Software Development and Trade[*]

by

Larry D. Qiu
Hong Kong University of Science and Technology

This version: April 2001

Abstract

We develop a model to study the implications of a legal environment on software development and international trade. We show that the degree of contract enforcement affects the organizational mode (i.e., in-house versus outsourcing) of customized software development. In autarky, a country with weak copyright protection develops customized software only, while a country with strong copyright protection develops both customized software and packaged software. After opening to trade, the strong-copyright-protecting country exports packaged software and the weak-copyright-protecting country exports customized software.

General field:	International Trade, Industrial Organization
JEL Classification No.:	F12, L22, L86
Key Words:	software, piracy, copyright protection, contract enforcement, vertical integration, contract, pattern of trade.

* I have benefited from comments by Leonard Cheng and Brian Copeland and discussions with Son Ku Kim, Zhigang Tao, Susheng Wang, Oliver Williamson, and seminar participants at HKUST, the "International Economics and Asia" Workshop held at the City University of Hong Kong (July 2000), and the Second Annual Conference of the European Trade Study Group held in Glasgow (September 2000).

Correspondence to: Department of Economics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Fax number: +(852)2358-2084, Email: larryqiu@ust.hk, Web site: http://www.bm.ust.hk/~larryqiu.

"Industry representatives believe that the problem of copyright piracy is the number one 'trade' issue faced by the U.S. motion picture and software industries alike."
—— Siwek and Furchtgott-Roth (1993, p. 60)

1. Introduction

We are now living in the new economy in which information technology is playing a more and more important role. For example, in the United States, computer software is vital to both the domestic economy and external trade. In 1998, the software industry became the second largest industry group in manufacturing.[1] In 1997, packaged software alone contributed a surplus of $13 billion to the U.S. trade balance, without which the U.S. trade deficit (excluding U.S. military and government transactions) would have been 36% higher.[2] However, software development is very uneven across countries. In 1994, the U.S. controlled about 75% of the global software market. Europe had 20% of the market, and Japan had 4.3%.[3]

Unlike most products and services, though, software is protected by copyright laws against piracy.[4] Software piracy is a serious problem all over the world, but it varies tremendously from country to country. According to a report by the Business Software Alliance (BSA, 1999a), in 1998, 38% of the business software applications in the world were pirated,[5] and, by countries, the software piracy rate ranged from 25% (in the U.S.) to 97% (in Vietnam).[6]

It is not a coincidence that the U.S. has the lowest software piracy rate and at the same time has the largest market share of software in the world. The BSA concludes that "[p]rotecting the intellectual property rights that are the basis of packaged software distribution is conducive

[1]The software industry tied with the electronic components and accessories industry for second place, after the motor vehicles and equipment industry, according to a report by the Business Software Alliance (BSA, 1999b, p. 6).

[2]This information is obtained from BSA (1999b, p. 16).

[3]The market share figures come from an estimation by International Data Corporation (*Fortune*, 1994).

[4]Besen and Raskind (1991) give a clear description of the American copyright law. Computer software is protected by copyright law rather than by patent law. Patents protect the application of an idea in the form of a machine, method, or matter. In contrast, copyright protects the expression of the idea.

[5]This was the first time that the annual piracy rate had dropped below 40% (BSA, 1999a).

[6]See BSA (1999a) for other countries' piracy rates.

to greater international trade and increased industry investment in the economies that provide such protections" (BSA, 1998, p. 24). In this paper, we develop a model to analyze software development and trade formally with a focus on the influence of a country's legal environment, namely its contract enforcement and copyright protection policies.

Software products are commonly classified into two groups: *packaged software*, which is designed for general purposes, like word processing, and *customized software*, which is designed for special use, like an accounting program for a particular company.[7] In our model, we show that the degree of contract enforcement affects the organizational *mode* of the customized software development, i.e., whether the software is developed in-house (vertical integration) or obtained by outsourcing (contracting). For packaged software, since piracy reduces legitimate consumption, copyright protection is essential to ensure investment in software development. As a result, under autarky, a country with weak copyright protection develops customized software only, while a country with strong copyright protection develops both customized and packaged software. After opening to trade, software developers in both countries face the same world market and consequently equal copyright protection, but the pattern of specialization and trade remains as if they were still facing unequal copyright protection as under autarky: the strong-copyright-protecting country exports packaged software, and the weak-copyright-protecting country exports customized software. This is because a strong copyright regime is a source of comparative advantage while learning-by-doing and network externalities tend to reenforce this comparative advantage.

