Naveen Jindal School of Management

2013-05-03

Optimal Software Reuse in Incremental Software Development: A Transfer Pricing Approach

UTD AUTHOR(S): Milind Dawande and Vijay S. Mookerjee

Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Optimal Software Reuse in Incremental Software Development: A Transfer Pricing Approach

Yasin Ceran, Milind Dawande, Dengpan Liu, Vijay Mookerjee

To cite this article:
Yasin Ceran, Milind Dawande, Dengpan Liu, Vijay Mookerjee (2014) Optimal Software Reuse in Incremental Software Development: A Transfer Pricing Approach. Management Science 60(3):541-559. http://dx.doi.org/10.1287/mnsc.2013.1757

MANAGEMENT SCIENCE
Vol. 60, No. 3, March 2014, pp. 541–559
ISSN 0025-1909 (print) | ISSN 1526-5501 (online)

http://dx.doi.org/10.1287/mnsc.2013.1757

Optimal Software Reuse in Incremental Software Development: A Transfer Pricing Approach

Yasin Ceran
Leavey School of Business, Santa Clara University, Santa Clara, California 95053, yceran@scu.edu

Milind Dawande
Naveen Jindal School of Management, University of Texas at Dallas, Richardson, Texas 75083, milind@utdallas.edu

Dengpan Liu
College of Business, Iowa State University, Ames, Iowa 50011, dliu@iastate.edu

Vijay Mookerjee
Naveen Jindal School of Management, University of Texas at Dallas, Richardson, Texas 75083, vijaym@utdallas.edu

This study develops optimal transfer pricing schemes that manage software reuse in incremental software development, namely, a development regime wherein users begin utilizing parts of the system that are released to them even before the system is entirely completed. In this setting, conflicts can arise between developers and users from divergent interests concerning the release of functionalities in the project. The release of functionalities is influenced by reuse, i.e., the effort spent by the development team to write code that can be reused within the same project or in future projects. For example, the development team may choose to spend extra effort to make certain portions of the system reusable because doing so could reduce the effort needed to develop the entire system. However, the additional effort spent on reuse could delay the release of certain critical functionality, making such a strategy suboptimal for the users. Thus, optimal reuse decisions for developers and users could be different. In addition, from the firm's perspective, reuse decisions must not only balance the objectives of developers and users for the current project, but reuse effort may be spent to benefit future projects. Our study also highlights the fact that reuse may not always be beneficial for the firm. To this end, we consider different instances of the user–developer conflict and provide transfer pricing schemes that operate under information asymmetry and achieve two key properties: firm-level optimality and truth revelation.

Keywords: software reuse; transfer pricing; information asymmetry; conflict resolution
History: Received June 22, 2011; accepted October 1, 2012, by Sandra Slaughter, information systems.
Published online in *Articles in Advance* August 19, 2013.

1. Introduction

The development process of large software systems raises various important managerial challenges. One such challenge is the presence of incongruent goals among the various stakeholders involved in the development process (Banker and Kemerer 1992; Gurbaxani and Kemerer 1989, 1990). Our study investigates user–developer conflicts concerning the release of functionality in a software project. We consider an in-house software development project where functionality is incrementally released to end users. In an incremental release regime, parts of the system being developed become usable and deliver value even before the system is entirely completed (Liu et al. 2007). In this setting, conflicts between developers and end users can arise concerning the release times of the functionalities. The driver of such conflict is software reuse: *although the effort on software reuse can delay the release of critical functions, it can speed up the delivery of the complete system*. Software reuse is "the systematic use of existing software assets to construct new assets or products" (Mohagheghi and Conradi 2008, p. 3). Reuse can affect both the development effort and the quality of a software system (Lim 1994). While reusable software assets could include code, executables, designs, architectures, etc., our focus here is on code reuse. A piece of code can either be created to explicitly perform a specific function or developed with more general capabilities with a view toward future reuse. For example, in a system requiring multiple varieties of a sorting algorithm, separate code can be written to perform each sorting function (e.g., rank students by grades, or employees by salaries, etc.). Alternatively, general-purpose code can be written to take as input a set of items to sort and a sorting criterion; each specific sorting function can then be created by extending (or specializing) this general

purpose code. A real example of successful code reuse is seen in CelsiusTech Systems, a Swedish naval defense contractor (Mohagheghi 2004). This firm was awarded two new contracts for systems that were to be built in parallel. The project management team first developed an initial code base that was common to the two systems. Then, the code base was not only reused and extended to create the two systems, but later, it also resulted in the successful completion of seven naval systems that were delivered on schedule and within budget. Unlike the case of CelsiusTech Systems, the *Fly Through* simulator project at the Boeing company provides an example of a case where code reuse was not aggressively pursued (Mattikalli 2011). Instead, each new generation of the system was created from scratch, leading to a higher total software development cost.

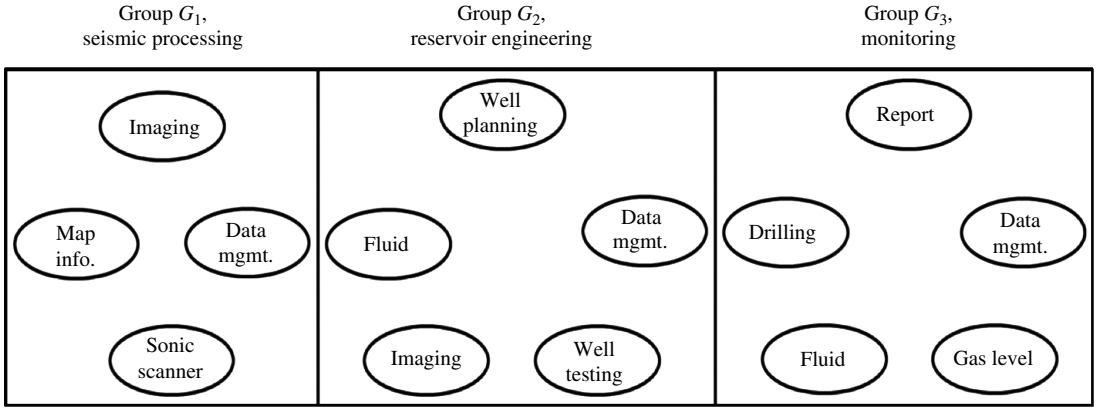
1.1. A Motivating Example from Oil Drilling Software

To motivate the problem, we report on the case of a real software project that was carried out within a big oil exploration company. The project consisted of migrating a large legacy oil drilling application to a Microsoft Windows-based system. The application facilitates the documenting and analysis of data generated during a drilling operation. In this application, an *element*, which is associated with a specific drilling tool, documents and analyzes the measurements being conducted while the tool is in operation. A *group*, which consists of a set of elements, is used to generate reports for users. The entire software application consisted of about 750 groups and 80 elements. The groups were to be delivered to users in a fixed sequence that was agreed upon between the user department and the development team. Data collected on the value (to the end user) provided by each completed group showed that the value of a group was independent of the presence of other completed groups. Furthermore, end users did not associate any significant value or functionality to individual elements; value to the end users accrued only when a group was released, i.e., when all elements of that group were created. In Figure 1, for illustrative purposes, we depict three groups (and some of their constituting elements) in the oil drilling application, namely, seismic processing, reservoir engineering, and monitoring. As shown in the figure, an element can be needed by more than one group; for instance, the element Fluid (used to identify the content of the fluid sample taken from the wells) is needed by both reservoir engineering and monitoring.

To explain the notion of reuse and specialization, consider the task of making the element Fluid reusable while developing it for its usage in reservoir engineering. To achieve this, some additional effort (*reuse effort*) is needed while creating the element for this group. This reuse effort must be spent during its creation, and not after the element has been created. Such a reuse strategy, referred to as *development with reuse*, is typical in practice (Poulin 1995, Lynex and Layzell 1998, Ramachandran 2005). If Fluid is made reusable, it can be specialized for monitoring with some additional effort (*specialization effort*). For instance, such specialization effort may be required to enable Fluid to handle reservoir depletion, effectiveness of water and gas injection, and changes in fluid contacts—specific tasks that are required for monitoring, but not needed for reservoir engineering. On the other hand, if element Fluid is not made reusable, the element would need to be recreated for deployment in monitoring.

To highlight the nature of the conflict concerning release times of the groups, consider two key stakeholders: developers and end users. Typically, users do not know (or care about) reuse details. Users are interested in receiving important groups (functionalities) as early as possible, even if this sometimes means that the delivery of the complete system is delayed. On the other hand, developers are interested

Figure 1 **An Illustrative Example: Elements *Data Management*, *Fluid*, and *Imaging* Are Considered for Reuse**



in minimizing the value of the overall effort spent on the project. For example, although some reuse effort would need to be spent to make Fluid reusable when developing it for reservoir engineering, this choice could pay off for the development team because this element would not have to be recreated for monitoring. Instead, with a relatively small amount of specialization effort, it may be possible to extend Fluid for use in monitoring. However, the extra (reuse) effort in making Fluid reusable would delay the release of reservoir engineering. This delay would hurt users if they urgently needed the functionality provided by reservoir engineering.

1.2. Contributions

As seen from the above example, the release times of the various groups in a software project (and hence the underlying reuse decisions) may give rise to possible disagreements and conflicts between users and developers, often leading to suboptimal outcomes for the firm. The aim of this research is to develop optimal transfer pricing (TP) schemes to manage software reuse in incremental software development. These pricing schemes are developed from the perspective of the firm that is interested in balancing the objectives of developers and users, while creating reusable assets for use in future projects. A key contribution is that the proposed schemes operate under information asymmetry; that is, the firm does not need to know the valuations of the users for the various functionalities. Furthermore, users are shielded from the architectural details of the project and the technical complexities of software reuse. By mapping reuse decisions to group release times, we ensure that users can directly evaluate release times (rather than reuse decisions). Although our work shares some similarities with the research on goal misalignment in software development, the focus of previous research has not been on software reuse (Wang et al. 1997). We propose optimal transfer pricing schemes under four settings: *single user and single developer* (*SU-SD*), *multiple users and single developer* (*MU-SD*), *single user and multiple developers* (*SU-MD*), and *multiple users and multiple developers* (*MU-MD*). These schemes are shown to possess two important properties: (1) firm-level optimality and (2) truth revelation under information asymmetry.

Transfer pricing provides a useful tool to management to coordinate the effective use of resources within a firm. TP is especially useful in a scenario where the firm has information of the cost of resources whereas divisions of the firm (that constitute the demand for these resources) have private information on the value of these resources. Transfer prices allow management to make firm-optimal decisions in the absence of full information (Westland 1992). Transfer prices act as a coordinating mechanism for optimally allocating resources within an organization (Baiman et al. 2007) and a considerable body of literature exists on this subject; see, for example, Groves and Loeb (1979), Vaysman (1998), and Baldenius et al. (2004).

As a specific example of the resource allocation problem, transfer prices have been used for charging a fair price for the services provided to multiple divisions of a firm (Miller and Buckman 1987). Here, the charge is calculated under full information. However, the charge can also be calculated when information asymmetry is considered (e.g., Harris et al. 1982, Banker and Datar 1992). The mechanism in Harris et al. (1982) induces truth telling by the divisional managers of the firm. Banker and Datar (1992) extend previous work to a situation where the division managers can collude at the expense of the firm and propose a mechanism to operate optimally under such collusion.

In the information systems (IS) literature, transfer pricing is used among the departments of a firm to coordinate a balanced utilization of information technology services that are offered as a public good. Tan and Mookerjee (2005) use transfer prices (calculated under full information) to balance the spending on advertising and information technology. Recently, Kumar et al. (2010) designed a pricing mechanism for a peer-to-peer network that allows users to effectively share the computing resources of a firm. In our paper, we consider multiple users (user departments) with different preferences over the release times of the functionalities of a software development project. Here, the preferences of a particular user can exert negative externalities on another user. Even if the users have a common preference for the release times of functionalities (e.g., the case of a *single* user), this preference may result in negative externalities for the developer and for the firm.

Designing a mechanism, as proposed by Harris et al. (1982), that deals with solving the direct revelation game can be complicated and expensive to implement (Baiman et al. 2007). To simplify the design of a transfer pricing mechanism, we can view the firm as an "economy" and use market power for resource allocation. In a few studies, researchers have developed and applied internal markets to solve resource allocation problems within a firm (Stuart 2005). As an example, Baiman et al. (2007) designed optimal managerial incentives when an internal auction market is used to allocate resources. In our paper, in the case of multiple users, we deploy an internal auction designed by the firm to elicit the preferences of users concerning the sequence in which functionalities should be released. We show that this process provides solutions that are optimal for the firm.

The rest of this paper is organized as follows. Section 2 defines the notation and formulates the basic models. Section 3 examines TP schemes that, in

the presence of information asymmetry, operate optimally from the perspective of the firm. Section 4 provides discussions for possible practical extensions of these schemes. Section 5 concludes this paper.

2. Model Development

We first introduce the notation and then provide the formulations for the basic models for the user, the developer, and the firm.

2.1. Notation and Assumptions

In general, a group may contain several elements and each element may be required by several groups (see Figure 1). A group is said to be completed if all the elements required by that group are developed. The software system under consideration consists of *m groups* (functionalities) $G_1, G_2, \ldots, G_m$, with each group consisting of a subset of *elements*. Let $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$ denote the set of elements, and let $\mathcal{I} = \{1, 2, \ldots, n\}$ be the corresponding index set. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$ and $\mathcal{K} = \{1, 2, \ldots, m\}$ denote the set of groups and the corresponding index set, respectively. We assume that a group is released for end-users as soon as it is completed.

