left[1 - \dfrac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l}k_l)}\right]}{-\sigma} < t_h^* \tag{10}$$

$$T_h^{pe} = \frac{k_h R^2(1-\pi_{1l})^2}{2\sigma^2(k_h - \pi_{1l}k_l)^2} + K \tag{11}$$

In Proposition 5.1, t_l^* denotes the time to upgrade the software of the low-cost type service provider in the case of a centralized decision with full-information, t_h^* denotes the time to upgrade the software of the high-cost type service provider in the case of a centralized decision with full-information.

In the context of pay-per-time contracts with asymmetric cost information, the optimal software upgrade time for low-cost service providers remains the same as in the case of the centralized decision with full-information, while the optimal software upgrade time for high-cost service providers is distorted downward. Specifically, $t_l^{pe} = t_l^*$ and $t_h^{pe} < t_h^*$. This indicates that the optimal solutions obtained through pay-per-time contracts have the characteristics of "no distortion at the high end (the low-cost type service provider) and downward distortion at the low end (the high-cost type service provider)" when the service provider has private cost information.

This is because, in a centralized decision with full-information, the enterprise can offer contracts that correspond to the cost type of the service provider, and maximize profits by ensuring that both types of service providers receive zero utility. However, when the service provider has private cost information, the enterprise needs to design the contract in a way that allows the high-cost type service provider to receive zero utility, while preventing the low-cost type provider from masquerading as the high-cost type to gain positive utility. In other words, to achieve a separation equilibrium, the enterprise may choose to transfer some information rent to the low-cost type service provider with a cost advantage, so that their profits are equal to those obtained by selecting the high-cost type service provider. In essence, enterprises must carefully craft contracts to effectively differentiate between the two cost types of service providers and maximize their profits accordingly.

5.1.2 Revenue-Sharing Contract Offered by the Enterprise

Next, we consider that when the service provider has private cost information, the enterprise offers a revenue-sharing contract that includes a term (t, α). In this contract, the service provider provides software upgrade services to the enterprise at time t, and both parties agree that after the upgrade, the enterprise will transfer a certain share α of the revenue to the service provider as a reward for services. The timing order between the parties can be divided into two stages: contract design and contract execution, as shown in Fig. 3.

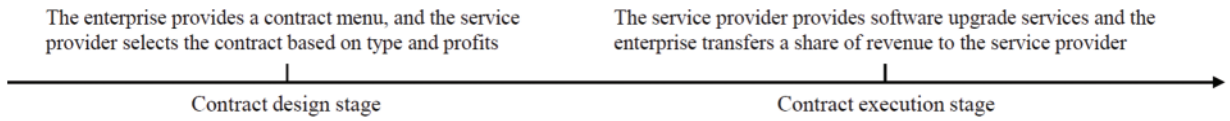

Figure 3: The time line using revenue-sharing contracts where the service provider has privare cost information

The enterprise design a contract menu $\left\{(t_l^{re}, \alpha_l^{re}), (t_h^{re}, \alpha_h^{re})\right\}$ when the service provider has private cost information. The *re* notation in the contract menu indicates that the revenue-sharing contract is provided by the enterprise. The contract menu is tailored to the cost type of the service provider, where $(t_l^{re}, \alpha_l^{re})$ represents the moment of software upgrade and revenue share transferred for the low-cost type service provider, and $(t_h^{re}, \alpha_h^{re})$ represents the same for the high-cost type service provider.

Therefore, under the asymmetric cost information of the service provider, the problem of the enterprise is to design the contract menu $\left\{(t_l^{re}, \alpha_l^{re}), (t_h^{re}, \alpha_h^{re})\right\}$ so that

$$\max_{\{t_n^{re}, \alpha_n^{re}\}_{n=l,h}} \quad U^{re} = \pi_{1l} P(t_l^{re})(1-\alpha_l^{re}) + \pi_{1h} P(t_h^{re})(1-\alpha_h^{re})$$

subject to

$$\alpha_l^{re} P(t_l^{re}) - \frac{1}{2} k_l [D(t_l^{re})]^2 - K \geq 0 \qquad (IR_l^{re})$$

$$\alpha_h^{re} P(t_h^{re}) - \frac{1}{2} k_h [D(t_h^{re})]^2 - K \geq 0 \qquad (IR_h^{re}) \tag{12}$$

$$\alpha_l^{re} P(t_l^{re}) - \frac{1}{2} k_l [D(t_l^{re})]^2 - K \geq \alpha_h^{re} P(t_h^{re}) - \frac{1}{2} k_l [D(t_h^{re})]^2 - K \qquad (IC_{lh}^{re})$$

$$\alpha_h^{re} P(t_h^{re}) - \frac{1}{2} k_h [D(t_h^{re})]^2 - K \geq \alpha_l^{re} P(t_l^{re}) - \frac{1}{2} k_h [D(t_l^{re})]^2 - K \qquad (IC_{hl}^{re})$$

$$t_n^{re},\ \alpha_n^{re} \geq 0 \quad \forall n \in \{l, h\} \qquad (NN_n^{re})$$

Proposition 5.2 characterizes the optimal contract menu and its proof is given in Appendix B.

Proposition 5.2. Under the condition that $0 \leq R < k_l \sigma$, using revenue-sharing contract in the case where the service provider has private cost information, the optimal contract menu designed by the enterprise is $\left\{(t_l^{re}, \alpha_l^{re}), (t_h^{re}, \alpha_h^{re})\right\}$, where

$$t_l^{re} = t_l^* = \frac{\ln\left(1 - \dfrac{R}{k_l \sigma}\right)}{-\sigma} \tag{13}$$

$$\alpha_l^{re} = \frac{(k_h - k_l)(1-\pi_{1l})^2}{2(k_h - \pi_{1l} k_l)^2} + \frac{K k_l \sigma^2}{R^2} + \frac{1}{2} \tag{14}$$

$$t_h^{re} = \frac{\ln\left[1 - \dfrac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l} k_l)}\right]}{-\sigma} < t_h^* \tag{15}$$

$$\alpha_h^{re} = \frac{k_h(1-\pi_{1l})}{2(k_h - \pi_{1l} k_l)} + \frac{K\sigma^2}{R^2(1-\pi_{1l})} \tag{16}$$

By comparing the optimal solutions achieved using the pay-per-time contract, revenue-sharing contract, and centralized decision-making with full-information, we can obtain

$$t_l^{re} = t_l^{pe} = t_l^* = \frac{\ln\left(1 - \dfrac{R}{k_l \sigma}\right)}{-\sigma} \tag{17}$$

$$t_h^{re} = t_h^{pe} = \frac{\ln\left[1 - \dfrac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l} k_l)}\right]}{-\sigma} < t_h^* \tag{18}$$

$$\alpha_l^{re} P(t_l^{re}) = T_l^{pe} = \frac{1}{2} k_h [D(t_h^{re})]^2 + \frac{1}{2} k_l [D(t_l^{re})]^2 - \frac{1}{2} k_l [D(t_h^{re})]^2 + K \tag{19}$$

$$\alpha_h^{re} P(t_h^{re}) = T_h^{pe} = \frac{1}{2} k_h [D(t_h^{re})]^2 + K \tag{20}$$

where $\alpha_l^{re} P(t_l^{re})$ denotes the fee paid by the enterprise to the low-cost type service provider, $\alpha_h^{re} P(t_h^{re})$ denotes the fee paid by the enterprise to the high-cost type service provider.