Although this paper does not attempt to explain the whole story of software development and trade, its prediction is in fact consistent with empirical observations. The U.S. is the dominant developer and exporter of packaged software in the world. In 1993, the U.S. producers of packaged software controlled about 60% of the world market (Siwek and Furchtgott-Roth, 1993, p. 60). It is also clear that the U.S. is far ahead of the EU and Japan in promoting legal

[7]Examples of packaged software (SIC 7372) include operating systems, word processing programs and spreadsheets. Examples of customized software (SIC 7370, excluding 7372) include some specific software designed for banks and some professional training programs. See Torrisi (1998) for more about this classification.

protection in software innovation.[8] The U.S. extended copyright protection to software in 1980 (the Software Amendment is based on the U.S. Copyright Act enacted in 1976), but it was not until 1991 that the European Commission first issued a directive concerning the application of copyright to software.[9]

All other countries, including those in the EU, more or less concentrate on developing customized software.[10] For example, customized software sales accounted for 84% of total Japanese software sales according to a 1990 survey by the MITI (Ministry of International Trade and Industry).[11] In-house development of customized software is popular. According to a survey by MITI in 1988, 58% of software engineers in Japan worked for user companies (i.e., in-house development), and 35% worked for software houses. Among the software companies in Japan, user spin-off software houses accounted for 31.3% of market sales (representing semi-in-house development).[12] Most software firms in China[13], India[14] and Latin America are reported to produce primarily customized software. As described by Zhang and Wang (1995, p. 66), in China, "[m]ost of the local software companies were generally limited to developing software for individual customers according to their specifications, on a customer-by-customer basis. Consequently, these software companies had hardly any market-oriented product of their own and acted only as subcontractors."[15] Although India has a large pool of highly skilled computer programmers, the country does not develop a significant amount of packaged software,[16] as predicted by our model, because its copyright protection is rather weak [the piracy rate was 65%

[8]This also leads Torrisi (1998, p. 156) to conclude that the differences in copyright protection, together with some other factors, "explain why US firms were the first to enter the market of packaged software, pre-empting the entrance of European producers."

[9]See Torrisi (1998, p. 101) for more details.

[10]See Torrisi (1998, p. 156).

[11]See Baba *et al.* (1995, p. 478).

[12]See Baba *et al.* (1995).

[13]See Zhang and Wang (1995).

[14]See Correa (1996, p. 172).

[15]In China, "the first law to protect copyright, the 'Copyright Law of the People's Republic of China,' came into effect in June 1991. Due to the inadequate legal framework, it is understandable that the software market simply was not ready to emerge in the 1980s." (Zhang and Wang, 1995, p. 43).

"With only a few software package products available in the market, most of the application systems were either developed in-house or ordered from outside on an individual customer-design basis." (Zhang and Wang, 1995, p. 48).

[16]See Arora and Asundi (1999) and Arora *et al.* (1999).

in 1998 according to BSA (1999a)]. With comparative advantage in this area, these countries export customized software.[17]

The model developed in this paper has several distinguishing features that are crucial for analyzing software development and trade. The industry is assumed to produce two distinct products, and we examine the implications of contract enforcement and copyright protection for these products, respectively. In this way, the present study is related to but significantly different from others in the literature that we discuss below.

First, consider customized software. There are three alternative modes through which a piece of software is developed and later delivered to the user: vertical integration, a contract and the market. To emphasize the difference between customized software and packaged software, we assume that any customized software is so specific that it has only one user. This immediately eliminates the market as a plausible development and delivery mode. Therefore, our analysis focuses on a comparison between vertical integration and contracts. Unlike us, in analyzing upstream and downstream relationships, McLaren (1998) and Grossman and Helpman (1999) take the incomplete contract approach and assume away contracts in their models. Naturally, they do not touch on the issue of imperfect enforcement. Their focus is a comparison of vertical integration with the market.[18]

In the present model, contracts are complete[19] but enforcement is imperfect, similar to the model proposed by Anderson and Young (2000).[20] Unlike us, however, they consider contracts on trade between two parties from different countries and examine the implications of imperfect enforcement for international trade. We examine whether contracts are preferred to vertical

[17]It is well recognized that India is the most successful developing country to export customized software. According to a study cited by Correa (1996, p. 178), a substantial part of Indian software exports ($85 - 90\%$) is a type of "body-shopping", i.e., customized software. In China, "almost all the exported software consists of customer-designed applications under one-time contracts" (Zhang and Wang, 1995, p. 71).