We refer to the membership of an element in a group as that element's *usage*. We assume that each element has at most one usage per group.[1] For example, if element M_i is associated with γ_i groups, then this element has exactly γ_i distinct usages (one for each group it is associated with). Let r_i be the development time of M_i, $i = 1, 2, \ldots, n$. Based on our discussion earlier related to the physical implementation of reuse, let $\alpha_i \in [0, 1]$ denote the fraction of the development time needed to make M_i ready for reuse. Let $\beta_{ik} \in [0, 1)$ denote the fraction of the development time needed to make M_i specialized for its usage at group G_k. The additional time needed to make M_i ready for reuse is $\alpha_i r_i$, and is referred to as its *reuse time*. Subsequently, the time needed to specialize this reusable element for use in a (different) group, say, G_k, is $\beta_{ik} r_i$, and is referred to as its *specialization time* for that group. Thus, if an element M_i, required by a group, say, G_k, has been made ready for reuse at an earlier group in the sequence, then only the specialization time $\beta_{ik} r_i$ is incurred for developing M_i in G_k. Otherwise, (a) if M_i is made ready for reuse at G_k, then a development time of $(1 + \alpha_i) r_i$ is incurred for its development in G_k; (b) if M_i is not made ready for reuse, then the development time incurred is r_i for its development in G_k. For an element M_i, let $y_{ik} = 1$ if M_i is made ready for reuse in G_k, and $y_{ik} = 0$ otherwise. Let $y_i = \sum_{k \in \mathcal{K}} y_{ik}$. Clearly, $y_i \leq 1$. Finally, let $\bar{Y} = (y_1, y_2, \ldots, y_n)$ denote the "reuse vector" of binary reuse decisions y_i, $i \in \mathcal{I}$; that is, $\bar{Y}$ holds the reuse decisions for all elements.

[1] This assumption is relaxed later in §4.1.

Groups are developed in a sequence. Since the software system consists of m groups, there are $\tau = m!$ possible distinct group sequences for the project. Let Σ be the set of all group sequences. Then, we have $\Sigma = \{S_1, S_2, \ldots, S_\tau\}$, where $S_\sigma \in \Sigma$ denotes a particular sequence drawn from the set Σ. Depending on both the sequence of the groups and the reuse decisions, the *release time* (the time a group is completed and released) of a group may be different. Let the release time of G_k be denoted by $T_k, k \in \mathcal{K}$. For groups $G_1, G_2, \ldots, G_m$, let the corresponding completion times, $T_1, T_2, \ldots, T_m$, be represented by a vector, $\bar{T} = (T_1, T_2, \ldots, T_m)$, referred to as a *release-times vector* (RTV). Note that the entries of RTV $\bar{T}$ may not be in ascending order. We are now ready to describe how an RTV, which corresponds to a specific group sequence and a set of reuse decisions, can be generated.

Recall that, given a sequence S_σ, for each M_i, we have $\gamma_i + 1$ different possible reuse decisions: (i) one corresponding to the case where M_i is not made ready for reuse and (ii) γ_i reuse decisions, with each corresponding to the case where M_i is made ready for reuse at one of the γ_i distinct usages of M_i. Each combination of reuse choices for the n elements corresponds to a vector of release times of the groups. Thus, we have $\xi = \prod_{i=1}^{n} (\gamma_i + 1)$ possible RTVs for any sequence S_σ. Since we have $m!$ group sequences, there are a total of $\Lambda = m! \cdot \xi$ possible RTVs for the project. Let $\mathcal{F} = \{1, 2, \ldots, \Lambda\}$ be the index set of RTVs, and let $\bar{T}^f = (T_1^f, T_2^f, \ldots, T_m^f), f \in \mathcal{F}$, be the RTV whose component $T_k^f, k \in \mathcal{K}$, denotes the release time of G_k. We now describe how the developer generates the RTVs for a given group sequence, say, $S_\sigma = (G_1, G_2, \ldots, G_m)$, and a set of reuse decisions (i.e., the values y_{ik}, $i \in \mathcal{I}$, $k \in \mathcal{K}$):

$$t_{ik} = A_{ik} r_i \left\{ \left(1 - \sum_{s<k} y_{is} \right) + \sum_{s<k} y_{is} \beta_{ik} + y_{ik} \alpha_i \right\}, \quad \forall\, i \in \mathcal{I},\ \forall\, k \in \mathcal{K}; \tag{1}$$

$$T_1 = \sum_{i \in \mathcal{I}} t_{i1}; \tag{2}$$

$$T_k = T_{k-1} + \sum_{i \in \mathcal{I}} t_{ik}, \quad \forall\, k \in \mathcal{K},\ k \geq 2; \tag{3}$$

$$y_{ik} \in \{0, 1\}, \quad \forall\, i \in \mathcal{I},\ \forall\, k \in \mathcal{K}.$$

The second and third equations ensure that the groups are completed in the given sequence S_σ. The first equation calculates the time t_{ik} spent on M_i for its usage in G_k and can be explained as follows: Let $A_{ik} = 1$ if M_i belongs to G_k and 0 otherwise. First, the entire expression on the right-hand side is nonzero only if $A_{ik} = 1$. Then, we have three cases:

1. If M_i is made ready for reuse before G_k, then we have the expression $\sum_{s<k} y_{is} = 1$ and $y_{ik} = 0$.

Table 1 Main Notation Used in This Paper

Symbol	Description
V_k	Value from group G_k, $k \in \mathcal{K}$, per unit of time
T_k	Completion time of group G_k, $k \in \mathcal{K}$
Σ	Set of all group sequences, $\Sigma = \{S_1, S_2, \ldots, S_\tau\}$, where $S_\sigma \in \Sigma$ denotes a particular sequence drawn from the set Σ
$\mathcal{F}$	Index set of all possible RTVs, where $\mathcal{F} = \{1, 2, \ldots, \Lambda\}$
X	Developer's monetary value of unit time
B_i	Estimated value of reusable element M_i, $i \in \mathcal{I}$, for future projects
r_i	Development time of element M_i
α_i	The fraction of its processing time needed to make element M_i ready for reuse
β_{ik}	The fraction of its processing time needed to specialize reusable element M_i for group G_k
y_{ik}	1 if element M_i is made ready for reuse in group G_k and 0 otherwise, $i \in \mathcal{I}$, $k \in \mathcal{K}$
y_i	1 if element M_i is made ready for reuse and 0 otherwise, $i \in \mathcal{I}$
y_i^f	1 if element M_i is made ready for reuse when RTV $\bar{T}^f$ is formed and 0 otherwise, $i \in \mathcal{I}$, $f \in \mathcal{F}$
$\bar{Y}$	Reuse vector holding the reuse decisions for all elements
γ_i	Number of distinct usages of M_i in the project, $i \in \mathcal{I}$

Therefore, $t_{ik} = A_{ik} r_i \beta_{ik}$, i.e., only the specialization time is incurred.

2. If M_i is not made ready for reuse before G_k but is made reusable in G_k, then $\sum_{s<k} y_{is} = 0$ and $y_{ik} = 1$. Thus, the time spent on M_i in G_k is $A_{ik} r_i (1 + \alpha_i)$.

3. If M_i has not made ready for reuse before and will not be made ready for reuse in G_k, then $t_{ik} = A_{ik} r_i$, i.e., M_i has to be recreated.

The main notation used in the remainder of this paper is summarized in Table 1.

As mentioned in §1, there are two primary stakeholders involved in the software development process: the user and the developer. Regarding the reuse of each element, there are two critical questions to be answered: (i) Should the element be made ready for reuse? (ii) If yes, when would be the best time to make the element ready for reuse? The second question is relevant because, even if the answer to the first question is positive, investing reuse effort for an element at its first usage may not always be optimal for both the developer and the user. Rather, it may be better to postpone the investment of the reuse effort to one of the element's later usages. Next, we formulate the problems of the developer, the user, and the firm.

2.2. Developer's Problem

As discussed earlier, the developer's objective is to minimize the monetary value of the total time spent on the project. Let T_z^f denote the maximum of the components of RTV $\bar{T}^f$, i.e., $T_z^f = \max_{k \in \mathcal{K}} T_k^f$. In other words, T_z^f is the completion time (makespan) of the project corresponding to RTV $\bar{T}^f$. Let X denote the developer's monetary value of one unit of time (e.g., day). We refer to the developer's problem as Problem D and formally state it as follows:

$$\min_{f \in \mathcal{F}} X T_z^f. \tag{4}$$

Since the developer focuses on minimizing the monetary cost corresponding to the makespan of the project, the reuse decisions corresponding to a developer-optimal RTV naturally serve the purpose of shortening the makespan.

Theorem 1. *If an element is developed as reusable in a developer-optimal RTV, then the element is made ready for reuse at its first usage in the sequence corresponding to this RTV.*[2]

Theorem 1 highlights the natural attitude of the developer toward creating reusable elements. If it is profitable for the developer to make an element ready for reuse, then it is better to do so at its first usage. All future usages following the first usage of that element benefit from this reuse decision. Although the reuse strategy indicated in Theorem 1 is optimal for the developer, such a strategy may not be optimal for the user. Next, we describe the problem of the user (or user department).

2.3. User's Problem

We assume that once a group in this system has been completed (and released), the users can use the functionality provided by this group throughout the system's useful life. Let V_k denote the monetary value per unit of time provided by group G_k, $k \in \mathcal{K}$. The benefit derived from a group is the product of the duration of its availability, i.e., the duration of the time from the moment it is completed (and released) until the moment the system's useful life ends, and the value it provides per unit of time. Then, the aggregate benefit from the system that the user can derive is the sum of such products over all the groups; the objective of the user is to maximize this benefit. Assuming the lifespan of the system is sufficiently large, i.e., at least greater than the maximum possible makespan, and, given a set of RTVs, it is easy to see that the user's objective is equivalent to that of finding the optimal RTV to minimize the sum of the products of the completion time (i.e., release time) of each group and its value per unit of time.

In general, each RTV could result in a different objective function value (cost) for the user. Therefore, from a given set of RTVs, the user selects one that minimizes its cost. By selecting its optimal RTV among the set of all possible RTVs, the user also selects the optimal group sequence in which the project should be completed. For the rest of this paper, we refer to

[2] The proofs of all technical results are provided in the appendix.

the user's RTV selection problem as Problem U. This problem can be formally stated as follows:

$$\min_{f\in\mathcal{F}} \sum_{k\in\mathcal{K}} T_k^f V_k. \tag{5}$$

The developer and the user have their respective costs, i.e., objective function values, for each RTV $\bar{T}^f$, $f \in \mathcal{F}$. By definition, each RTV corresponds to a group sequence and a set of reuse decisions for the elements; i.e., whether or not an element should be made reusable and, if yes, the group for which it should be made reusable. Note that, even though the user is not directly involved in making reuse or group sequencing decisions, by selecting an RTV, the user implicitly signals a preference for a set of reuse decisions as well as a group sequence.

2.4. Firm's Problem

In addition to the total cost incurred by both the user and the developer, the firm's concern (Problem F below) also includes the benefit derivable from reusable elements for future projects. If an element has a reuse value for future projects, we refer to this value as the element's *external reuse value*. Let B_i denote the estimated external reuse value of element M_i, $i \in \mathcal{I}$. Let B^f be the total external value of all elements that are made ready for reuse when RTV $\bar{T}^f$, $f \in \mathcal{F}$, is formed. Let $y_i^f = 1$ if M_i is made ready for reuse for RTV $\bar{T}^f$; $y_i^f = 0$ otherwise. Let $\bar{Y}^f = (y_1^f, y_2^f, \ldots, y_n^f)$ be the reuse decision vector corresponding to RTV $\bar{T}^f$. Then, we have $B^f = \sum_{i\in\mathcal{I}} B_i y_i^f$. The external reuse values can be assumed to be estimated by a separate organizational unit that evaluates software architectures across projects to facilitate reuse. Such departments are common in large organizations (Christensen and Ron 2000).[3]

2.4.1. Problem F. The firm's objective is to choose an RTV that minimizes the sum of weighted completion times (the user's objective) and the total effort incurred during the project (the developer's objective) minus the external reuse values of all the elements:

$$\min_{f\in\mathcal{F}} \left(\sum_{k\in\mathcal{K}} T_k^f V_k + X T_z^f - B^f \right). \tag{6}$$

In the above, the first two components of the firm's objective are, respectively, the costs incurred by the user and the developer. The third component calculates the total external reuse value of the elements. The firm does not know the user's group valuations (V_k, $k \in \mathcal{K}$). However, the developer's pay rate (X) and the external reuse values (B^f, $f \in \mathcal{F}$) are known to the firm.

[3] For example, the Earth Science Data Systems Software Reuse Group at NASA aims to "establish a knowledge sharing community for software reuse in Earth Science and, possibly, to establish a 'marketplace' for reusable software development artifacts" (NASA 2012).

2.5. The Conflict Between the Developer and the User

It is easy to see that the RTVs corresponding to the optimal solutions to Problems D and U can be different. To illustrate this, we provide an example below.

Example 1. Consider three groups G_1, G_2, and G_3, which are to be developed and released in the sequence (G_1, G_2, G_3). Let the values of the groups per unit of time be as follows: $V_1 = 4$ and $V_2 = V_3 = 1$. Assume that G_1 consists of element M_a, G_2 consists of elements M_a and M_b, and G_3 consists of element M_b. Let $r_a = r_b = 1$, $\alpha_a = \alpha_b = 0.5$, $\beta_{a1} = \beta_{a2} = \beta_{b2} = \beta_{b3} = 0.4$, $B_a = 0$, $B_b = 1$, and $X = 5$. We summarize the results in Table 2.

As shown in Table 2, the cost for the user, $\sum_{k\in K} T_k^f V_k$, is minimum when RTV $\bar{T}^1$ is selected. Thus, the user's optimal RTV $\bar{T}^1$ corresponds to the case where no reuse occurs. However, the developer's objective function is minimum when RTV $\bar{T}^2$ is selected. This RTV corresponds to the case where M_a and M_b are made ready for reuse during their usages at G_1 and G_2, respectively.

For the situation in Example 1, the groups will be completed either according to RTV $\bar{T}^2$ in a developer-dominant scenario, or $\bar{T}^1$ in a user-dominant scenario. However, neither of these vectors is optimal for the firm; as shown in Table 2, if the firm had full information, then the firm-optimal solution would be RTV $\bar{T}^3$. The main takeaway from this example is that since the optimal RTVs for the problems of the three parties can all be different, there may be no common solution (or RTV) that they can agree on. Furthermore, reuse is not always a good thing for the overall goal of the firm. This is clear when we observe that the developer's optimal choice is to make both M_a and M_b reusable. However, the firm would not choose to make M_a reusable because doing so would significantly hurt the user. This motivates the need for a common solution for the developer and the user. Of further interest is the issue of whether or not this common solution is optimal for the firm.