Eqs. (17)–(18) demonstrate that the optimal software upgrade time for both high and low-cost service providers is the same under both the revenue-sharing and pay-per-time contracts. Likewise, Eqs. (19)–(20) show that the payouts paid to high and low-cost service providers under the revenue-sharing contract are equivalent to the payouts paid under the pay-per-time contract. Therefore, our findings suggest that these two contracts are equivalent in cases where there is asymmetric information about the service providers' costs.

Fig. 4 compares the profits of the enterprise under centralized decision-making with full-information, pay-per-time contract, and revenue-sharing contract when the service provider has private cost information. The profit curves for pay-per-time and revenue-sharing contracts overlap exactly, indicating their equivalence in the presence of asymmetric information about the service provider's costs. However, due to the need to transfer some information rent to low-cost type service providers, the enterprise's profits under these contracts are always lower than those under centralized decision-making with full-information, as shown in Fig. 4.

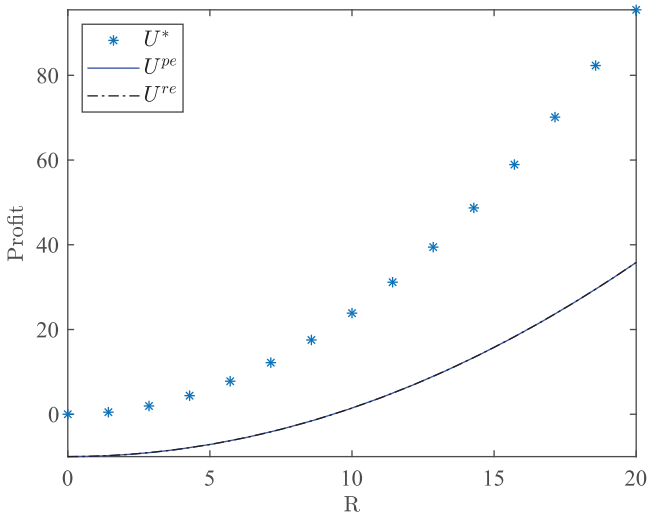

Figure 4: Profits comparison for the enterprise, $\sigma = 0.2, k_l = 100, k_h = 110, K = 10, \pi_{1l} = 0.5$

5.2 Contract Design for the Enterprise with Private Revenue Information

As previously noted, the revenue generated by the software fully adapting to the current environment is denoted as R, and it is proportional to the scale of the enterprise. However, enterprises of the same scale may have different revenues due to their unique characteristics and industry conditions. We assume that enterprises of the same scale in the market can be classified as either high-revenue or low-revenue types. Let R_h represent the revenue of the high-revenue type enterprise when the software

is fully adapted to the current environment, and R_l represents the revenue of the low-revenue type enterprise under the same condition, where $R_h > R_l$. The average revenue of both types of enterprises is denoted as R. The service provider does not know in advance which type of the enterprise but has a prior belief about its distribution. Assuming that the probability that the enterprise is a high-revenue type is denoted as π_{2h}, the probability that it is a low-revenue type is π_{2l}, and $\pi_{2h} + \pi_{2l} = 1$.

5.2.1 Pay-Per-Time Contract Offered by the Service Provider

We now explore the use of a pay-per-time contract when the enterprise possesses private revenue information. In contrast to the scenario where the service provider has private cost information, when the enterprise has private revenue information, the service provider designs the contract. This contract has a term (t, T), where the service provider performs software upgrades for the enterprise at time t, and the enterprise pays T to the service provider in return. The time line between two parties is altered and divided into two stages, as depicted in Fig. 5.

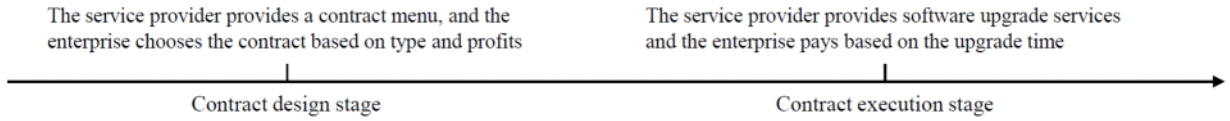

Figure 5: The time line using pay-per-time contracts where the enterprise has privare revenue information

In the contract design stage, before signing the contract, the enterprise is aware of its revenue type, and the revenue generated by the software remains confidential to the enterprise. The service provider designs a contract menu $\left\{(t_h^{ps}, T_h^{ps}), (t_l^{ps}, T_l^{ps})\right\}$ for the enterprise. Here, the superscript ps denotes that the pay-per-time contract is offered by the service provider. (t_h^{ps}, T_h^{ps}) specifies the moment of software upgrade and the corresponding software upgrade fee for the high-revenue type enterprise, while (t_l^{ps}, T_l^{ps}) denotes the moment of software upgrade and the corresponding software upgrade fee for the low-revenue type enterprise. The enterprise selects a contract based on its expected profits and revenue type.

In the contract execution stage, the service provider determines the revenue information type based on the contract selected by the enterprise. Once the contract is signed, the service provider offers software upgrade services at the agreed-upon time. The enterprise then pays for the upgrade based on the upgrade time.

In this case, we can express the optimal contract menu between the parties as follows:

$$\max_{\{t_n^{ps},T_n^{ps}\}_{n=l,h}} \quad V^{ps} = \pi_{1l}[T_l^{ps} - C[D(T_l^{ps})] - K] + \pi_{1h}[T_h^{ps} - C[D(T_h^{ps})] - K]$$

subject to

$$R_l \int_0^{t_l^{ps}} [1 - D(x)]\mathrm{d}x - T_l^{ps} \geq 0 \qquad (IR_l^{ps})$$

$$R_h \int_0^{t_h^{ps}} [1 - D(x)]\mathrm{d}x - T_h^{ps} \geq 0 \qquad (IR_h^{ps}) \qquad (21)$$

$$R_l \int_0^{t_l^{ps}} [1 - D(x)]\mathrm{d}x - T_l^{ps} \geq R_l \int_0^{t_h^{ps}} [1 - D(x)]\mathrm{d}x - T_h^{ps} \qquad (IC_{lh}^{ps})$$

$$R_h \int_0^{t_h^{ps}} [1 - D(x)]\mathrm{d}x - T_h^{ps} \geq R_h \int_0^{t_l^{ps}} [1 - D(x)]\mathrm{d}x - T_l^{ps} \qquad (IC_{hl}^{ps})$$

$$t_n^{ps},\ T_n^{ps} \geq 0 \quad \forall n \in \{l, h\} \qquad (NN_n^{ps})$$

Proposition 5.3 describes the optimal menu of the pay-per-time contract offered by the service provider when the enterprise has private revenue information, which is proved in Appendix B.