Table 2 Computations in Example 1

$f\downarrow$	$\bar{Y}^f$ (y_a^f, y_b^f) (reuse vector)	$\bar{T}^f$ (RTV)	$\sum_{k\in\mathcal{K}} T_k^f V_k$ (user's objective)	$T_3^f X$ (developer's objective)	$\sum_{k\in\mathcal{K}} T_k^f V_k + T_3^f X - B^f$ (firm's objective)
1	(0, 0)	(1, 3, 4)	**11**	$4X = 20$	31
2	(1, 1)	(1.5, 3.4, 3.8)	13.2	**$3.8X = 19$**	31.2
3	(0, 1)	(1, 3.5, 3.9)	11.4	$3.9X = 19.5$	**29.9**

Note. The boldfaced numbers in the fourth, fifth, and sixth columns (i.e., the numbers 11, 19, and 29.9) correspond to the respective optimal objective values for the user, the developer, and the firm.

3. Optimal Transfer Pricing

As discussed above, because of differences in their individual objective functions, the RTVs corresponding to the optimal solutions of the developer and the user may be different, leaving both parties without a common solution. Even worse, the individual solutions of the two sides for their respective problems may be different than the firm-optimal solution.

In this section, we propose TP schemes through which both the user and the developer arrive at a firm-optimal solution. The firm is assumed to know the developer's pay-rate (X), but does not know a user's group valuations. We consider four possibilities: SU-SD, MU-SD, SU-MD, and MU-MD. For each TP scheme, we discuss the following three questions:

1. Will the TP scheme result in a firm-optimal solution?
2. Will the developer(s) be better off as a result of implementing the TP scheme?
3. Will the user(s) be better off as a result of implementing the TP scheme?

Our main result is that *each TP scheme results in optimal transfer prices, even though the firm may not know the user group valuations*. The term "optimal transfer prices" refers to those transfer prices that lead to decisions from the user(s) and the developer(s), such that the overall solution is optimal for the firm. The transfer prices computed under these schemes are incentive compatible to both the user and the developer, i.e., the concerned parties are weakly better off in the presence of the proposed schemes than without them. In general, the firm-optimal solution (i.e., the solution vector) is not unique and neither are the optimal transfer prices. The primary focus of our transfer pricing schemes is to find *one vector* of transfer prices that guarantees a firm-optimal solution and satisfies incentive compatibility for both parties. As we will see later, the answers to the second and third questions above depend on the particular TP scheme being considered.

3.1. Single User and Single Developer

In this case, we assume a single developer who develops elements one at a time. Note that the development sequence of the elements within a group does not matter, since all the elements in a group must be developed for the group to be completed.

3.1.1. Description of SU-SD. Under the SU-SD scheme, the developer reports all possible RTVs (over all reuse decisions and all group sequences) and the corresponding reuse vectors to the firm. The firm conveys to the user these RTVs together with a transfer price (p_f) for each RTV ($\bar{T}^f, f \in \mathcal{F}$). Next, the user selects the RTV that minimizes the time-weighted group valuations plus the price, i.e., $\min_{f\in\mathcal{F}}(\sum T_k^f V_k + p_f)$, and pays the firm the price for the selected RTV. Note that the total cost the user incurs for an RTV is separable, in the sense that it is the sum of the user's time-weighted valuation for that RTV and the corresponding transfer price. Finally, the firm pays an amount that makes the developer indifferent between the selected RTV and the developer-optimal RTV. A formal description of the procedure is provided in the appendix.

Sequence of Events in SU-SD.

1. The developer provides the RTVs and their corresponding reuse vectors to the firm.
2. The firm identifies the developer-optimal RTV and, based on this RTV, computes a transfer price for each RTV.
3. The firm shares with the user the RTVs along with their associated prices.
4. To select an optimal RTV, the user minimizes its cost (the user's objective function value plus the transfer price) over the RTVs.
5. The user pays the firm (if needed) and the firm pays the developer (if needed).

3.1.2. Computing the Transfer Prices. The transfer price for a given RTV reflects two kinds of externalities: (a) the difference between the total external reuse values of the elements that are not made ready for reuse for that RTV and for the developer-optimal RTV, and (b) the loss the developer incurs if that RTV does not correspond to a developer-optimal solution. For each RTV, say, $\bar{T}^f$, the firm computes the quantity $T_z^f X - B^f$ (referred to hereafter as the RTV-cost), where $T_z^f X$ is the cost the developer incurs, and B^f is the total external value of all the elements that are made ready for reuse in this RTV. The firm then identifies the developer-optimal RTV, i.e., the RTV that minimizes the developer's cost across all RTVs ($\min_{f\in\mathcal{F}} T_z^f X$). Next, the price for the RTV $\bar{T}^f$, i.e., p_f, is computed by taking the difference between the RTV-cost for RTV $\bar{T}^f$ and the RTV-cost for the developer-optimal RTV.

Below, we illustrate SU-SD with an example:

Example 2. In this example, we specialize the data in Example 1 by having identical external reuse values for the elements M_a and M_b, i.e., $B_a = B_b = 1$. Table 3 summarizes *who knows what* in this example. Table 4 lists the objective function values of the user, the developer, and the firm.

Table 3 Information Possessed by Each Party

	$\bar{T}^f, f \in \mathcal{F}$ (RTVs)	$B_i, i \in \mathcal{I}$ (external reuse value)	X (developer's monetary value of unit time)	$V_k, k \in \mathcal{K}$ (group valuations)
The firm	✓	✓	✓	
The developer	✓		✓	
The user	✓			✓

Table 4 Computations in Example 2 for Elements M_a and M_b

$f \downarrow$	$\bar{Y}^f$ (y_a^f, y_b^f) (reuse vector)	$\bar{T}^f$ (RTV)	$T_3^f X$ (developer's objective)	$T_3^f X - B^f$ (RTV-cost)	p_f (TP)	$\sum_{k\in\mathcal{K}} T_k^f V_k$ (user's objective)	$\sum_{k\in\mathcal{K}} T_k^f V_k + p_f$ (user's total cost)	$\sum_{k\in\mathcal{K}} T_k^f V_k +$ $T_3^f X - B^f$ (firm's objective)
1	(0, 0)	(1, 3, 4)	20	20	3	**11**	14	31
2	(1, 1)	(1.5, 3.4, 3.8)	**19**	17	0	13.2	13.2	30.2
3	(0, 1)	(1, 3.5, 3.9)	19.5	18.5	1.5	11.4	**12.9**	**29.9**

Notes. The boldfaced numbers in the fourth, seventh, and ninth columns (i.e., the numbers 19, 11, and 29.9) correspond to the respective optimal objective values, without the transfer pricing scheme, for the developer, the user, and the firm. The boldfaced number in the eighth column (i.e., the number 12.9) corresponds to the user's optimal cost under the transfer pricing scheme.

Since the number of usages for M_a and M_b are $\gamma_a = 2$ and $\gamma_b = 2$, respectively, the total number of possible combinations of reuse decisions for the two elements is $\xi = (\gamma_a + 1)(\gamma_b + 1) = 9$. Given that there are three groups, the total number of possible group sequences is $\tau = 6$. Thus, there are $\Lambda = 6 \times 9 = 54$ possible RTVs. For brevity, we present only three RTVs corresponding to the sequence $S_1 = (G_1, G_2, G_3)$. Without considering the transfer prices, the user would select RTV $\bar{T}^1$ and not make either of the elements reusable. The developer's optimal solution is RTV $\bar{T}^2$, which corresponds to making both the elements reusable. If the firm knew the group valuations, then the firm's optimal solution would be RTV $\bar{T}^3$, which corresponds to making only M_b reusable. Under SU-SD, the firm first identifies the developer-optimal RTV, which is RTV $\bar{T}^2$. Then, to compute the transfer prices, the firm takes the difference between the RTV-cost for any RTV and the RTV-cost for the developer-optimal RTV (see §3.1.2 for the RTV-cost formula). The transfer prices are shown in the sixth column of Table 4. Notice that in the case of SU-SD, the firm only uses X and RTVs for determining the transfer prices. Given the transfer prices, the user selects the RTV $\bar{T}^3$, and pays the firm 1.5. Finally, the developer receives 0.5 from the firm and is indifferent between the RTVs $\bar{T}^3$ and $\bar{T}^2$. In this example, the user is better off as a result of implementing SU-SD. Without SU-SD, it is natural to expect that the developer's optimal solution would be implemented. In this case, the user incurs a cost of 13.2, which is strictly greater than the cost incurred under SU-SD (12.9). Note that the transfer price in this example (1.5) is positive; i.e., the mechanism is not budget balanced. Also, the total cost the user incurs for the firm-optimal RTV is the sum of the user's valuation of that RTV and the corresponding transfer price, i.e., the total cost the user incurs is $11.4 + 1.5 = 12.9$.

3.1.3. Properties of SU-SD. We now revisit the three questions posed earlier concerning the properties of a transfer pricing scheme. The following theorems address these questions. We provide an intuitive understanding of each result in this paper; the formal proofs are provided in the appendix.

Theorem 2. *The procedure* SU-SD *yields optimal transfer prices*.

An intuitive understanding of the above theorem is as follows. The firm's objective consists of the sum of three components: a user component, a developer component, and an external reuse value component. Of these components, the firm does not know the user component because it does not know the user group valuations. However, since the user considers these valuations while selecting a preferred RTV, the transfer price associated with an RTV does not need to reflect the user component. The other two components, namely, the developer component and the external reuse value component, must be reflected in the transfer price in such a way that the firm-optimal RTV is selected by the user. Thus, under SU-SD, the transfer price includes the externalities imposed on the developer's cost and on the external reuse value. Note that, under SU-SD, the firm does not need to elicit the time-weighted group valuations of the user.

Theorem 3. *The user is weakly better off under* SU-SD *than under a developer-optimal solution*.

Under SU-SD, the transfer price for the developer-optimal solution is set to zero. Furthermore, this solution is also feasible for the user. Thus, the total cost the user incurs (the user's own objective function value plus the transfer price) for the solution under SU-SD is at most that incurred for the developer-optimal solution. Note that our version of SU-SD uses the developer-optimal RTV as a benchmark to set the transfer prices for the user. Instead, any arbitrary RTV can be used as a benchmark to set the transfer prices. In this sense, the TP scheme under SU-SD also bears a resemblance with Nash bargaining. The transfer prices under SU-SD amount to a Nash bargaining solution in which the developer is always guaranteed to obtain its optimal solution, i.e., the developer's cost is minimized.

Remark 1. The total cost (objective function value minus any payment received) incurred by the developer for the solution chosen under SU-SD is equal to that for the optimal solution to Problem D. In SU-SD,

the externality imposed on the developer by the user is collected from the user as part of the transfer price. This externality is paid to the developer as a compensation for any loss incurred as compared to the developer-optimal solution. Thus, the developer is indifferent between the user's preferred RTV and the developer-optimal RTV. As shown in Theorem 1, in the absence of any compensation, if the developer benefits from making an element reusable, then the element would be made reusable in its first usage. However, it is possible for the firm or the user to prefer a different reuse decision for that element: either make the element reusable in a later usage or not make it reusable at all. Thus, on certain occasions, a developer-suboptimal solution may be implemented but the developer would be appropriately compensated for the loss.

Remark 2. Theorem 3 and Remark 1 imply voluntary participation by the user and the developer. Thus, SU-SD is sustainable.

Remark 3. Under SU-SD, the firm does not need to elicit group valuations from the user. Given the RTV prices, the user identifies its optimal RTV, which is also a firm-optimal RTV.

Next we examine the scenario where there are multiple users, with each interested in a subset of groups. We provide a TP scheme where users naturally report their true costs to the firm, and the firm picks an RTV that results in optimal transfer prices.

3.2. Multiple Users and Single Developer

Consider the situation when there is a single developer but multiple users, e.g., accounts payable, purchasing, inventory, with each benefiting from a subset of the groups. For example, the user-department "inventory" requires only the groups (functionalities) related to tracking inventory accurately, managing orders and purchases, determining reorder points, etc. Thus, each user values a subset of the groups. Note that the firm does not know the group valuations of the users. Although our proposed TP scheme remains valid when multiple users have positive valuations for the same group, for simplicity of exposition we will assume that the subsets of groups corresponding to the different users are mutually exclusive. For example, suppose we have four groups, G_1, G_2, G_3, and G_4, and two users. Then, a possible scenario is that the first user only requires groups G_1 and G_4, whereas the second requires only G_2 and G_3. Since the users are interested in different subsets of groups, they naturally may have different preferred RTVs. This, in turn, adds an additional complexity (relative to the case where we had a single user) of finding an RTV that all users, the developer, and the firm agree on. In the case of multiple users, in addition to the conflict between the developer and users, there might also be conflicts among the users themselves. This latter type of conflict is the main difference between a setting that considers multiple users and that for a single user. Under the MU-SD scheme below, the firm uses transfer prices to identify a firm-optimal RTV, i.e., firm-optimal reuse and group sequencing decisions.

3.2.1. Description of MU-SD. Under MU-SD, identify its optimal RTV, the firm designs an auction for the users based on the well-known Clarke (1971) mechanism.[4] The Clarke mechanism (also referred to as the Pivotal mechanism) is an individual-rational, efficient, and strategy-proof mechanism that can be used to determine taxpayers' true valuations of public projects. We use this mechanism in the context of our problem.[5] The key point in designing an RTV auction for the users is that the firm computes a transfer price for each RTV in such a way that each user truthfully reveals the sum of its time-weighted group valuations. Before providing a detailed description of how the firm determines the transfer prices under MU-SD, we summarize below the sequence of events in this TP scheme. A formal description of MU-SD is provided in the appendix.

Sequence of Events in MU-SD.

1. The firm announces how the transfer prices will be determined (to be discussed shortly in §3.2.2). Note that the firm only announces the pricing scheme, not the prices themselves. The prices are calculated using the Clarke mechanism, where each user is asked to pay for the externality imposed on the other users, as well as on the developer and the firm.
2. The developer provides the RTVs and their corresponding reuse vectors to the firm. The firm shares these RTVs with the users.
3. For each RTV, each user reveals its bid, which is the sum of the time-weighted group valuations (in Theorem 4, we show that it is a dominant strategy for the users to reveal their bids truthfully).
4. Based on the bids of the users, the firm picks the firm-optimal RTV. For this RTV, the firm computes and then reveals user-specific transfer prices to each user.
5. Users pay the firm (if needed) and the firm pays the developer (if needed).

Next, we explain how the transfer prices are determined.

3.2.2. Computing the Transfer Prices. At the end of MU-SD, each user pays the firm a transfer price,

[4] The Clarke mechanism is a special case, where budget balance is not required, of the more general VCG (Vickrey–Clarke–Groves) mechanism. The intuition behind the mechanism is provided in §B.5.

[5] A more detailed explanation of the application of the mechanism to our problem is briefly explained in §B.6.

which equals the sum of the externalities each user imposes on the other users, the developer, and the firm. Let U_η be the user of interest. Given the bids of the users, the firm first identifies the firm-optimal RTV by solving Problem F (see §2.4). Let the firm-optimal RTV be denoted by $\bar{T}^w$. Next, to determine the total externality U_η imposes on the other parties, the firm solves Problem F without considering U_η's bids, i.e., the firm determines the firm-optimal RTV in the absence of U_η. Let this RTV be denoted by $\bar{T}^{\varpi}$. If these two RTVs are the same, then U_η does not impose an externality on any of the parties; otherwise, it is clear that U_η influences the firm's decision to choose RTV $\bar{T}^w$ (instead of $\bar{T}^{\varpi}$). This influence on the firm's choice leads to three externalities:

(i) The objective function value of the developer may be different for the two RTVs $\bar{T}^w$ and $\bar{T}^{\varpi}$. We refer to this difference as the *developer externality*.

(ii) The sum of the external reuse values of the elements that are made ready for reuse in $\bar{T}^w$ may be different from that in $\bar{T}^{\varpi}$. We refer to this externality as the *reuse externality*.

(iii) Because the release times of the groups in $\bar{T}^w$ may be different from those in $\bar{T}^{\varpi}$, users may incur different costs for these RTVs. We refer to the total change in the sum of the costs of the other users (i.e., users other than U_η) as the *user externality*.

Note that each of the externalities above can be negative, zero, or positive. Example 3 below illustrates how the transfer price, which is the sum of the three externalities, is computed for each user.

Example 3. In this example, we consider the same data as in Example 1. Suppose groups G_1, G_2, and G_3 are required by users U_1, U_2, and U_3, respectively. Thus, $\mathcal{K} = \{1, 2, 3\}$. Let $\mathcal{K}_\eta \subseteq \mathcal{K}$ denote the set of the groups that U_η utilizes, and let $\mathcal{K}_1 = \{1\}$, $\mathcal{K}_2 = \{2\}$, and $\mathcal{K}_3 = \{3\}$. Let $b_{\eta f}$ denote the bid U_η makes for RTV $\bar{T}^f$. For brevity, we only consider four RTVs corresponding to the sequence (G_1, G_2, G_3).

As shown in Table 5, $\bar{T}^1$ is the optimal RTV for U_1, $\bar{T}^2$ for both U_3 and the developer, $\bar{T}^3$ for both U_1 and the firm, and $\bar{T}^4$ for U_2. Note that the group valuations are not known to the firm. In the following, we explain how the firm-optimal RTV is obtained through the use of the transfer prices:

1. The firm announces how the transfer prices are going to be computed, as discussed in §3.2.2.

2. Following this announcement, the users bid for each RTV, $b_{\eta f}$, $\eta = 1, 2, 3$; $f = 1, 2, 3, 4$.

3. The firm computes the transfer prices by considering the externality each user imposes on other parties. We now explain how the externalities that U_1 imposes on the other parties are computed. The firm solves Problem F by considering the bids of all three users and identifies RTV $\bar{T}^3$ as the firm-optimal solution. If U_1 were not to participate in the auction, i.e., if the firm solved Problem F with U_1's bids being excluded, then the firm-optimal solution would be RTV $\bar{T}^2$. Thus, U_1's presence in the auction influences the firm's optimal solution. As a result of the firm choosing RTV $\bar{T}^3$ instead of $\bar{T}^2$, the developer's cost increases by $19.5 - 19 = 0.5$. Thus, the developer externality equals 0.5. The total external reuse value of the elements that are made ready for reuse is the same for $\bar{T}^2$ and $\bar{T}^3$. Thus, there is no reuse externality imposed by U_1. However, the sum of the bids of the other two users (U_2 and U_3) for RTV $\bar{T}^3$ is 7.4, whereas this sum is 7.2 for $\bar{T}^2$. Thus, U_1 causes a user externality of $7.4 - 7.2 = 0.2$. Thus, the transfer price for U_1 is $0.5 + 0 + 0.2 = 0.7$. Similarly, the firm computes the transfer prices for the other two users. The other users (U_2 and U_3) do not impose any externality on any of the parties.

4. At the end of MU-SD, the firm chooses RTV $\bar{T}^3$ as its solution, and U_1 pays its transfer price to the firm. Since the firm's choice is not developer optimal, the developer incurs a cost of $19.5 - 19 = 0.5$. The firm compensates the developer for this loss.

As a property of the Clarke mechanism, the bids in Step 2 above reflect the true time-weighted group valuations of the users; that is, $b_{\eta f} = \sum_{k \in \mathcal{K}_\eta} T_k^f V_k$. Note that the sum of the transfer prices in this example is positive, i.e., $0.7 + 0 + 0 > 0$, implying that the mechanism is not necessarily budget balanced. Also, the total cost each user incurs for the firm-optimal RTV is the sum of that user's valuation for that RTV and

Table 5 **Computations in Example 3 for Users U_1, U_2, U_3, the Developer, and the Firm**

$f \downarrow$	$\bar{Y}^f$ (y_a^f, y_b^f)	$\bar{T}^f$ (RTV)	b_{1f} (U_1's bid)	b_{2f} (U_2's bid)	b_{3f} (U_3's bid)	$T_3^f X$ (developer's objective)	$\sum_{\eta \in \mathcal{K}} b_{\eta f} + X T_z^f - B^f$ (firm's objective)
1	(0, 0)	(1, 3, 4)	**4**	3	4	$4X = 20$	31
2	(1, 1)	(1.5, 3.4, 3.8)	6	3.4	**3.8**	$\mathbf{3.8X = 19}$	31.2
3	(0, 1)	(1, 3.5, 3.9)	**4**	3.5	3.9	$3.9X = 19.5$	**29.9**
4	(1, 0)	(1.5, 2.9, 3.9)	6	**2.9**	3.9	$3.9X = 19.5$	32.3

Notes. The boldfaced numbers in the fourth, fifth, and sixth columns (i.e., the numbers 4, 2.9, and 3.8) correspond to the respective minimum bids of U_1, U_2, and U_3 (i.e., their optimum solutions to Problem U). The boldfaced number in the seventh and eighth columns (i.e., the numbers 19 and 29.9) correspond to the respective optimal objective values of the developer and the firm. The transfer prices for users U_1, U_2, and U_3 are 0.7, 0, and 0, respectively (not shown).

the corresponding transfer price, i.e., the total cost of U_1 is $4 + 0.7 = 4.7$, whereas those of U_2 and U_3 are $3.5 + 0 = 3.5$ and $3.9 + 0 = 3.9$, respectively.

To summarize, under MU-SD, given the pricing mechanism introduced above, users reveal their true costs for each RTV. At the end of this TP scheme, each user pays the firm a transfer price. Thus, the total cost for each user is the sum of the user's own objective function value and the transfer price. If the firm chooses a developer-suboptimal solution, the firm compensates the developer for the loss.

3.2.3. Properties of MU-SD. The following three results and the subsequent remark highlight the properties of this scheme.

Theorem 4. *Under* MU-SD, *it is a dominant strategy for users to reveal their true costs, i.e., the sum of their time-weighted group valuations.*

We now provide an intuitive explanation of this result. Note that any RTV that is chosen by the firm is common to the users, the developer, and the firm. Implicitly, by reporting its valuations for the RTVs, each user influences the firm's choice. For example, consider two RTVs $\bar{T}^a$ and $\bar{T}^b$. Suppose user U_1 (say) incurs a higher cost for $\bar{T}^a$ than $\bar{T}^b$, whereas this RTV is a cost-minimizing one for all the other users and is also firm-optimal. Let $\bar{T}^b$ be a cost-minimizing RTV for U_1. To induce the firm to choose $\bar{T}^b$, suppose U_1 considers misrepresentation by reporting a substantially high (respectively, low) cost for $\bar{T}^a$ (respectively, $\bar{T}^b$). Clearly, if the firm chooses $\bar{T}^b$, the users other than U_1 will incur a higher cost relative to that for $\bar{T}^a$. The externality U_1 imposes on the other parties is the total increase in their cost due to the firm selecting $\bar{T}^b$ instead of $\bar{T}^a$. By paying this externality to the firm, U_1 would incur a higher total cost (externality plus its true cost) than that incurred under truthful revelation. Since users know that they will be charged based on how much damage their reported valuations cause to the others, they reveal their true costs; i.e., given the RTVs, users truthfully reveal the sum of their time-weighted group valuations to the firm.

Theorem 5. *The procedure* MU-SD *yields optimal transfer prices.*

Under MU-SD, the firm has enough information to solve Problem F. This follows from the fact that the bids from the users reflect their truthful valuations, which, in turn, is an implication of the pricing scheme set by the firm. Thus, the firm achieves optimal transfer prices even under information asymmetry. Observe that MU-SD is more centralized relative to SU-SD, in the sense that here the firm itself chooses an RTV at the end of the process.

Theorem 6. *Any user is weakly better off under* MU-SD *than its worst solution without any transfer pricing intervention.*

Formally, under MU-SD, the total cost a user incurs (user's objective function value plus the transfer price) is at most its maximum possible objective function value (i.e., the maximum value of the sum of its time-weighted group valuations). An intuitive explanation for the above result is as follows. Let the RTV corresponding to the maximum objective function value a user, say, user U_1, incurs in the absence of MU-SD be $\bar{T}^*$. Let the firm-optimal RTVs at the end of MU-SD with U_1 and without U_1 be $\bar{T}^1$ and $\bar{T}^2$, respectively. If $\bar{T}^* = \bar{T}^1$, then the firm-optimal solution does not change whether or not the firm considers the bids of U_1, i.e., $\bar{T}^1 = \bar{T}^2$. That is, U_1 does not exert any externality on the others, so the transfer price for U_1 is equal to 0. If $\bar{T}^* \neq \bar{T}^1$ and $\bar{T}^1 = \bar{T}^2$, then, again, U_1's presence does not influence the firm's decision. Therefore, U_1 does not pose any externality, and its transfer price is again 0. Also, since U_1's objective function value at $\bar{T}^1$ is less than that at $\bar{T}^*$, U_1 is better off under MU-SD. If $\bar{T}^* \neq \bar{T}^1$ and $\bar{T}^1 \neq \bar{T}^2$, then the absolute value of the difference between U_1's objective function values at RTVs $\bar{T}^1$ and $\bar{T}^2$ is larger than or equal to the difference of the sums of the objective function values of other users at these two RTVs. Otherwise, the firm would choose $\bar{T}^2$ as the solution even in the presence of U_1. Thus, U_1 pays a transfer price that is less than or equal to the gain U_1 achieves from the firm choosing $\bar{T}^1$ as the solution instead of $\bar{T}^2$. Therefore, U_1 is weakly better off under MU-SD as compared to the scenario where the firm implements $\bar{T}^*$ as the solution without MU-SD.

Remark 4. The total cost (objective function value minus any payment) incurred by the developer for the solution chosen under MU-SD is equal to that for the optimal solution to Problem D. As in SU-SD, the firm compensates the developer for its loss if MU-SD results in a developer-suboptimal solution.

3.3. Multiple Developers

In contrast to the preceding sections, we now assume that we have multiple developers with different pay rates that are known to the firm. We assume that a single developer creates an element for its usage in a particular group. However, across the different usages of an element, the developers are allowed to be different. We further relax the assumption that the elements are developed sequentially and assume that elements can be developed in parallel within a group, but the groups are developed sequentially and are immediately released once all the elements required by each group are completed. We refer to our TP scheme for multiple developers as MD. In this scheme, we avoid

specifying whether we have one or many users; as will become clear shortly, to determine the transfer prices for the user side, the firm follows the same steps introduced for either SU-SD (§3.1) if there is single user, or MU-SD (§3.2) if there are multiple users.

Let $\mathscr{D} = \{D_1, D_2, \ldots, D_\phi\}$ and $\Phi = \{1, 2, \ldots, \phi\}$ be the set of developers and the corresponding index set, respectively. Let X_θ denote the payment per unit effort for D_θ, $\theta \in \Phi$. The firm assigns usages of the elements to the developers. As before, the objective of each developer is to minimize its effort for the project. In the absence of any TP scheme, each developer solves its individual effort minimization problem and develops its assigned usages based on this solution. The groups are released as and when they are completed; that is, whenever all the usages needed for a group are completed, the corresponding group is released. As shown in Example 4 below, in this situation as well, the reuse decisions made by the developers may not be firm optimal.

Example 4. Consider three groups G_1, G_2, and G_3, which are to be developed and released in the sequence (G_1, G_2, G_3). Assume that G_1 consists of element M_a, G_2 consists of elements M_a and M_b, and G_3 consists of M_b. Let $r_a = r_b = 1$, $\alpha_a = \alpha_b = 0.5$, and $\beta_{a1} = \beta_{a2} = \beta_{b2} = \beta_{b3} = 0.4$. Consider two developers D_1 and D_2. Developer D_1 is assigned to the usages of M_a and M_b for G_1 and G_3, respectively, and D_2 is assigned to the usages of M_a and M_b for G_2. If the developers solve their own local problems, then neither of the two makes the two elements reusable. However, from the firm's perspective, the optimal solution is to make both elements reusable.