Proposition 5.3. Under the conditions that $0 < R_h < k\sigma$, $\pi_{2h}R_h < R_l < R_h$, using pay-per-time contract in the case where the enterprise has private revenue information, the optimal contract designed by the service provider is $\left\{(t_l^{ps}, T_l^{ps}), (t_h^{ps}, T_h^{ps})\right\}$, where

$$t_l^{ps} = \frac{\ln\left[1 - \dfrac{R_l - R_h\pi_{2h}}{(1-\pi_{2h})k\sigma}\right]}{-\sigma} < t_l^* \qquad (22)$$

$$T_l^{ps} = \frac{R^2 - R_l R_h \pi_{2h}}{(1-\pi_{2h})k\sigma^2} \qquad (23)$$

$$t_h^{ps} = t_h^* = \frac{\ln\left(1 - \dfrac{R_h}{k\sigma}\right)}{-\sigma} \qquad (24)$$

$$T_h^{ps} = \frac{R^2}{k\sigma^2} - \frac{(R_l - R_h)(R_l - R_h\pi_{2h})}{(1-\pi_{2h})k\sigma^2} \qquad (25)$$

Compared with the optimal contract in the case of centralized decision with full-information, the optimal software upgrade time of the high-revenue type enterprise remains unchanged, while the optimal software upgrade time of the low-revenue type enterprise is distorted downward, $t_h^* = t_h^{ps}, t_l^* > t_l^{ps}$, when offering the pay-per-time contract with asymmetric information about the enterprise's revenue. This finding suggests that the optimal software upgrade time is characterized by "no distortion at the high end (the high-revenue type enterprise) and downward distortion at the low end (the low-revenue type enterprise)" when the enterprise has private revenue information and the pay-per-time contract is used.

The above phenomenon can be attributed to the following reasons: when the enterprise has private revenue information, the service provider can still design the contract such that the low-revenue type

enterprise obtains zero utility. However, to prevent the high-revenue type enterprise from gaining positive utility by pretending to be the low-revenue type enterprise, the service provider must eliminate the incentive to imitate. This is achieved by ensuring that high-revenue type enterprises earn the same profit by selecting a contract that matches their type as they would if they imitated low-revenue type enterprises. In other words, the service provider chooses to pay a portion of the information rent to the revenue-advantaged enterprise in order to achieve a separation equilibrium.

5.2.2 Revenue-Sharing Contract Offered by the Service Provider

In the following, we turn to the scenario where the service provider proposes a revenue-sharing contract in the presence of private revenue information held by the enterprise. The terms of the contract are denoted as (t, α), where t represents the moment when the software upgrade is provided, and α represents the share of revenue that the enterprise agrees to transfer to the service provider as compensation for the software upgrade. The contracting parties engage in two phases, namely contract design and contract execution, which are depicted in Fig. 6. During the contract design stage, the service provider offers a contract menu $\left\{(t_l^{rs}, \alpha_l^{rs}), (t_h^{rs}, \alpha_h^{rs})\right\}$ to the enterprise, where the subscript *rs* denotes the revenue-sharing contract is offered by the service provider. Here, $(t_l^{rs}, \alpha_l^{rs})$ and $(t_h^{rs}, \alpha_h^{rs})$ respectively represent the moment of software upgrade and the corresponding share of revenue transferred from the enterprise to the service provider for low-revenue and high-revenue types of enterprises.

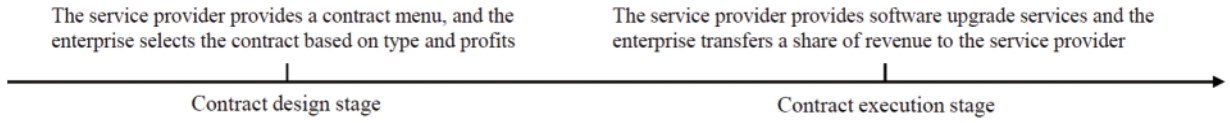

Figure 6: The time line using revenue-sharing contracts where the enterprise has privare revenue information

Thus, in the case where the enterprise has private revenue information, the service provider's problem is to design the optimal contract menu $\left\{(t_l^{rs}, \alpha_l^{rs}), (t_h^{rs}, \alpha_h^{rs})\right\}$ that satisfies the following:

$$\max_{\{t_n^{rs}, \alpha_n^{rs}\}_{n=l,h}} \quad V^{rs} = \pi_{2l}[\alpha_l^{rs} P(t_l^{rs}) - C[D(t_l^{rs})] - K] + \pi_{2h}[\alpha_h^{rs} P(t_h^{rs}) - C[D(t_h^{rs})] - K]$$

subject to

$$\begin{aligned}
&(1-\alpha_l^{rs})R_l \int_0^{t_l^{rs}} [1 - D(x)]\mathrm{d}x \geq 0 && (IR_l^{rs}) \\
&(1-\alpha_h^{rs})R_h \int_0^{t_h^{rs}} [1 - D(x)]\mathrm{d}x \geq 0 && (IR_h^{rs}) \\
&(1-\alpha_l^{rs})R_l \int_0^{t_l^{rs}} [1 - D(x)]\mathrm{d}x \geq (1-\alpha_h^{rs})R_l \int_0^{t_h^{rs}} [1 - D(x)]\mathrm{d}x && (IC_{lh}^{rs}) \\
&(1-\alpha_h^{rs})R_h \int_0^{t_h^{rs}} [1 - D(x)]\mathrm{d}x \geq (1-\alpha_l^{rs})R_h \int_0^{t_l^{rs}} [1 - D(x)]\mathrm{d}x && (IC_{hl}^{rs}) \\
&t_n^{rs},\ \alpha_n^{rs} \geq 0 \quad \forall n \in \{l, h\} && (NN_n^{rs})
\end{aligned} \tag{26}$$

Proposition 5.4, proved in Appendix B, characterizes the optimal revenue-sharing contract menu offered by the service provider when there is asymmetric revenue information from the enterprise.

Proposition 5.4. Under the condition that $0 < R_l < R_h < k\sigma$, using a revenue-sharing contract in the case where the enterprise has private revenue information, the optimal contract designed by the service provider is $\left\{(t_l^{rs}, \alpha_l^{rs}), (t_h^{rs}, \alpha_h^{rs})\right\}$, where

$$t_l^{rs} = t_l^* = \frac{\ln\left(1 - \frac{R_l}{k\sigma}\right)}{-\sigma} \tag{27}$$

$$t_h^{rs} = t_h^* = \frac{\ln\left(1 - \frac{R_h}{k\sigma}\right)}{-\sigma} \tag{28}$$

$$\alpha_l^{rs} = \alpha_h^{rs} = 1 \tag{29}$$

Proposition 5.4 shows that when using a revenue-sharing contract with private revenue information from the enterprise, the optimal software upgrade times for high and low-revenue enterprises are the same as those in the centralized decision case, with "no distortion at either the high or low end (the high-revenue or low-revenue type enterprise)". Additionally, both high and low-revenue enterprises are required to transfer all revenues to the service provider. These findings suggest that a revenue-sharing contract can achieve the optimal solution in the case of a centralized decision with full-information when there is asymmetric information about the enterprise's revenues.

The pay-per-time contract and the revenue-sharing contract yield quite different results when the enterprise has private revenue information. The reason is that in a revenue-sharing contract, the service provider treats both high and low-revenue enterprises equally, requiring both types to transfer all revenues. This prevents the high-revenue enterprise from imitating the low-revenue enterprise and obtaining positive profits, leading both types to obtain zero utility. Thus, the enterprise selects the contract that matches its true type regardless of revenue. Therefore, using a revenue-sharing contract under private revenue information achieves the optimal solution in the case of a centralized decision with full-information.