3.3.1. Description of MD. Under MD, the developers first generate the RTVs corresponding to different group sequencing and reuse decisions. Similar to SU-SD and MU-SD, the firm is provided these RTVs and their corresponding reuse vectors. In addition, the firm is also provided with the effort each developer incurs for each RTV. The firm next turns to the user side. If there is a single user, as discussed in §3.1, (a) the firm computes a price (p_f) for each RTV ($\bar{T}^f$, $f \in \mathscr{F}$), (b) the firm shares the RTVs and their corresponding prices with the user, and (c) the user picks an RTV that minimizes its time-weighted group valuations plus the price ($\min_{f\in\mathscr{F}}(\sum T_k^f V_k + p_f)$) and pays the firm the price for the selected RTV. On the other hand, if there are multiple users, then the firm auctions the RTVs following the same pricing scheme discussed for MU-SD in §3.2. After the auction, the firm obtains the transfer prices for the users. At the end of MD, if a developer-suboptimal RTV is chosen, the firm pays each developer a fraction—determined as the ratio of the developer's effort to the sum of the efforts of all developers—of the price for that RTV. A formal description of MD is provided in the appendix.

Sequence of Events in MD.

1. The developers provide the RTVs and the corresponding reuse vectors. In addition, each developer reports the effort associated with each RTV (explained in the next subsection).

2. In the case of a single user, the firm identifies the global developer-optimal RTV (explained in the next subsection). Based on this RTV, the firm computes a transfer price for each RTV. The remainder of the scheme follows the same steps as in SU-SD.

3. In the case of multiple users, the firm auctions the RTVs among the users. The remainder of the scheme follows the same steps as in MU-SD.

4. The user(s) pays the firm (if needed) and the firm pays the developers (if needed), based on each developer's individual total effort for the project.

3.3.2. Computing the Transfer Prices. As in our previous schemes, MD needs to consider developer externality. The main difference, however, is that the developer externality is computed with respect to the global developer-optimal solution rather than the solution of any individual developer. Let $\bar{E}^f = (E_1^f, E_2^f, \ldots, E_\phi^f)$ be the "effort vector," whose component E_θ^f, $\theta \in \Phi$, denotes the effort of developer D_θ corresponding to RTV $\bar{T}^f$, $f \in \mathscr{F}$. Thus, the objective function value of developer D_θ corresponding to RTV $\bar{T}^f$ is $E_\theta^f X_\theta$. Given the RTVs and the effort vector, the firm determines an RTV that minimizes the sum of the costs of the developers (i.e., $\sum_{\theta\in\Phi} E_\theta^f X_\theta$). We refer to this RTV as the global developer-optimal RTV. Next, depending on the number of users (single or multiple), the firm computes the transfer prices as below.

Single User. For an RTV, say, $\bar{T}^f$, the firm computes an "RTV-cost" $\sum_{\theta\in\Phi} E_\theta^f X_\theta - B^f$, where $\sum_{\theta\in\Phi} E_\theta^f X_\theta$ is the sum of the costs incurred by the developers, and B^f is the total external reuse value of all the elements that are made ready for reuse in this RTV. Then, the firm computes the price (p_f) associated with an RTV $\bar{T}^f$ by taking the difference between the RTV-cost for $\bar{T}^f$ and the RTV-cost for the global developer-optimal RTV. Clearly, this price equals 0 for the global developer-optimal RTV.

Multiple Users. At the end of an RTV auction that is executed by the firm, each user pays the externality it imposes on the firm, the developers, and other users. Let the user of interest be U_η. Suppose the firm-optimal RTV is $\bar{T}^w$ (respectively, $\bar{T}^{\varpi}$) when the firm considers (respectively, does not consider) the bids of U_η. The computation of the reuse externality and the user externality is similar to that under MU-SD. However, when computing the developer externality, the firm takes the difference of the sum of the developers' objective function values at RTVs $\bar{T}^w$ and $\bar{T}^{\varpi}$, i.e.,

$\sum_{\theta\in\Phi} E_\theta^w X_\theta - \sum_{\theta\in\Phi} E_\theta^{\varpi} X_\theta$. Finally, as a transfer price, U_η pays the total externality it imposes on the others, i.e., U_η pays the sum of the developer externality, the reuse externality, and the user externality.

The following result states a key property of MD. Because the proof is similar to that for Theorems 2 and 5, for brevity, we omit it.

Theorem 7. *The TP scheme* MD *results in optimal transfer prices*.

Given that each developer's effort for an RTV is known to the firm, the procedure to determine a firm-optimal solution under a single-user, multiple-developer scenario (respectively, a multiple-user, multiple-developer scenario) is similar to that under SU-SD (respectively, MU-SD). The additional step in a multiple-developer situation (as compared to the single-developer setting) is that the firm determines a developer's compensation individually. Note that, as with our previous TP schemes, the firm is not required to know the group valuations of the users to achieve a firm-optimal solution under MD.

Remark 5. The introduction of multiple developers has no particular bearing on the question of whether the user(s) are better off. Under MD with a single user, Theorem 3 still holds, i.e., the user is weakly better off as compared to a developer-optimal solution in the absence of MD. Similarly, under MD with multiple users, Theorem 6 still holds.

Remark 6. Under MD, a developer is not guaranteed to be better off as compared to its own optimal solution. However, it is easy to show that no developer is worse off (after receiving compensation) under MD than its worst solution (without transfer prices). Furthermore, our assumption under MD is that the effort incurred by each developer for an RTV is reported to the firm. Although we have assumed such reporting occurs naturally, the firm can execute an RTV auction for the developers—similar to the one for the users under MU-SD—to have the developers reveal their efforts truthfully.

4. Discussions

In this section, we confirm the validity of our TP schemes for two practical extensions: (a) when an element has multiple usages within a particular group and (b) when an element is partially reusable; that is, some part of an element is suitable for reuse while the remainder of the element is not. We end by a short discussion on complexity issues.

4.1. Multiple Usages of an Element for the Same Group

Contrary to the assumption that each element has one usage per group, we consider the case where an element may have multiple usages in a single group. Let element M_j, $j \in \mathcal{J}$, have multiple usages for group G_h, $h \in \mathcal{K}$. We now revisit our TP schemes to investigate whether the firm can still obtain optimal transfer prices in the presence of M_j. Recall that the TP schemes introduced in this study begin with generating the RTVs. Each RTV corresponds to a possible reuse decision for each element; that is, a reuse decision on an element may change the release times of the groups. The developer can generate RTVs corresponding to alternative reuse decisions for M_j; however, if M_j is to be made reusable at one of its usages for G_h, then it is optimal for the developer, the user(s), and the firm to have M_j made ready for reuse at its first usage for G_h. Recall that the reuse decision variable $y_{ik} = 1$ if M_i is made ready for reuse at its usage for G_k. Given a group sequence and its corresponding reuse decisions, if, for each element, the number of usages per group is known to the developer, then the RTVs can be determined. Let ρ_{ik} denote M_i's number of usages for G_k, where $i \in \mathcal{J}$ and $k \in \mathcal{K}$. The total effort corresponding to all usages of M_i for G_k is denoted by t_{ik} and can be expressed as follows:

$$t_{ik} = A_{ik} r_i \left\{ \rho_{ik} \left(\left(1 - \sum_{s<k} y_{is} \right) + \sum_{s<k} y_{is} \beta_{ik} \right) + y_{ik} \left(\alpha_i + (1-\rho_{ik})(1-\beta_{ik}) \right) \right\}, \quad \forall\, i \in \mathcal{J},\ \forall\, k \in \mathcal{K}.$$

Recall that $A_{ik} = 1$ if M_i belongs to G_k. We have three cases for the expression above:

(i) If M_i is made ready for reuse before G_k, then we have the expression $\sum_{s<k} y_{is} = 1$ and $y_{ik} = 0$. Therefore, $t_{ik} = A_{ik} r_i \rho_{ik} \beta_{ik}$, i.e., only the specialization effort for each usage is incurred.

(ii) If M_i is not made ready for reuse before G_k, but is made reusable in G_k, then $\sum_{s<k} y_{is} = 0$ and $y_{ik} = 1$. Thus, the total effort for M_i in G_k is $A_{ik} r_i((1+\alpha_i) + (\rho_{ik} - 1)\beta_{ik})$.

(iii) If M_i is not made ready for reuse before and will not be made ready for reuse in G_k, then $t_{ik} = A_{ik} r_i \rho_{ik}$, i.e., the developer has to recreate M_i at its every usage for G_k.

After computing t_{ik} for each M_i, as it was under SU-SD, the developer can determine the release times of the groups by using Equations (2) and (3).

Under a multiple-developer scenario where M_j has multiple usages for G_h, it is critical to know the sequence in which the elements are created. Consider again that M_j is to be made ready for reuse at one of its usages for G_h. Since the developers can manage parallel development in the same group, clearly one cannot claim that it is optimal to make M_j in its first usage for G_h. We further explain this with the following simple example. Suppose that M_j in G_h has three

usages, j_1, j_2, and j_3, each developed by a different developer. Suppose j_1 and j_2 are to be developed in parallel by two of the developers, and subsequently j_3 is to be developed by the last developer. If M_j is to be made ready for reuse in G_h, then, neither of the first two developers benefit from this reuse decision. To generate the release time of G_h, it is essential to know the sequence in which M_j is developed for its usages at G_h. Since both the group and element sequences as well as the reuse decisions are known to the firm, as it was under MD, the release times of the groups can be computed. This explains the generation of RTVs under both single-developer and multiple-developer scenarios. Finally, given these RTVs, the firm can simply implement the corresponding TP schemes that are shown to yield optimal transfer prices; therefore, these schemes do not need any modification.

4.2. Partially Reusable Elements

Next, we continue with our discussion on how our model treats a situation where an element is partially reusable. If an element is partially reusable, i.e., one part of the element can be developed as reusable while the rest of the element is not suitable for developing for reuse, then we consider each part as a separate element in our model. The reusable part has its own values for the reuse and specialization efforts. On the other hand, the reuse and specialization effort parameters for the nonreusable part can be set to sufficiently high numbers that make it infeasible to prepare this part for reuse.

4.3. Complexity Issues

The consideration of all possible reuse decisions for all the elements and group sequences results in the generation of a significant number of RTVs. For instance, recall from §2.1 that the total number of possible RTVs is $\Lambda = m! \cdot \prod_{i=1}^{n}(\gamma_i + 1)$, which is $O(m^{m+n})$. Depending on the number of groups and elements, this may add a substantial amount of computational burden in the execution of the TP schemes. To reduce computational complexity, a possible alternative in single developer situations (i.e., SU-SD and MU-SD) is to proceed with the TP schemes in phases, with each phase corresponding to an element. The justification for this idea comes from the following result:

Theorem 8. *Under a single developer scenario, for Problems D, U, and F, the optimal reuse decision for each element can be obtained independently from that for any other element.*

During a phase, the developer only creates RTVs considering alternative reuse decisions for a single element and the transfer prices for this element are obtained. Since the scheme proceeds element by element, the number of RTVs explored is $\Upsilon = m! \cdot \sum_{i=1}^{n}(\gamma_i + 1)$, which is $O(m^{m+1}n)$. Thus, the time required to arrive at a solution (i.e., a firm-optimal RTV) is linear in the number of the elements. A detailed discussion of the phase version of SU-SD (denoted by SU-SD-P) is provided in the appendix.

5. Summary

This paper studies different transfer pricing mechanisms to manage reuse conflicts in incremental software development. There are three stakeholders involved in the reuse problem (the user, the developer, and the firm), with each party having its own reuse priorities. We argue that because software reuse is a complex and technical issue, it is not well understood by the end users of a software application. End users ultimately care about the functionality provided by the software system, so that their work can be supported. Similarly, the firm (or the management within a firm) does not appreciate the technical difficulties involved in making optimal reuse decisions. Thus, of the three parties, only developers understand reuse, but all three parties are (differently) affected by it. This leads to possible conflicts among the stakeholders, but the firm is ultimately interested in solving the reuse problem from its overall perspective. A key step in solving the above problem is to map reuse decisions to RTVs to ensure that users can directly evaluate release times (rather than reuse decisions). The problems of identifying the optimal reuse vectors and optimal RTVs are shown to be isomorphic. This transformation of the problem (from the reuse space to the RTV space) is essential to ensure that users can provide the necessary input needed to find the optimal solution. Next, we apply transfer prices to manage in the RTV space. The calculation of optimal transfer prices is complicated by the fact that the firm does not know the true valuations that users place on the functionalities provided by the different groups in the software system. Under such information asymmetry, the Clarke mechanism is used as a truth revealing mechanism. Thus, the optimal RTV and the corresponding optimal reuse decisions are obtained. We present different transfer pricing problems starting from the simplest scenario to the most complex one. In the single-user, single-developer scenario, we demonstrate that there is no need to elicit the user's group valuations (e.g., in the form of bids); that is, the optimal transfer prices can be calculated without any user involvement. Of course, to do so, the firm must know the developer's cost of effort. As we move away from the simplest case and bring in the complexity of multiple users, it becomes necessary to elicit user valuations (in the form of bids, that are shown to be truth revealing) to arrive at the optimal transfer prices. Similarly, the added complexity of introducing multiple developers requires that we

charge transfer prices that account for the different costs of per unit effort across developers. The different scenarios illustrate different applications of transfer pricing that could occur in optimizing reuse decisions within a firm.

Acknowledgments

The authors thank Sandra Slaughter (the department editor), an associate editor, and three anonymous referees for insightful comments and suggestions that have greatly improved this paper.