However, under a pay-per-time contract, the service provider values high-revenue enterprises more than low-revenue enterprises. As a result, the service provider takes all the revenue from the low-revenue enterprise and offers information rent to the high-revenue enterprise as an incentive to choose the contract that matches its true type. This leads to a phenomenon known as "no distortion at the high end and downward distortion at the low end," where the contract is accurate for high-revenue enterprises but distorted for low-revenue enterprises.

Fig. 7 plots the profits of the service provider offering a pay-per-time contract and a revenue-sharing contract when the enterprise has private revenue information. The two curved surfaces in Fig. 7 illustrate that, under the same enterprise scale and market distribution, the blue curved surface is consistently higher than the red curved surface. This indicates that the service provider can generate greater profits by providing a revenue-sharing contract than by providing a pay-per-time contract when the enterprise has private revenue information. The reason for this is that under a pay-per-time contract, the service provider must transfer part of the information rent to the high-revenue enterprise. However, by offering a revenue-sharing contract, the service provider not only avoids the need to transfer information rent but can also achieve the optimal solution similar to the centralized decision with full-information. Thus, when the enterprise has private revenue information, the service provider prefers to choose the revenue-sharing contract.

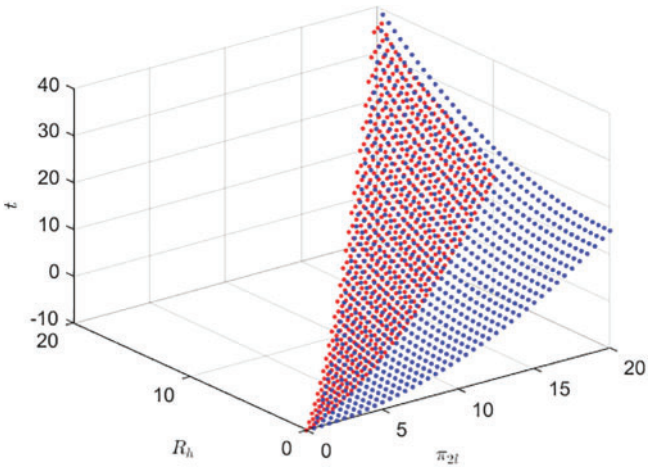

Figure 7: Profits for the service provider under the pay-per-time contracts and revenue-sharing contracts, $\sigma = 0.2, k = 100, K = 10, \pi_{2l} = 0.5, \pi_{2h} = 0.5$ (The blue surface is under the revenue-sharing contract, the red surface is under the pay-per-time contract)

5.3 Comparision of Optimal Solutions and Incentive Effect

Table 2 presents the optimal solutions obtained through the pay-per-time and revenue-sharing contracts. These contracts address the adverse selection problem in two scenarios where the service provider has private cost information and the enterprise has private revenue information. Next, we compared the results with those obtained from centralized decision-making with full-information to assess the incentive effects of the contracts.

Table 2: Comparison of the optimal software upgrade time

Contracts	The service provider has private cost information		The enterprise has private revenue information	
	Low-cost	High-cost	Low-revenue	High-revenue
Pay-per-time	$t_l^{pe} = t_l^{re} = t_l^*$	$t_h^{pe} = t_h^{re} < t_h^*$	$t_l^{ps} < t_l^*$	$t_h^{ps} = t_h^*$
Revenue-sharing	$t_l^{re} = t_l^{pe} = t_l^*$	$t_h^{re} = t_h^{pe} < t_h^*$	$t_l^{rs} = t_l^*$	$t_h^{rs} = t_h^*$

Notes: ^{pe}: The pay-per-time contracts provided by the enterprise. ^{re}: The revenue-sharing contracts provided by the enterprise. ^{ps}: The pay-per-time contracts provided by the service provider. ^{rs}: The revenue-sharing contracts provided by the service provider.*: Centralized decision with full-information.

Section 5.1 investigates the impact of contracts on software upgrade decisions when service providers hold private cost information. We found that the pay-per-time and revenue-sharing contracts yield the same optimal solutions under this scenario. That is, both contracts fail to achieve optimal incentives when the service provider is a high-cost type, leading to suboptimal solutions. Only when the service provider is a low-cost type do the pay-per-time and revenue-sharing contracts produce outcomes consistent with the centralized decision-making approach. These findings suggest that both contracts have similar incentive effects on service providers when cost information is asymmetric, but neither can fully achieve optimal incentives.

Section 5.2 examines the impact of contracts on software upgrade decisions when the enterprise has private revenue information. The results show that when the enterprise is a high-revenue type, the optimal solutions for both the pay-per-time and revenue-sharing contracts align with the centralized decision case. However, when the enterprise is a low-revenue type, the pay-per-time contract yields suboptimal solutions, while the revenue-sharing contract maintains optimal incentives. These findings suggest that the revenue-sharing contract provides superior incentives compared to the pay-per-time contract and can achieve optimal results in the presence of asymmetric revenue information about the enterprise.

The comparison of the optimal solutions and incentive effects in different information scenarios provides several managerial insights for software upgrade outsourcing:

Managing Information Asymmetry: Asymmetric information about costs and revenues can significantly impact the outcomes of software upgrade outsourcing. Service providers should strive to improve transparency regarding their cost structure, allowing enterprises to make more informed decisions. Similarly, enterprises should share relevant revenue information with service providers to align incentives and achieve better outcomes.

Contract Design: The choice of contract plays a crucial role in managing information asymmetry and aligning incentives. The revenue-sharing contract proves to be more effective in providing optimal incentives for both low-revenue and high-revenue enterprises. Enterprises should consider adopting revenue-sharing contracts to ensure service providers are motivated to deliver the best outcomes.

Risk Mitigation: Software upgrade outsourcing involves inherent risks associated with market fluctuations and potential losses due to outdated software. Enterprises should carefully assess these risks and develop risk mitigation strategies such as timely upgrades and collaborations with service providers. Effective risk management can help enterprises maintain their market position and competitiveness.

Collaborative Decision-Making: While centralized decision-making with full-information yields optimal solutions, achieving such an ideal scenario may be challenging in practice. However, enterprises and service providers can adopt collaborative decision-making approaches, sharing relevant information and working together to maximize joint profits. Regular communication and collaboration can lead to better outcomes and stronger partnerships.

6 Conclusions

With the rapid advancement of technology and automation, Industry 4.0 represents the integration of digital technologies and industrial processes, leading to increased efficiency, productivity, and competitiveness. Our research on software upgrade outsourcing aligns with the principles of Industry 4.0 by enabling enterprises to leverage external expertise and resources to optimize their software systems, thereby enhancing their ability to adapt and thrive in the digital era. This paper takes the impact of market fluctuations on software adaptability as an entry point and establishes a software upgrade outsourcing model between the enterprise and the service provider.

In the centralized decision with full-information, the service provider functions as the enterprise's information technology department, collaborating with the enterprise to optimize the expected profit of the system through joint decision-making. This approach yields the most favorable outcomes for software upgrades. There are two cases to consider when determining the optimal software upgrade time. In the first case, the optimal software upgrade time and expected system profits are contingent upon the scale of the enterprise within a specific range. As the enterprise expands, the optimal upgrade

time is delayed, and the system's expected profits increase. Additionally, the optimal software upgrade time is also affected by market fluctuations. When substantial market fluctuations occur, resulting in decreased revenue and increased software upgrade costs, the system's expected profits decline, prompting enterprises to opt for earlier software upgrades. In the second case, if the enterprise scale exceeds the aforementioned range, the enterprise will have a continuous increase in profits without software upgrades.

In decentralized decision-making with incomplete information, the enterprise and the service provider operate as two independent entities with inconsistent interests and asymmetric information. To avoid the adverse selection problem arising from the ex-ante asymmetric information about the service provider's cost and the enterprise's revenue, this paper proposes a software upgrade outsourcing model based on the principal-agent and uses well-designed contracts to screen the true type of the information superior party. Furthermore, we investigate the impact of pay-per-time and revenue-sharing contracts on the software upgrade timing decision and the profits of both parties. Our findings reveal that, in cases where the service provider has private cost information, the pay-per-time and revenue-sharing contracts are equivalent, and the optimal software upgrade time exhibits characteristics of "no distortion at the high end and downward distortion at the low end" compared to the optimal solution under centralized decision-making with full-information. In the cases where the enterprise has private revenue information, the optimal software upgrade time under the pay-per-time contract demonstrates characteristics of "no distortion at the high end and downward distortion at the low end," while the optimal software upgrade time under the revenue-sharing contract can achieve the optimal solution under centralized decision-making with full-information, showing "no distortion at either the high or low end." At this point, the incentive effect of the revenue-sharing contract is superior to that of the pay-per-time contract.

In summary, our proposed framework empowers all parties involved in software upgrade outsourcing contracts to make informed decisions, resulting in improved return on investment and increased operational efficiency. By considering the impact of information asymmetry and market fluctuations, enterprises can strategically determine the optimal software upgrade time and maximize their expected profits. This research contributes to the growing body of knowledge in the field of software upgrade outsourcing contract design and provides certain insights for practitioners and scholars.

Acknowledgement: Authors are very much thankful to the reviewers and Journal Authorities.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: supervision: C. W.; draft manuscript preparation: Z. W. Z. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Liu, C., Zheng, P., Xu, X. (2023). Digitalisation and servitisation of machine tools in the era of Industry 4.0: A review. *International Journal of Production Research, 61(12),* 4069–4101.
2. Hussain, M., Memon, T. D., Hussain, I., Memon, Z. A., Kumar, D. (2022). Fault detection and identification using deep learning algorithms in induction motors. *Computer Modeling in Engineering & Sciences, 133(2),* 435–470. https://doi.org/10.32604/cmes.2022.020583
3. Oztemel, E., Gursev, S. (2020). Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing, 31,* 127–182.
4. Rafferty, C., Smith, R. (2000). Making a prophet. *Computer Modeling in Engineering & Sciences, 1(1),* 151–159. https://doi.org/10.3970/cmes.2000.001.151
5. Furneaux, B., Mannina, S., Rieser, L. (2022). Responding to information system obsolescence: Should we upgrade or replace? *Journal of Computer Information Systems, 62(2),* 372–383.
6. Liu, Y., Xu, X. (2017). Industry 4.0 and cloud manufacturing: A comparative analysis. *Journal of Manufacturing Science and Engineering, 139(3),* 034701.
7. Stock, T., Seliger, G. (2016). Opportunities of sustainable manufacturing in Industry 4.0. *Procedia CIRP, 40,* 536–541.
8. Baron, D. P. (1979). The incentive problem and the design of investment banking contracts. *Journal of Banking & Finance, 3(2),* 157–175.
9. Baron, D. P., Holmström, B. (1980). The investment banking contract for new issues under asymmetric information: Delegation and the incentive problem. *The Journal of Finance, 35(5),* 1115–1138.
10. Sappington, D. E. M. (1991). Incentives in principal-agent relationships. *Journal of Economic Perspectives, 5(2),* 45–66.
11. Prikladnicki, R., Audy, J. L. N. (2012). Managing global software engineering: A comparative analysis of offshore outsourcing and the internal offshoring of software development. *Information Systems Management, 29(3),* 216–232.
12. Rajaeian, M. M., Cater-Steel, A., Lane, M. (2017). A systematic literature review and critical assessment of model-driven decision support for it outsourcing. *Decision Support Systems, 102,* 42–56.
13. Lacity, M. C., Khan, S. A., Willcocks, L. P. (2009). A review of the it outsourcing literature: Insights for practice. *The Journal of Strategic Information Systems, 18(3),* 130–146.
14. Kern, T., Willcocks, L. P., Lacity, M. (2002). Application service provision: Risk assessment and mitigation. *MIS Quarterly Executive, 1(2),* 113–126.
15. Richmond, W. B., Seidmann, A., Whinston, A. B. (1992). Incomplete contracting issues in information systems development outsourcing. *Decision Support Systems, 8(5),* 459–477.
16. Wang, E. T., Barron, T., Seidmann, A. (1997). Contracting structures for custom software development: The impacts of informational rents and uncertainty on internal development and outsourcing. *Management Science, 43(12),* 1726–1744.
17. Casado-Lumbreras, C., Colomo-Palacios, R., Ogwueleka, F. N., Misra, S. (2014). Software development outsourcing: Challenges and opportunities in Nigeria. *Journal of Global Information Technology Management, 17(4),* 267–282.
18. Bandyopadhyay, S., Pathak, P. (2007). Knowledge sharing and cooperation in outsourcing projects–A game theoretic analysis. *Decision Support Systems, 43(2),* 349–358.
19. Kornish, L. J. (2001). Pricing for a durable-goods monopolist under rapid sequential innovation. *Management Science, 47(11),* 1552–1561.
20. Bala, R., Carr, S. (2009). Pricing software upgrades: The role of product improvement and user costs. *Production and Operations Management, 18(5),* 560–580.
21. Whang, S. (1992). Contracting for software development. *Management Science, 38(3),* 307–324.

22. Elitzur, R., Gavious, A., Wensley, A. K. (2012). Information systems outsourcing projects as a double moral hazard problem. *Omega, 40(3),* 379–389.
23. Zhang, Z., Xu, X. (2017). Principal agent model based design and outsourcing of information value. *Cluster Computing, 20,* 67–79.
24. Pavlou, P. A., Liang, H., Xue, Y. (2007). Understanding and mitigating uncertainty in online exchange relationships: A principal-agent perspective. *MIS Quarterly, 31(1),* 105–136.
25. Grossman, S. J., Hart, O. D. (1992). An analysis of the principal-agent problem. In: *Foundations of insurance economics*, pp. 302–340. Dordrecht: Springer Netherlands.
26. Ciliberti, F., de Haan, J., de Groot, G., Pontrandolfo, P. (2011). CSR codes and the principal-agent problem in supply chains: Four case studies. *Journal of Cleaner Production, 19(8),* 885–894.
27. Cai, W., Singham, D. I. (2018). A principal–agent problem with heterogeneous demand distributions for a carbon capture and storage system. *European Journal of Operational Research, 264(1),* 239–256.
28. Yao, T., Jiang, B., Young, S. T., Talluri, S. (2010). Outsourcing timing, contract selection, and negotiation. *International Journal of Production Research, 48(2),* 305–326.
29. Banerjee, A. V., Duflo, E. (2000). Reputation effects and the limits of contracting: A study of the indian software industry. *The Quarterly Journal of Economics, 115(3),* 989–1017.
30. Benaroch, M., Lichtenstein, Y., Fink, L. (2016). Contract design choices and the balance of ex ante and ex post transaction costs in software development outsourcing. *MIS Quarterly, 40(1),* 57–82.
31. Gopal, A., Koka, B. R. (2010). The role of contracts on quality and returns to quality in offshore software development outsourcing. *Decision Sciences, 41(3),* 491–516.
32. Huang, H., Hu, M., Xu, H. (2023). Time-and-materials or revenue-sharing? Implications of monitoring for software outsourcing contract. *Information & Management, 60(3),* 103776.
33. Wang, C., Deng, L., Zhou, W. (2022). "We missed you!": A joint optimization strategy of appointment window and reminder sending. *Computers & Industrial Engineering, 169,* 108198.