Appendix A. Model Development

A.1. Proof of Theorem 1

Let M_i, $i \in \mathcal{I}$, be the element of interest. Recall that (1) M_i has γ_i usages in the project, (2) r_i is the development time of M_i, (3) $\alpha_i \in [0, 1]$ denotes the fraction of the development time needed to make M_i ready for reuse, and (4) $\beta_{ik} \in [0, 1)$ denotes the fraction of the development time needed to specialize M_i for its usage at G_k. For an arbitrary sequence S_σ, $\sigma \in \Sigma$, let $\mathcal{L}_i^\sigma = \{i_1^\sigma, i_2^\sigma, \ldots, i_{\gamma_i}^\sigma\}$ denote the index set of the groups that contain M_i. Let the members of set $\mathcal{L}_i^\sigma$ be listed in the same order M_i is required in sequence S_σ; that is, the lth usage of M_i is developed for $G_{i_l^\sigma}$, $i_l^\sigma \in \mathcal{L}_i^\sigma$. For example, suppose element M_a has one usage per group for the group sequence (G_4, G_5, G_8). Then, we have $\gamma_a = 3$ and $\mathcal{L}_a^\sigma = \{4, 5, 8\}$. Here, the second usage of M_a is developed for G_5.

Consider a developer-optimal solution, say, RTV $\bar{T}^{d^*}$. Assume that in this RTV, M_i is made ready for reuse at its lth usage where $\gamma_i \geq l > 1$, i.e., M_i is made ready for reuse at a usage other than its first usage. From Equation (1), the total time required to develop M_i in the project if M_i is made ready for reuse at its lth usage is as follows:

$$r_i l + r_i \alpha_i + r_i \sum_{t=l+1}^{\gamma_i} \beta_{i i_t^\sigma}. \tag{A1}$$

In the expression above, the first term $r_i l$ denotes the total effort spent in developing M_i for its first l usages in the sequence, the second term $r_i \alpha_i$ is the extra effort for making M_i reusable, and the last term is the total effort to specialize M_i for its usages following its lth usage in the sequence. If M_i is made ready for reuse at its first usage instead of its lth usage, then the change in the makespan of the project is $r_i(1 - l) + r_i \sum_{t=2}^{l} \beta_{i i_t^\sigma}$. Since $\beta_{ik} < 1$, $i \in \mathcal{I}$, $k \in \mathcal{K}$, we have $\sum_{t=2}^{l} \beta_{i i_t^\sigma} < l - 1$. Thus, we have $r_i(1 - l) + r_i \sum_{t=2}^{l} \beta_{i i_t^\sigma} < 0$, which implies that the solution $\bar{T}^{d^*}$ can be improved by making M_i ready for reuse at its first usage. This contradicts the assumption that $\bar{T}^{d^*}$ is a developer-optimal solution. The result now follows. □

Appendix B. Optimal Transfer Pricing

In this section, we provide the proofs of the results and the formal descriptions of the procedures that are introduced in §3.

B.1. Formal Description of SU-SD

Recall that $\mathcal{F} = \{1, 2, \ldots, \Lambda\}$ is the index set of all possible RTVs. Recall also that T_z^f is the release time of the last group for RTV $\bar{T}^f$.

Once the developer determines the full set of RTVs for the project, it forms the release-time matrix $\mathbf{T} = [\bar{T}^1 \bar{T}^2 \cdots \bar{T}^\Lambda]$ and the reuse-decision matrix $\mathbf{Y} = [\bar{Y}^1 \bar{Y}^2 \cdots \bar{Y}^\Lambda]$, and provides these two matrices to the firm. Matrix $\mathbf{T}$ holds the RTVs, whereas $\mathbf{Y}$ holds the corresponding reuse decision vectors. The formal description of SU-SD is as follows:

1. The developer generates the release-time matrix $\mathbf{T}$ and reuse-decision matrix $\mathbf{Y}$ and shares them with the firm.
2. The firm computes an RTV-cost, i.e., $XT_z^f - B^f$, for each $\bar{T}^f$, $f \in \mathcal{F}$. The firm then solves the problem $\min_{f \in \mathcal{F}} T_z^f X$ to obtain the developer-optimal RTV $\bar{T}^{d^*}$.
3. The firm forms the vector of transfer prices $\bar{p}$, with price components $p_f = X(T_z^f - T_z^{d^*}) - (B^f - B^{d^*})$, $f \in \mathcal{F}$.
4. The firm then shares the RTV matrix $\mathbf{T}$ and the transfer price vector $\bar{p}$ with the user.
5. The user solves the problem $\min_{f \in \mathcal{F}}(\sum_{k \in \mathcal{K}} T_k^f V_k + p_f)$. Let the user's optimal RTV be $\bar{T}^{u^*}$ and the corresponding transfer price be p_{u^*}.
6. If $p_{u^*} > 0$, the firm receives p_{u^*} from the user; otherwise, the user receives p_{u^*} from the firm. The firm also pays $X(T_z^{u^*} - T_z^{d^*})$ to the developer. The process ends. □

B.2. Proof of Theorem 2

Recall from §2.4 that the firm solves Problem F, i.e., $\min_{f \in \mathcal{F}}(\sum_{k \in \mathcal{K}} T_k^f V_k + XT_z^f - B^f)$. Recall from §3.1 that given the RTVs and their associated prices, the user solves $\min_{f \in \mathcal{F}}(\sum_{k \in \mathcal{K}} T_k^f V_k + p_f)$. Let us refer to this latter problem as Problem UP. From SU-SD, the process ends when the user identifies its optimal RTV as a solution to Problem UP. Using $p_f = X(T_z^{u^*} - T_z^{d^*}) - (B^f - B^{d^*})$ from Step 3 of SU-SD, Problem UP can be written as $\min_{f \in \mathcal{F}}(\sum_{k \in \mathcal{K}} T_k^f V_k + X(T_z^f - T_z^{d^*}) - (B^f - B^{d^*}))$. Since $XT_z^{d^*}$ and B^{d^*} are constants, the optimal solution is the same for both Problems F and UP. The result follows. □

B.3. Proof of Theorem 3

Recall from §§3.1 and B.2 that (1) the firm charges the user a price p_f for each RTV $\bar{T}^f$, $f \in \mathcal{F}$, (2) the firm does not charge the user any price for the developer-optimal RTV $\bar{T}^{d^*}$, i.e., $p_{d^*} = 0$, and (3) the optimal solution to Problem UP (i.e., the RTV selected by the user at the end of SU-SD; see §B.2) is RTV $\bar{T}^{u^*}$. Suppose the user incurs a higher cost at the end of SU-SD than that incurred for the developer-optimal solution, i.e., $\sum T_k^{u^*} V_k + p_{u^*} > \sum T_k^{d^*} V_k$. Since $p_{d^*} = 0$, this condition is equivalent to $\sum T_k^{u^*} V_k + p_{u^*} > \sum T_k^{d^*} V_k + p_{d^*}$. However, since both RTVs $\bar{T}^{u^*}$ and $\bar{T}^{d^*} \in \mathcal{F}$, this contradicts the fact that $\bar{T}^{u^*}$ is an optimal solution to the problem $\min_{f \in \mathcal{F}}(\sum_{k \in \mathcal{K}} T_k^f V_k + p_f)$. The result follows. □

B.4. Proof of Remark 1

Recall from §3.1 that (1) T_z^f denotes the release time of the last group for RTV $\bar{T}^f$, (2) $\bar{T}^{d^*}$ denotes the optimal RTV for the developer, (3) RTV $\bar{T}^{u^*}$ denotes the optimal solution to Problem UP (defined in §B.2), i.e., the RTV the user selects at the end of SU-SD, and (4) $X(T_z^{u^*} - T_z^{d^*})$ denotes the amount

that the firm pays the developer at the end of SU-SD. The total cost the developer incurs at the end of SU-SD can be written as the sum of the objective function value of the developer corresponding to the RTV $\bar{T}^{u^*}$ minus the firm's payment, i.e., $XT_z^{u^*} - X(T_z^{u^*} - T_z^{d^*}) = XT_z^{d^*}$. Thus, the developer attains its own optimal objective function value at the end of SU-SD. □

B.5. The Clarke Mechanism

The Clarke mechanism (also referred to as the Clarke tax algorithm; Clarke 1971) "ensures that individuals/groups will adequately consider the social cost of their influence on social outcomes, thereby ensuring truthful revelation of preferences" (Clarke 2000). The Clarke mechanism can be conveniently explained in the following setting: suppose the government is to choose a public project to implement from a set of available projects. Let each individual in society have intrinsic valuations for each of these projects. It is the government's aim to pick the project that maximizes social welfare. This can be achieved if the government knows the true valuations of the individuals. However, eliciting this information from the individuals may lead them to falsify, i.e., either deflate or inflate, their valuations for various projects to promote the selection of their preferred choices. Under the Clarke mechanism, it is a dominant strategy for each individual to report her true valuation for each project. Each person pays a tax equal to the externality her choice imposes on the other individuals, if she is pivotal to the project (i.e., if her choice influences the outcome). Otherwise, she pays no tax (Mas-Colell et al. 1995). In other words, each individual pays for the damage her preference can cause to the others.

B.6. MU-SD Transfer Price Scheme

Recall that (i) $\mathcal{Q} = \{U_1, U_2, \ldots, U_q\}$ denotes the set of user departments, and $\mathcal{E} = \{1, 2, \ldots, q\}$ denotes the corresponding index set; (ii) each U_η, $\eta \in \mathcal{E}$, uses a subset of the groups, and $\mathcal{K}_\eta \subseteq \mathcal{K}$ denotes the set of the groups that U_η utilizes; (iii) $\bar{T}^{d^*}$ denotes the optimal RTV for the developer; and (iv) T_z^f denotes the release time of the last group for the sequence corresponding to RTV $\bar{T}^f$. To choose the firm-optimal RTV, the firm places the RTVs $\bar{T}^1, \bar{T}^2, \ldots, \bar{T}^\Lambda$, on auction for the users. Prior to executing the auction, the firm informs the users and the developer of the pricing scheme. The developer generates the RTVs and shares the RTVs and their corresponding reuse decisions with the firm. Next, the firm shares the RTVs with the users.

The valuation of user U_η for RTV $\bar{T}^f$, $f \in \mathcal{F}$, is simply its objective function value, i.e., $v_{\eta f} = \sum_{k \in \mathcal{K}_\eta} T_k^f V_k$. Let the bid U_η makes for RTV $\bar{T}^f$ be denoted by $b_{\eta f}$. Thus, if U_η bids truthfully, then $b_{\eta f} = v_{\eta f}$. Let RTV $\bar{T}^w$ be the winning RTV at the end of the auction. Let $p_{\eta w}$, $w \in \mathcal{F}$, denote the transfer price (sum of the user externality, the developer externality, and the reuse externality) U_η pays the firm for RTV $\bar{T}^w$. Let $H_{\eta w}$ denote the total cost incurred by U_η for RTV $\bar{T}^w$, which is the sum of the transfer price and the bidder's valuation (bidder's true cost) for the RTV, i.e., $H_{\eta w} = p_{\eta w} + v_{\eta w}$.

B.6.1. Formal Description of MU-SD.

1. The firm announces the transfer pricing scheme that will be used under MU-SD.

2. The developer generates the release-time matrix **T** and reuse-decision matrix **Y** and shares them with the firm. The firm solves the problem $\min_{f \in \mathcal{F}} XT_z^f$ and identifies the developer-optimal RTV $\bar{T}^{d^*}$.

3. The firm shares the matrix **T** with the users. Each user U_η, $\eta \in \mathcal{E}$, reveals its bid $b_{\eta f}$ for RTV $\bar{T}^f$, $f \in \mathcal{F}$.

4. The firm picks the RTV $\bar{T}^w$, where $w = \arg\min_{f \in \mathcal{F}} (\sum_{\eta \in \mathcal{E}} b_{\eta f} + XT_z^f - B^f)$.

5. For U_η, $\eta \in \mathcal{E}$, the firm identifies the RTV $\bar{T}^{\varpi_\eta}$, where $\varpi_\eta = \arg\min_{f \in \mathcal{F}} (\sum_{\mu \neq \eta \in \mathcal{E}} (b_{\mu f}) + XT_z^f - B^f)$.

6. The firm computes the transfer prices $p_{\eta w} = \sum_{\mu \neq \eta \in \mathcal{E}} (b_{\mu w} - b_{\mu \varpi_\eta}) + B^w - B^{\varpi_\eta} + X(T_z^w - T_z^{\varpi_\eta})$ and informs the users of their corresponding prices.