Appendix A.

Proof of Proposition 4.1. Firstly, the actual revenue function of the enterprise is deformed:

$$P(t) = R\int_0^t [1 - D(x)]\mathrm{d}x = R\int_0^t e^{-\sigma x}\mathrm{d}x = R\int_0^t -\frac{1}{\sigma}\mathrm{d}e^{-\sigma x} = \frac{R}{\sigma}(1 - e^{-\sigma t}) = \frac{R}{\sigma}D(t) \tag{30}$$

Secondly, convert $W(t)$ into a function of D, denoted as $W(D)$

$$W(D) = P(t) - C(D) - K = \frac{R}{\sigma}D(t) - \frac{1}{2}kD^2(t) - K \tag{31}$$

Thirdly, the first and second derivatives of $W(D)$ with respect to D are calculated, respectively:

$$\frac{dW(D)}{dD} = \frac{R}{\sigma} - kD(t) \tag{32}$$

$$\frac{d^2W(D)}{dD^2} = -k \tag{33}$$

Since $k > 0$, the second derivative $\frac{d^2W(D)}{dD^2}$ is negative, indicating that $W(D)$ is strictly concave with respect to D. By setting the first derivative $dW(D)/dD = 0$, we obtain the optimal software deviation degree:

$$D^*(t) = \frac{R}{k\sigma} \tag{34}$$

Since D is a monotonically increasing function of t, we can solve for the optimal software upgrade time t^* using the optimal deviation degree, which can be divided into two cases:

(1) If $\frac{R}{k\sigma} \geq 1$, then $D(t^*) = 1 - e^{-\sigma t^*} \geq 1$, which implies that t^* must be infinity.

(2) If $\frac{R}{k\sigma} < 1$, then $D(t^*) = 1 - e^{-\sigma t^*} = \frac{R}{k\sigma}$, and t^* can be calculated as $t^* = \frac{\ln\left(1 - \frac{R}{k\sigma}\right)}{-\sigma}$.

When $t^* = \frac{\ln\left(1 - \frac{R}{k\sigma}\right)}{-\sigma}$, the discussion can be divided into three cases:

(1) If $1 - \frac{R}{k\sigma} > 1$, then $\frac{\ln\left(1 - \frac{R}{k\sigma}\right)}{-\sigma} < 0$, and the constraint $t \geq 0$ is not satisfied, so it is not considered.

(2) If $0 < 1 - \frac{R}{k\sigma} \leq 1$, then $t^* = \frac{\ln\left(1 - \frac{R}{k\sigma}\right)}{-\sigma} > 0$, which satisfies the conditions.

(3) If $1 - \frac{R}{k\sigma} \leq 0$, it does not meet the basic conditions of the logarithmic function and is not considered.

It can be observed that for the centralized decision-making with full-information, the optimal software upgrade time exists and is positive only when the value of R satisfies $0 \leq R < k\sigma$.

Finally, we can substitute t^* into the objective function to obtain the optimal expected profits of the system:

$$W(t^*) = P(t^*) - C[D(t^*)] - K = \frac{R}{\sigma}D^*(t) - \frac{1}{2}k[D^*(t)]^2 - K = \frac{R^2}{2k\sigma^2} - K \tag{35}$$

The optimal expected profits of the system are $\frac{R^2}{2k\sigma^2} - K$.

Calculate the first derivatives of t^* and $W(t^*)$ to R, respectively:

$$\frac{\partial t^*}{\partial R} > 0, \frac{\partial W(t^*)}{\partial R} > 0 \tag{36}$$

The first derivatives of t^* and $W(t^*)$ with respect to R are positive, indicating that the optimal software upgrade time and expected profits of the system increase with R.

We can observe the molecule $\ln\left(1-\frac{R}{k\sigma}\right)$ in t^*. As σ increases, $k\sigma$ and $\left(1-\frac{R}{k\sigma}\right)$ also increase. Since the logarithmic function is monotonically increasing, and $0 \leq R < k\sigma$, $\ln\left(1-\frac{R}{k\sigma}\right)$ becomes larger as $\left(1-\frac{R}{k\sigma}\right)$ becomes larger and is a negative number. The denominator $-\sigma$ becomes smaller as σ increases. Therefore, the greater the market fluctuations, the smaller the optimal software upgrade time t^* and the earlier the software upgrade.

Appendix B.

Proof of Proposition 5.1. Suppose the information rent $\Delta_n^{pe} = T_n^{pe} - \frac{1}{2}k_n[D(t_n^{pe})]^2 - K$, then the information rent of the low-cost type service provider is

$$\Delta_l^{pe} = T_l^{pe} - \frac{1}{2}k_l[D(t_l^{pe})]^2 - K \tag{37}$$

The information rent of high-cost type service providers is

$$\Delta_h^{pe} = T_h^{pe} - \frac{1}{2}k_h[D(t_h^{pe})]^2 - K \tag{38}$$

In cases where the service provider has private cost information, it is common for the low-cost type service provider to imitate the high-cost type service provider. Consequently, it can be deduced that

$$\begin{aligned} T_h^{pe} - \frac{1}{2}k_l[D(t_h^{pe})]^2 - K &= T_h^{pe} - \frac{1}{2}k_h[D(t_h^{pe})]^2 + \frac{1}{2}(k_h - k_l)[D(t_h^{pe})]^2 - K \\ &= \Delta_h^{pe} + \frac{1}{2}(k_h - k_l)[D(t_h^{pe})]^2 \end{aligned} \tag{39}$$

Although an effective contract can reduce the information rent Δ_h^{pe} of the high-cost service provider to zero, the low-cost provider can still gain an additional utility $\frac{1}{2}(k_h - k_l)[D(t_h^{pe})]^2 > 0$ by imitating their high-cost counterparts, which can be translated to the information rent Δ_l^{pe} of the low-cost providers under information asymmetry.

The expressions for information rent, Δ_l^{pe} and Δ_h^{pe}, allow us to reformulate the individual rational constraints (IR) and incentive compatibility constraints (IC) as follows:

$$\Delta_l^{pe} \geq 0 \tag{40}$$

$$\Delta_h^{pe} \geq 0 \tag{41}$$

$$\Delta_l^{pe} \geq \Delta_h^{pe} + \frac{1}{2}k_h[D(t_h^{pe})]^2 - \frac{1}{2}k_l[D(t_h^{pe})]^2 \tag{42}$$

$$\Delta_h^{pe} \geq \Delta_l^{pe} + \frac{1}{2}k_l[D(t_l^{pe})]^2 - \frac{1}{2}k_h[D(t_l^{pe})]^2 \tag{43}$$

The enterprise aims to design a contract that meets the incentive constraints and minimizes the information rent paid. Thus, it is essential that the constraint Δ_h^{pe} is tight, that is, $\Delta_h^{pe} = 0$. Substituting $\Delta_h^{pe} = 0$ into Eq. (38), we obtain

$$T_h^{pe} = \frac{1}{2}k_h[D(t_h^{pe})]^2 + K \tag{44}$$

Substitute $\Delta_h^{pe} = 0$ into constraint (42), we obtain

$$\Delta_l^{pe} = \frac{1}{2}k_h[D(t_h^{pe})]^2 - \frac{1}{2}k_l[D(t_h^{pe})]^2 \tag{45}$$

According to Eq. (37), it can be solved

$$T_l^{pe} = \frac{1}{2}k_h[D(t_h^{pe})]^2 - \frac{1}{2}k_l[D(t_h^{pe})]^2 + \frac{1}{2}k_l[D(t_l^{pe})]^2 + K \tag{46}$$

Finally, substitute T_h^{pe} and T_l^{pe} into the enterprise's objective function and solve for $\frac{\partial U^{pe}}{\partial t_l^{pe}} = 0$ and $\frac{\partial U^{pe}}{\partial t_h^{pe}} = 0$ to obtain

$$t_l^{pe} = \frac{\ln\left(1 - \frac{R}{k\sigma}\right)}{-\sigma} \tag{47}$$

$$t_h^{pe} = \frac{\ln\left[1 - \frac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l}k_l)}\right]}{-\sigma} \tag{48}$$

The value of t_l^{pe} can be divided into three cases:

(1) If $1 - \frac{R}{k_l\sigma} > 1$, then t_l^{pe} is negative and violates the constraint (40), so it is not a valid solution.

(2) If $0 < 1 - \frac{R}{k_l\sigma} \leq 1$, then $0 \leq R < k_l\sigma$ and $t_l^{pe} \geq 0$ meet the necessary constraints.

(3) If $1 - \frac{R}{k_l\sigma} < 0$, then the logarithmic function's basic conditions are not met, and this situation is also not considered.

The value of t_h^{pe} is also discussed in three cases:

(1) If $1 - \frac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l}k_l)} > 1$, t_h^{pe} is negative and does not meet the constraint (41), so it is not considered.

(2) If $0 < 1 - \frac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l}k_l)} \leq 1$, then $0 \leq R < \frac{\sigma(k_h - \pi_{1l}k_l)}{1-\pi_{1l}}$ and $t_h^{pe} \geq 0$ meets the constraint (41).

(3) If $1 - \frac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l}k_l)} < 0$, it does not meet the basic conditions of the logarithmic function, and this case is not considered.

To determine the range of R further, we can compare $\frac{\sigma(k_h - \pi_{1l}k_l)}{1-\pi_{1l}}$ and $k_l\sigma$, which gives us the inequality $\frac{\sigma(k_h - \pi_{1l}k_l)}{1-\pi_{1l}} > k_l\sigma$. Therefore, we can conclude that R should satisfy the condition $0 \leq R < k_l\sigma$.

By substituting t_h^{pe} and t_l^{pe} into Eqs. (44) and (46), we can obtain T_h^{pe} and T_l^{pe}, respectively.

Take the first derivative of t^* with respect to k

$$\frac{\partial t^*}{\partial k} = -\frac{R}{\sigma^2 k^2\left(1-\frac{R}{k\sigma}\right)} < 0 \tag{49}$$

As a result, the variable cost coefficient has an inverse proportionality relationship with the optimal software upgrade time, which leads to the deduction that $t_h^* < t_l^*$.

Comparing t_l^{pe} with t_l^*, we can get

$$t_l^{pe} = t_l^* = \frac{\ln\left(1-\frac{R}{k_l\sigma}\right)}{-\sigma} \tag{50}$$

Consider using a pay-per-time contract when there is asymmetry in the service provider's cost information

$$t_h^{pe} = \frac{\ln\left[1-\frac{R(1-\pi_{1l})}{\sigma(k_h-\pi_{1l}k_l)}\right]}{-\sigma} = \frac{\ln\left[1-\frac{R(1-\pi_{1l})}{\sigma[k_h-k_l+k_l(1-\pi_{1l})]}\right]}{-\sigma} < \frac{\ln\left(1-\frac{R}{k_h\sigma}\right)}{-\sigma} = t_h^* \tag{51}$$

Proof of Proposition 5.2. Suppose the information rent $\Delta_n^{re} = \alpha_n^{re}P(t_n^{re}) - \frac{1}{2}k_n[D(t_n^{re})]^2 - K$, then the information rent of the low-cost type service provider is

$$\Delta_l^{re} = \alpha_l^{re}P(t_l^{re}) - \frac{1}{2}k_l[D(t_l^{re})]^2 - K \tag{52}$$

The information rent of the high-cost type service provider is

$$\Delta_h^{re} = \alpha_h^{re}P(t_h^{re}) - \frac{1}{2}k_h[D(t_h^{re})]^2 - K \tag{53}$$

Based on the expressions of information rent Δ_l^{re} and Δ_h^{re}, we can reformulate the individual rational constraints (IR) and incentive compatibility constraints (IC) as follows:

$$\Delta_l^{re} \geq 0 \tag{54}$$

$$\Delta_h^{re} \geq 0 \tag{55}$$

$$\Delta_l^{re} \geq \Delta_h^{re} + \frac{1}{2}k_h[D(t_h^{re})]^2 - \frac{1}{2}k_l[D(t_h^{re})]^2 \tag{56}$$

$$\Delta_h^{re} \geq \Delta_l^{re} + \frac{1}{2}k_l[D(t_l^{re})]^2 - \frac{1}{2}k_h[D(t_l^{re})]^2 \tag{57}$$

Constraint (55) is tight, meaning that $\Delta_h^{re} = 0$. Reducing $\Delta_h^{re} = 0$ to 0 does not affect the constraint, but it would lower the enterprise's objective function, which conflicts with the maximization of the objective function. Thus, by substituting $\Delta_h^{re} = 0$ into Eq. (53), we obtain

$$\alpha_h^{re} P(t_h^{re}) = \frac{1}{2} k_h [D(t_h^{re})]^2 + K \tag{58}$$

$\alpha_h^{re} P(t_h^{re})$ represents the fee paid by the enterprise to the high-cost type service provider. Substituting the expression for Δ_h^{re} into constraint (56) and combining it with Eq. (52) yields

$$\alpha_l^{re} P(t_l^{re}) = \frac{1}{2} k_h [D(t_h^{re})]^2 + \frac{1}{2} k_l [D(t_l^{re})]^2 - \frac{1}{2} k_l [D(t_h^{re})]^2 + K \tag{59}$$

$\alpha_l^{re} P(t_l^{re})$ represents the fee paid by the enterprise to the low-cost type service provider. Bringing Eqs. (58)–(59) into the objective function and solving for $\frac{\partial U^{re}}{\partial t_l^{re}} = 0$ and $\frac{\partial U^{re}}{\partial t_h^{re}} = 0$, we obtain

$$t_l^{re} = \frac{\ln\left(1 - \frac{R}{k_l \sigma}\right)}{-\sigma} \tag{60}$$

$$t_h^{re} = \frac{\ln\left[1 - \frac{R(1-\pi_{1l})}{\sigma(k_h - \pi_{1l} k_l)}\right]}{-\sigma} \tag{61}$$

Similar to the proof of Proposition 5.1, we discuss the three cases of t_l^{re} and t_h^{re} respectively to deduce that R satisfies the condition $0 \leq R < k_l \sigma$. Finally, we can find α_h^{re} and α_l^{re} by plugging in t_l^{re} and t_h^{re} into Eqs. (58)–(59).