7. Each user pays the firm its transfer price. The firm pays the developer $X(T_z^w - T_z^{d^*})$. The process ends.

B.7. Proof of Theorem 4

Recall from §B.6.1 that the firm-optimal RTV is identified by solving the following problem (in Step 4 of MU-SD):

$$w = \arg\min_{f \in \mathcal{F}} \left(\sum_{\eta \in \mathcal{E}} b_{\eta f} + XT_z^f - B^f \right). \tag{B1}$$

Recall from §B.6 that $v_{\eta f}$ and $b_{\eta f}$ denote the true cost and the bid of U_η for RTV $\bar{T}^f$, respectively. Let $\bar{T}^{w^*}$ be the choice the firm picks if U_η reveals its bids truthfully, and let $\bar{T}^{w'}$ be the firm's choice for an arbitrary misrepresented bid. If U_η reveals its group valuations truthfully, then we have $w^* = \arg\min_{f \in \mathcal{F}} (\sum_{\mu \neq \eta \in \mathcal{E}} (b_{\mu f}) + v_{\eta f} + XT_z^f - B^f)$. Thus, for the RTVs $\bar{T}^{w^*}$ and $\bar{T}^{w'}$, from (B1) we have the following inequality:

$$\sum_{\mu \neq \eta \in \mathcal{E}} b_{\mu w^*} + v_{\eta w^*} + XT_z^{w^*} - B^{w^*} \leq \sum_{\mu \neq \eta \in \mathcal{E}} b_{\mu w'} + v_{\eta w'} + XT_z^{w'} - B^{w'}. \tag{B2}$$

Let $\bar{T}^{\varpi}$ be the RTV the firm picks if the bids of U_η are excluded from the auction. Note that no assumption is made about whether the other users bid truthfully or not. Recall that $H_{\eta w}$ denotes the total cost U_η incurs if the firm picks RTV $\bar{T}^w$, i.e., $H_{\eta w} = v_{\eta w} + p_{\eta w}$. We examine the difference between the total cost U_η incurs when it truthfully reveals its valuations and when it does not, i.e., we examine $H_{\eta w^*} - H_{\eta w'}$.

$$\begin{aligned}
&H_{\eta w^*} - H_{\eta w'} \\
&= (v_{\eta w^*} + p_{\eta w^*}) - (v_{\eta w'} + p_{\eta w'}) \\
&= (v_{\eta w^*} - v_{\eta w'}) - (p_{\eta w'} - p_{\eta w^*}) = (v_{\eta w^*} - v_{\eta w'}) \\
&\quad - \left(\left(\sum_{\mu \neq \eta \in \mathcal{E}} (b_{\mu w'} - b_{\mu \varpi_\eta}) + XT_z^{w'} - B^{w'} - XT_z^{\varpi_\eta} + B^{\varpi_\eta} \right) \right. \\
&\quad \left. - \left(\sum_{\mu \neq \eta \in \mathcal{E}} (b_{\mu w^*} - b_{\mu \varpi_\eta}) + XT_z^{w^*} - B^{w^*} - XT_z^{\varpi_\eta} + B^{\varpi_\eta} \right) \right) \\
&= \left(v_{\eta w^*} + \sum_{\mu \neq \eta \in \mathcal{E}} b_{\mu w^*} + XT_z^{w^*} - B^{w^*} \right) \\
&\quad - \left(v_{\eta w'} + \sum_{\mu \neq \eta \in \mathcal{E}} b_{\mu w'} + XT_z^{w'} - B^{w'} \right).
\end{aligned} \tag{B3}$$

From (B2), we have $H_{\eta w^*} - H_{\eta w'} \leq 0$. The result follows. □

B.8. Proof of Theorem 5

Under MU-SD, the firm solves the problem $w = \arg\min_{f\in\mathcal{F}}(\sum_{\eta\in\mathcal{E}} b_{\eta f} + XT_z^f - B^f)$ to identify the firm-optimal solution. From Theorem 4, since the users' bids represent their true costs, we have $b_{\eta f} = \sum_{k\in\mathcal{K}_\eta} T_k^f V_k$, where $\eta \in \mathcal{E}$, $f \in \mathcal{F}$. Substituting these true costs, the firm's problem under MU-SD becomes $w = \arg\min_{f\in\mathcal{F}}(\sum_{\eta\in\mathcal{E}}(\sum_{k\in\mathcal{K}_\eta} T_k^f V_k) + XT_z^f - B^f)$. It is straightforward to see that this problem is the same as Problem F. The result follows. □

B.9 Proof of Theorem 6

Let $\bar{T}^{\eta^*}$ denote the RTV corresponding to the maximum objective function value of user U_η, $\eta \in \mathcal{E}$. Let $\bar{T}^w$ (respectively, $\bar{T}^{\varpi}$) denote the firm-optimal RTV under MU-SD when the bids of U_η are considered (respectively, are not considered). Recall from Theorem 4 that, under MU-SD, the users bid their true costs, i.e., they bid their true objective function values. We examine the following three cases:

(i) If $\bar{T}^{\eta^*} = \bar{T}^w$, then the firm-optimal solution corresponds to the RTV for which U_η incurs its maximum true cost. Then, clearly the *sum* of the true costs of the other parties is minimum at this RTV. Thus, the bids of U_η do not have any influence on the firm's choice. Therefore, we have $\bar{T}^w = \bar{T}^{\varpi}$. In other words, U_η does not exert any externality on the other users, the firm, and the developer, i.e., $p_{\eta w} = 0$. Since U_η does not pay any transfer price, the total cost U_η incurs under MU-SD equals its maximum true cost.

(ii) If $\bar{T}^{\eta^*} \neq \bar{T}^w$ and $\bar{T}^w = \bar{T}^{\varpi}$, then the bids of U_η do not influence the firm's decision. Therefore, the transfer price $p_{\eta w} = 0$. Also, by definition, the objective function value of U_η for $\bar{T}^w$ is at most that for $\bar{T}^{\eta^*}$. Thus, the result again follows.

(iii) If $\bar{T}^{\eta^*} \neq \bar{T}^w$ and $\bar{T}^w \neq \bar{T}^{\varpi}$, then the transfer price that U_η pays is equal to the difference between the sum of the costs of the other parties for RTVs $\bar{T}^w$ and $\bar{T}^{\varpi}$, i.e., $p_{\eta w} = (\sum_{\mu\neq\eta\in\mathcal{E}} b_{\mu w} + XT_z^w - B^w) - (\sum_{\mu\neq\eta\in\mathcal{E}} b_{\mu\varpi} + XT_z^{\varpi} - B^{\varpi}) \geq 0$. Recall from §B.6 that $v_{\eta w}$ denotes the true cost of U_η for RTV $\bar{T}^w$. When the bids of U_η are considered, the firm-optimal solution is RTV $\bar{T}^w$. Thus, we have $(\sum_{\mu\neq\eta\in\mathcal{E}} b_{\mu w} + v_{\eta w} + XT_z^w - B^w) \leq (\sum_{\mu\neq\eta\in\mathcal{E}} b_{\mu\varpi} + v_{\eta\varpi} + XT_z^{\varpi} - B^{\varpi})$. Using the expression for $p_{\eta w}$, this condition is $p_{\eta w} \leq v_{\eta\varpi} - v_{\eta w}$. Since $p_{\eta w} \leq v_{\eta\varpi} - v_{\eta w}$, we have $v_{\eta w} + p_{\eta w} \leq v_{\eta\varpi} \leq v_{\eta\eta^*}$. The result follows. □

B.10. Formal Description of MD

Recall that $\bar{E}^f = (E_1^f, E_2^f, \ldots, E_\phi^f)$ is the "effort vector" whose component E_θ^f denotes the total effort of developer D_θ corresponding to RTV $\bar{T}^f$, where $\theta \in \Phi$ and $f \in \mathcal{F}$. Let the effort matrix of the developers be denoted by $\mathbf{E} = [\bar{E}^1 \bar{E}^2 \ldots \bar{E}^\Lambda]$. A formal description of MD is as follows:

1. The developers generate the release-time matrix **T**, the reuse-decision matrix **Y**, and the effort matrix **E** and share them with the firm.

2. Using the matrices **T** and **E**, the firm solves the problem $d^* = \arg\min_{f\in\mathcal{F}} \sum_{\theta\in\Phi} E_\theta^f X_\theta$ to identify the global developer-optimal RTV $\bar{T}^{d^*}$, $\theta \in \Phi$, $f \in \mathcal{F}$.

Single-user (*SU-MD*) *case*:

3. The firm forms the vector of transfer prices $\bar{p}$, with price components $p_f = \sum_{\theta\in\Phi}(E_\theta^f - E_\theta^{d^*})X_\theta - (B^f - B^{d^*})$, $f \in \mathcal{F}$.

4. The firm follows Steps 4 and 5 of SU-SD and then goes to Step 8 of MD.

Multiple-user (*MU-MD*) *case*:

5. The firm identifies RTV $\bar{T}^w$, where $w = \arg\min_{f\in\mathcal{F}} \sum_{\eta\in\mathcal{E}} b_{\eta f} + \sum_{\theta\in\Phi} E_\theta^f X_\theta - B^f$.

6. For U_η, $\eta \in \mathcal{E}$, the firm identifies RTV $\bar{T}^{\varpi_\eta}$, where $\varpi_\eta = \arg\min_{f\in\mathcal{F}} \sum_{\mu\neq\eta\in\mathcal{E}}(b_{\mu f}) + \sum_{\theta\in\Phi} E_\theta^f X_\theta - B^f$.

7. The firm computes the transfer prices $p_{\eta w} = \sum_{\mu\neq\eta\in\mathcal{E}}(b_{\mu w} - b_{\mu\varpi_\eta}) + B^w - B^{\varpi_\eta} + \sum_{\theta\in\Phi} X_\theta(E_\theta^w - E_\theta^{\varpi_\eta})$ and informs the users of their corresponding prices.

8. The firm receives the respective payment(s) from the user(s). If $\bar{T}^f$ denotes the RTV finally chosen by the firm, then it pays developer D_θ the amount $\sum_{\theta\in\Phi}(E_\theta^f - E_\theta^{d^*}) \cdot X_\theta(E_\theta^f/(\sum_{\theta\in\Phi} E_\theta^f))$. The process ends. □

B.11. Proof of Theorem 8

For ease of exposition, we consider a fixed group sequence, say, $S_\sigma = (G_1, G_2, \ldots, G_m)$, and suppress the index σ hereafter in this proof.

Consider element M_i. Let $\mathcal{L}_i = \{i_1, i_2, \ldots, i_{\gamma_i}\}$ denote the index set of the groups that contain M_i. In other words, M_i is required by groups $G_{i_1}, G_{i_2}, \ldots, G_{i_{\gamma_i}}$. Except for M_i, fix the reuse decision for each of the other elements to any arbitrary choice. Thus, only the reuse decision of M_i remains to be decided. Let $\bar{T}^{i_0}$ be the corresponding RTV if M_i is not made ready for reuse, and let $\bar{T}^{i_l}$ be the corresponding RTV if M_i is made ready for reuse at its lth usage, $l \in \mathcal{L}_i$. We examine the RTVs corresponding to these $\gamma_i + 1$ different reuse decisions for M_i:

- *M_i is never made ready for reuse*. In this case, let the corresponding RTV be

$$\bar{T}^{i_0} = (T_1^{i_0}, T_2^{i_0}, \ldots, T_{i_1-1}^{i_0}, T_{i_1}^{i_0}, \ldots, T_{i_2-1}^{i_0}, T_{i_2}^{i_0}, \ldots, T_{i_3-1}^{i_0}, T_{i_3}^{i_0}, \ldots, T_{m-1}^{i_0}, T_m^{i_0}),$$

where $T_k^{i_0}$ denotes the release time of G_k.

- *M_i is made ready for reuse at its first usage, i.e., at its usage for G_{i_1}*. The RTV corresponding to this case is

$$\begin{aligned}\bar{T}^{i_1} = \Bigg(&T_1^{i_0}, T_2^{i_0}, \ldots, T_{i_1-1}^{i_0}, T_{i_1}^{i_0} + r_i\alpha_i, \ldots, T_{i_2-1}^{i_0} + r_i\alpha_i, T_{i_2}^{i_0} + r_i\alpha_i \\ &- r_i(1-\beta_{ii_2}), \ldots, T_{i_3-1}^{i_0} + r_i\alpha_i - r_i(1-\beta_{ii_2}) T_{i_3}^{i_0} + r_i\alpha_i \\ &- r_i(1-\beta_{ii_2}) - r_i(1-\beta_{ii_3}), \ldots, T_{m-1}^{i_0} \\ &+ r_i\Big(\alpha_i - \sum_{j=2}^{\gamma_i}(1-\beta_{ii_j})\Big), T_m^{i_0} + r_i\Big(\alpha_i - \sum_{j=2}^{\gamma_i}(1-\beta_{ii_j})\Big)\Bigg).\end{aligned}$$

We examine the difference in the corresponding entries of RTVs $\bar{T}^{i_1}$ and $\bar{T}^{i_0}$. This difference is 0 for groups G_1 through G_{i_1-1}, $r_i\alpha_i$ for groups G_{i_1} through G_{i_2-1}, and $r_i\alpha_i - ri(1-\beta_{ii_2})$ for groups G_{i_2} through G_{i_3-1}, etc.; that is,

$$\begin{aligned}\bar{T}^{i_1} - \bar{T}^{i_0} = \Bigg(&\underbrace{0, 0, \ldots, 0}_{i_1-1}, \underbrace{r_i\alpha_i, \ldots, r_i\alpha_i}_{i_2-i_1}, \underbrace{r_i\alpha_i - r_i(1-\beta_{ii_2}), \ldots, r_i\alpha_i - r_i(1-\beta_{ii_2})}_{i_3-i_2}, \\ &\underbrace{r_i\alpha_i - r_i(1-\beta_{ii_2}) - r_i(1-\beta_{ii_3}), \ldots, r_i\alpha_i - r_i(1-\beta_{ii_2}) - r_i(1-\beta_{ii_3})}_{i_4-i_3}, \\ &\ldots, r_i\Big(\alpha_i - \sum_{j=2}^{\gamma_i}(1-\beta_{ii_j})\Big), r_i\Big(\alpha_i - \sum_{j=2}^{\gamma_i}(1-\beta_{ii_j})\Big)\Bigg).\end{aligned}$$

• *M_i is made ready for reuse at its lth usage, i.e., at its usage for G_{i_l}.* To express the change in the release times of the groups as compared to those in $\bar{T}^{i_0}$, we need the following notation: (1) Let $h_{jk}=1$ if $j\le k$ and 0 otherwise, and (2) let $\Delta_k^{i_l}$ be the difference between the kth members of RTVs $\bar{T}^{i_l}$ and $\bar{T}^{i_0}$, i.e., $\Delta_k^{i_l}=T_k^{i_l}-T_k^{i_0}$. We have the following general expression for this difference:

$$\Delta_k^{i_l}=h_{i_lk}r_i\left(\alpha_i-\sum_{j=l+1}^{\gamma_i}h_{i_jk}(1-\beta_{ii_j})\right). \tag{B4}$$

In the above expression:

(i) If $k\le i_l-1$, then $h_{i_lk}=0$ and the release time of G_k is not affected by this reuse decision, i.e., $\Delta_k^{i_l}=0$.

(ii) If $k=i_l$, then $h_{i_lk}=1$ and $h_{i_jk}=0$ for $j>l$. The release time of G_k changes only by the amount of extra reuse effort, $\alpha_i r_i$. Thus, $\Delta_k^{i_l}=\alpha_i r_i$.