Proof of Proposition 5.3. Suppose the information rent $\Delta_n^{ps} = P(t_n^{ps}) - T_n^{ps}$, then the information rent of the low-revenue type enterprise is

$$\Delta_l^{ps} = P(t_l^{ps}) - T_l^{ps} \tag{62}$$

The information rent of the high-revenue type enterprise is

$$\Delta_h^{ps} = P(t_h^{ps}) - T_h^{ps} \tag{63}$$

According to the expressions of information rent Δ_l^{ps} and Δ_h^{ps}, individual rational constraints (IR) and incentive compatibility constraints (IC) can be reformulated as

$$\Delta_l^{ps} \geq 0 \tag{64}$$

$$\Delta_h^{ps} \geq 0 \tag{65}$$

$$\Delta_l^{ps} \geq \Delta_h^{ps} + R_l \int_0^{t_h^{ps}} [1 - D(x)]dx - R_h \int_0^{t_h^{ps}} [1 - D(x)]dx \tag{66}$$

$$\Delta_h^{ps} \geq \Delta_l^{ps} + R_h \int_0^{t_l^{ps}} [1 - D(x)]dx - R_l \int_0^{t_l^{ps}} [1 - D(x)]dx \tag{67}$$

As mentioned previously, the service provider in a design contract seeks to minimize the payment of information rent in order to maximize its own interests. Hence, constraint (64) is tight, which means that $\Delta_l^{ps} = 0$. Substituting this value into Eq. (62), we obtain

$$T_l^{ps} = P(t_l^{ps}) \tag{68}$$

Then, substituting $\Delta_l^{ps} = 0$ into constraint (67), we obtain

$$\Delta_h^{ps} = R_h \int_0^{t_l^{ps}} [1 - D(x)]dx - R_l \int_0^{t_l^{ps}} [1 - D(x)]dx \tag{69}$$

Substitute the expression for Δ_h^{ps} into Eq. (63), and we can figure out

$$T_h^{ps} = P(t_h^{ps}) - R_h \int_0^{t_l^{ps}} [1 - D(x)]dx - R_l \int_0^{t_l^{ps}} [1 - D(x)]dx \tag{70}$$

Bringing T_l^{ps} and T_h^{ps} into the objective function, by solving for $\frac{\partial V^{ps}}{\partial t_l^{ps}} = 0$, $\frac{\partial V^{ps}}{\partial t_h^{ps}} = 0$, can be calculated

$$t_l^{ps} = \frac{\ln\left[1 - \frac{R_l - R_h\pi_{2h}}{(1-\pi_{2h})k\sigma}\right]}{-\sigma} \tag{71}$$

$$t_h^{ps} = \frac{\ln\left(1 - \frac{R_h}{k\sigma}\right)}{-\sigma} \tag{72}$$

Similarly, the results of t_l^{ps} and t_h^{ps} are discussed in three cases respectively. We conclude that if the optimal upgrade time exists, then R_h and R_l satisfy the requirements of $0 \le R_h < k\sigma, \pi_{2h}R_h \le R_l < R_h$, respectively. Finally, we substitute t_l^{ps} and t_h^{ps} into Eqs. (68) and (70) to obtain T_l^{ps} and T_h^{ps}, respectively.

$$t_l^{ps} = \frac{\ln\left[1 - \frac{R_l - R_h\pi_{2h}}{(1-\pi_{2h})\,k\sigma}\right]}{-\sigma} = \frac{\ln\left[1 - \frac{R_l(1-\pi_{2h}) + (R_l - R_h)\pi_{2h}}{(1-\pi_{2h})k\sigma}\right]}{-\sigma} < \frac{\ln\left(1 - \frac{R_l}{k\sigma}\right)}{-\sigma} = t_l^* \tag{73}$$

Proof of Proposition 5.4. Define the information rent $\Delta_n^{rs} = (1 - \alpha_n^{rs})R_n \int_0^{t_n^{rs}} [1 - D(x)]\mathrm{d}x$, and then the information rent of the low-revenue type enterprise is

$$\Delta_l^{rs} = (1 - \alpha_l^{rs})R_l \int_0^{t_l^{rs}} [1 - D(x)]\mathrm{d}x \tag{74}$$

The information rent of the high-revenue type enterprise is

$$\Delta_h^{rs} = (1 - \alpha_h^{rs})R_h \int_0^{t_h^{rs}} [1 - D(x)]\mathrm{d}x \tag{75}$$

According to the expressions of information rent Δ_l^{rs} and Δ_h^{rs}, individual rational constraints (IR) and incentive compatibility constraints (IC) can be reformulated as

$$\Delta_l^{rs} \ge 0 \tag{76}$$

$$\Delta_h^{rs} \geq 0 \tag{77}$$

$$\Delta_l^{rs} \geq \Delta_h^{rs} + (1-\alpha_l^{rs})(R_l - R_h)\int_0^{t_h^{rs}}[1-D(x)]\mathrm{d}x \tag{78}$$

$$\Delta_h^{rs} \geq \Delta_l^{rs} + (1-\alpha_l^{rs})(R_h - R_l)\int_0^{t_l^{rs}}[1-D(x)]\mathrm{d}x \tag{79}$$

To minimize information rent and maximize profits, the service provider aims to design a contract that pays as little information rent as possible. Thus, the constrain (76) is tight. Substituting $\Delta_l^{rs} = 0$ into Eq. (74), we can get

$$\alpha_l^{rs} = 1 \tag{80}$$

Bringing $\Delta_l^{rs} = 0$, $\alpha_l^{rs} = 1$ into constraint (79) and combining it with Eq. (75) yields

$$\alpha_h^{rs} = 1 \tag{81}$$

Bringing T_l^{rs} and T_h^{rs} into the objective function, by solving for $\frac{\partial V^{rs}}{\partial t_l^{rs}} = 0$, $\frac{\partial V^{rs}}{\partial t_h^{rs}} = 0$, can be calculated

$$t_l^{rs} = \frac{\ln\left(1-\frac{R_l}{k\sigma}\right)}{-\sigma} \tag{82}$$

$$t_h^{rs} = \frac{\ln\left(1-\frac{R_h}{k\sigma}\right)}{-\sigma} \tag{83}$$

The results for t_l^{rs} and t_h^{rs} are discussed in three cases. It is concluded that if the optimal software upgrade time exists, R_h must satisfy the condition $0 \leq R_h < k\sigma$.