(iii) If $k\ge i_l+1$, then $h_{i_lk}=1$. The change in the release time of G_k in this case has two components: (1) $\alpha_i r_i$, when M_i is made ready for reuse at its lth usage, and (2) total savings in development time of M_i for each of its usages following that for G_{i_l} and until G_k. This component can be expressed as $r_i\sum_{j=l+1}^{\gamma_i}h_{i_jk}(1-\beta_{ii_j})$. Notice that for $i_j>k$, we have $h_{i_jk}=0$, i.e., we do not consider any of M_i's usages following G_k.

The key observation is that $\Delta_k^{i_l}$, $l\in\mathcal{L}_i$, only depends on the parameters of M_i. In other words, the reuse decisions for other elements do not have any impact on the decision for M_i. It is therefore immediate that the optimum reuse decision for M_i can be obtained by enumerating its γ_i+1 alternatives. It follows that the optimal solution to Problems D, U, and F can be obtained by enumerating the reuse alternatives of each element M_i, $i\in\mathcal{I}$, separately. □

B.12. Description of SU-SD-P

The TP schemes introduced earlier in this paper require the generation of all possible RTVs corresponding to all group sequences and reuse vectors. The number of RTVs that are required to be searched through the procedure increases exponentially in the number of elements. In this subsection, we develop a new TP scheme, namely, SU-SD-P, which reduces the number of RTVs to be explored under SU-SD. Under SU-SD-P, the number of RTVs required increases *linearly* in the number of elements. Furthermore, all the properties of SU-SD (see §3.1.3) hold for SU-SD-P as well.

Recall that $\Sigma=\{S_1,S_2,\ldots,S_\tau\}$ is the set of all group sequences, and $S_\sigma\in\Sigma$ denotes a particular sequence drawn from this set. Let $\Pi=\{1,2,\ldots,\tau\}$ be the index set for Σ. SU-SD-P proceeds by identifying an optimal RTV for each group sequence in Σ. Consider an arbitrary group sequence, say, $S_\sigma\in\Sigma$. For sequence S_σ, the reuse decisions and transfer prices for the elements are decided one at a time (i.e., in phases), starting from element M_1, then M_2, and so on, until M_n. During any part of SU-SD-P, if the firm is yet to determine the transfer price for an element, then that element is (temporarily) assumed to be not made ready for reuse (i.e., the reuse decision for this element is *no-reuse*).

Let M_i be the element of interest. We denote the corresponding phase by e_i. Thus, the reuse decision for M_i is taken in phase e_i. After an RTV corresponding to M_i is selected, i.e., a reuse decision for M_i is made, phase e_{i+1} starts for the next element, M_{i+1}. The reuse decisions from the previous phases are carried over to the next phase. Thus, during phase e_i, the reuse decisions for the elements prior to M_i are known, while the elements subsequent to M_i are assumed to be not made ready for reuse. Since the reuse decisions for elements prior to and after M_i are now fixed, the developer forms γ_i+1 RTVs, with each corresponding to a distinct reuse decision for M_i. Thus, the developer determines γ_i+1 RTVs for M_i. Let $\mathcal{R}_i^\sigma=\{\sigma_1,\sigma_2,\ldots,\sigma_{\gamma_i+1}\}$ be the index set of these RTVs. The RTVs themselves are denoted by $\bar{T}_i^{\sigma_l}=(T_{i1}^{\sigma_l},T_{i2}^{\sigma_l},\ldots,T_{im}^{\sigma_l})$, $\sigma_l\in\mathcal{R}_i^\sigma$, with the component $T_{ik}^{\sigma_l}$ denoting the release time of G_k. Let $T_{iz}^{\sigma_l}=\max_{k\in\mathcal{K}}T_{ik}^{\sigma_l}$; in words, $T_{iz}^{\sigma_l}$ is the makespan corresponding to RTV $\bar{T}_i^{\sigma_l}$. Once the developer generates the γ_i+1 RTVs for M_i, it forms the RTV matrix $\mathbf{T}_i^\sigma=[\bar{T}_i^{\sigma_1}\bar{T}_i^{\sigma_2}\ldots\bar{T}_i^{\sigma_{\gamma_i+1}}]$. Let $y_i^{\sigma_l}=1$ if M_i is made ready for reuse in RTV $\bar{T}_i^{\sigma_l}$; otherwise, $y_i^{\sigma_l}=0$. Let $\bar{Y}_i^{\sigma_l}=(y_i^{\sigma_1},y_i^{\sigma_2},\ldots,y_i^{\sigma_{\gamma_i+1}})$ be the reuse vector corresponding to the RTV matrix $\mathbf{T}_i^\sigma$.

After determining the RTV matrix $\mathbf{T}_i^\sigma$ and the reuse vector $\bar{Y}_i^{\sigma_l}$, the developer provides them to the firm. Similar to SU-SD, the firm first identifies the developer-optimal RTV $\bar{T}_i^{\sigma_{d^*}}$ and then attaches a transfer price $p_i^{\sigma_l}$ for each RTV, calculated as $p_i^{\sigma_l}=X(T_{iz}^{\sigma_l}-T_{iz}^{\sigma_{d^*}})-B_i(y_i^{\sigma_l}-y_i^{\sigma_{d^*}})$. Thus, we have a vector of transfer prices for M_i. On receiving the RTVs and the corresponding vector of transfer prices from the firm, the user obtains its optimal RTV (and thereby also chooses the reuse decision for M_i), and the firm records the transfer price for this RTV. Once the reuse decision corresponding to the user-chosen RTV for M_i is fixed, the process moves forward to M_{i+1} (i.e., phase e_{i+1}) and continues until the reuse decision for the last element M_n is taken in phase e_n.

Let RTV $\bar{T}_n^{\sigma^*}$ be the RTV the user selects at the end of phase e_n for sequence S_σ, $\sigma\in\Pi$, and let RTV $\bar{T}_n^{u^*}$ be the RTV the user incurs its minimum total cost (the sum of its objective function value and the transfer price) across all sequences, i.e., $\bar{T}_n^{u^*}=\min_{\sigma\in\Pi}(\sum_{k\in\mathcal{K}}T_{nk}^{\sigma*}V_k+\sum_{i=1}^n p_i^{\sigma^*})$. The user then pays (if needed) the firm the transfer price corresponding to RTV $\bar{T}_n^{u^*}$, and the firm compensates (if needed) the developer for its loss.

B.13. Formal Description of SU-SD-P

1. Set $\text{Temp}_1=\infty$, $T_z^{d^*}=\infty$, and $p^{\sigma^*}=0$. The variable Temp_1 is a temporary variable used to compare the total cost of the user across the different sequences. The variable p^{σ^*} holds the sum of the transfer prices for all elements in the solution at the end of SU-SD-P.

2. Starting with group sequence S_1, SU-SD-P proceeds in order from sequence S_1, S_2, through S_τ.

3. Let the sequence of interest be S_σ, $\sigma\in\Pi$.

4. Set the reuse decisions for all elements to *no-reuse*. Let $p^\sigma=0$.

5. Starting with element M_1, proceed in order starting with M_1, then M_2, and so on, until M_n.

6. Let the element of interest be M_i, $i\in\mathcal{I}$.

7. The developer generates the release-time matrix $\mathbf{T}_i^\sigma$ and the corresponding reuse vector $\bar{Y}_i^{\sigma_l}$ and shares them with the firm. The firm solves the problem $\min_{\sigma_l\in\mathcal{R}_i^\sigma}T_{iz}^{\sigma_l}X$ to identify the developer-optimal RTV $\text{PT}_i^{\sigma_d^*}$.

8. The firm forms the vector of transfer prices $\bar{p}_i^\sigma$ with price components $p_i^{\sigma_l}=(T_{iz}^{\sigma_l}-T_{iz}^{\sigma_d^*})X-B_i(y_i^{\sigma_l}-y_i^{\sigma_d^*})$. The

firm then shares the release-time matrix $\mathbf{T}_i^{\sigma}$ and the vector of transfer prices $\bar{p}_i^{\sigma}$ with the user.

9. The user solves the problem $\min_{\sigma_l \in \mathcal{R}_i^{\sigma}} (\sum_{k \in \mathcal{K}} T_{ik}^{\sigma_l} V_k + p_i^{\sigma_l})$. Let the user's optimal RTV be $\bar{T}_i^{\sigma}$ and the corresponding optimal transfer price be p_i^{σ}. Note that RTV $\bar{T}_i^{\sigma}$ corresponds to a reuse decision for M_i.

10. The reuse decision corresponding to RTV $\bar{T}_i^{\sigma}$ is fixed and $p^{\sigma} \leftarrow p^{\sigma} + p_i^{\sigma}$.

11. If $i \neq n$, then $i = i + 1$ and the firm initiates the next phase for M_{i+1} by returning to Step 6.

12. If $i = n$, then $\text{Temp}_2 = \sum_{k \in \mathcal{K}} T_{nk}^{\sigma} V_k + p^{\sigma}$. If $\text{Temp}_2 < \text{Temp}_1$, then $\text{Temp}_1 \leftarrow \text{Temp}_2$, $p^{\sigma^*} \leftarrow p^{\sigma}$, $\bar{T}^{u^*} \leftarrow \bar{T}_n^{\sigma}$, and $T_{nz}^{\sigma} \leftarrow \max_{k \in \mathcal{K}} T_{nk}^{\sigma}$. If $T_{nz}^{\sigma} < T_z^{d^*}$, then $T_z^{d^*} \leftarrow T_{nz}^{\sigma}$.

13. If $\sigma \neq \tau$, then the firm initiates Step 3 for sequence $S_{\sigma+1}$.

14. If $\sigma = \tau$, the result of SU-SD-P is the RTV $\bar{T}^{u^*}$. If the transfer price $p^{\sigma^*} > 0$ (respectively, $p^{\sigma^*} < 0$), then the user pays (respectively, receives) this price to (respectively, from) the firm. The completion time corresponding to RTV $\bar{T}^{u^*}$ is $T_z^{u^*} = \max_{k \in \mathcal{K}} T_k^{u^*}$. The firm pays the developer $X(\bar{T}_z^{u^*} - \bar{T}_z^{d^*})$. The process ends. □

References

Baiman S, Fischer P, Rajan MV, Saouma R (2007) Resource allocation auctions within firms. *J. Accounting Res.* 45(5):915–946.

Baldenius T, Melumad ND, Reichelstein S (2004) Integrating managerial and tax objectives in transfer pricing. *Accounting Rev.* 79(3):591–615.

Banker RD, Datar SM (1992) Optimal transfer pricing under postcontract information. *Contemporary Accounting Res.* 8(2): 329–352.

Banker RD, Kemerer CF (1992) Performance evaluation metrics for information systems development: A principal-agent model. *Inform. Systems Res.* 3(4):379–400.

Christensen HB, Ron H (2000) A case study of horizontal reuse in a project-driven organization. *Proc. Seventh Asia Pasific Software Engrg. Conf.* (IEEE, Washington, DC), 292–298.

Clarke EH (1971) Multipart pricing of public goods. *Public Choice* 11(1):17–33.

Clarke EH (2000) Public Goods Home Page. Accessed July 3, 2013, http://www.clarke.pair.com/Testpubgoods.html.

Groves T, Loeb M (1979) Incentives in a divisionalized firm. *Management Sci.* 25(3):221–230.

Gurbaxani V, Kemerer CF (1989) An agent-theoretic perspective on the management of information systems. *Proc. 22nd Hawaii Internat. Conf. System Sci.* (IEEE, Washington, DC), 141–150.

Gurbaxani V, Kemerer CF (1990) An agency theory view of the management of end-user computing. *Proc. 11th Internat. Conf. Inform. Systems* (Association for Information Systems, Atlanta), 279–289.

Harris M, Kriebel C, Raviv A (1982) Asymmetric information, incentives, and intrafirm resource allocation. *Management Sci.* 28(6):604–620.

Kumar C, Altinkemer K, De P (2010) A mechanism for pricing and resource allocation in peer-to-peer networks. *Electronic Commerce Res. Appl.* 10(1):26–37.

Lim WC (1994) Effects of reuse on quality, productivity, and economics. *IEEE Software* 11(5):23–30.

Liu D, Dawande M, Mookerjee V (2007) Value-driven creation of functionality in software projects: Optimal sequencing and reuse. *Productions Oper. Management* 16(3):381–399.

Lynex A, Layzell PJ (1998) Organisational considerations for software reuse. *Ann. Software Engrg.* 5(1):105–124.

Mas-Colell A, Whinston MD, Green JR (1995) *Microeconomic Theory* (Oxford University Press, New York).

Mattikalli R (2011) Personal communication. The Boeing Company, Chicago.

Miller BL, Buckman AG (1987) Cost allocation and opportunity costs. *Management Sci.* 33(5):626–639.

Mohagheghi P (2004) The impact of software reuse and incremental development on the quality of large systems. Ph.D. thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Mohagheghi P, Conradi R (2008) An empirical investigation of software reuse benefits in a large telecom product. *ACM Trans. Software Engrg. Methodology* 17(3, Article 13):1–32.

NASA (National Aeronautics and Space Administration) (2012) Earth science data systems software reuse group. Accessed July 4, 2013, http://earthdata-uat.nasa.gov/our-community/esdswg/software-reuse-srwg/about-us.

Poulin JS (1995) Populating software repositories: Incentives and domain-specific software. *J. Systems Software* 30(3):187–199.

Ramachandran M (2005) Software reuse guidelines. *ACM SIGSOFT Software Engrg. Notes* 30(3):1–8.

Stuart A (2005) Market magic. *CFO Magazine* (November 1), http://www.cfo.com/article.cfm/5077917/.

Tan Y, Mookerjee V (2005) Optimal spending allocation between advertising and information technology in electronic retailing. *Management Sci.* 51(2):1236–1249.

Vaysman I (1998) A model of negotiated transfer pricing. *J. Accounting Econom.* 25(1):349–384.

Wang ETG, Barron T, Seidmann A (1997) Contracting structures for custom software development: The impacts of informational rents and uncertainty on internal development and outsourcing. *Management Sci.* 43(12):1726–1744.

Westland JC (1992) Congestion and network externalities in the short run pricing of information system services. *Management Sci.* 38(7):992–1009.