[18]McLaren (1998) shows that there exist multiple equilibria about the production modes, but international openness makes the market (or outsourcing) more likely than vertical integration to be the equilibrium. Grossman and Helpman (1999) characterize the general equilibrium of industrial organization in the presence of downstream competition.

[19]See Tirole (1999) for an excellent review of the incomplete contract literature and the "debate" between complete contract theorists and incomplete contract theorists.

[20]Recently, Zhang and Zhu (2000) showed that contracts can still be complete even if the court has imperfect information to verify each contingency.

integration as an organizational mode for customized software development between two parties within the same country.[21]

Second, the development of packaged software is similar to product innovation in that the sunk R&D costs are high, the marginal costs of production are low, and the rents legally accruing to the software developers or product innovators are easily appropriated without sufficient copyright protection. However, software development still differs considerably from innovation in many ways. For example, in product innovation, imitators are also producers who compete against the innovator in the market. In contrast, software is copied by consumers.[22] As a result, the pricing strategy and equilibrium profit of the product innovator could be very different from those of the software developer. In a recent study, Chen and Png (2000) examine how a software publisher should optimally choose its price and degree of copyright enforcement. Our analysis of packaged software differs from theirs in many aspects. In addition to pricing, we also consider the race between competitors inherent in software development.[23]

Finally, the present study explores the implications of legal institutions on patterns of trade, as opposed to almost all other studies in international trade literature, which are concerned mainly with the effects of the differences in economic factors or trade policies on patterns of trade.[24]

In short, the present model draws on two main building blocks: 1) the effects of contract enforcement and legal protection on the organization of software production: and 2) the interplay between international trade and institutions. It attempts to bridge the "industrial organization" story with the "international trade" one by taking explicitly into account the importance of "history": the country-specific legal regime inherited from the past affects the dynamics of

[21]As discussed by Bolton and Whinston (1993), there are two important theories of vertical integration, one based on incomplete contracts (or transaction costs) and the other on supply assurance. Williamson (1971) is an example of the former theory, and Bolton and Whinston (1993) is an example of the latter. In the business literature, Stuckey and White (1993) present an intensive discussion of when and when not to integrate vertically. They point out that one important reason for vertical integration is that the market is too risky and unreliable and contracts are too costly.

[22]Although selling counterfeit copies is commom in some Asian countries, it is rare in other areas, especially in western countries.

[23]See Chen and Png's (2000) references for some papers in the economic literature on software.

[24]For example, the Ricardian model (considering productivity differences) and the Hechscher-Ohlin model (considering factor endowment differences).

comparative advantages through the action of learning effects or network externalities.

The rest of the paper is organized as follows. The model for a single economy is set up in Section 2. In Section 3, we analyze the customized software and packaged software to describe the equilibrium under autarky. Section 4 analyzes the pattern of specialization and trade in software. Concluding remarks are presented in Section 5.

2. Model of a Closed Economy

There are M computer programmers.[25] The reservation wage is normalized to 1, which a programmer can earn from a nonsoftware job. When engaging in software development, each programmer has to expend additional effort, denoted e.

2.1. Customized Software

There are N firms that are potential users of customized software. One piece of customized software is designed specifically for one firm.[26] Software i ($i = 1, ..., N$) can generate a value equal to u for firm i,[27] but has no value for firm j ($\neq i$). This is assumed in order to capture and emphasize the product specificity of customized software, as opposed to the generic nature of packaged software.[28] One programmer is needed to develop software i. We consider and compare two organizational modes of obtaining software i. First, firm i can hire a programmer to develop it. This is referred to as in-house development or vertical integration. In this case, we

[25]For expositional ease, we simply assume that a programmer can do all the work required for software development (system design, coding, testing, etc.).

[26]According to the definition in a study by PricewaterhouseCoopers, customized software is written to individual customers specifications. In contrast, packaged software consists of all software written for multiple customers and for all types of computer platforms.

[27]We can consider that firm i earns a fixed amount of profit in its industry, but if it uses software i, its profit will increase by u.

[28]Footnote 26 also provides a good justification for this assumption. In a model not specific for software, McLaren (1998) considers that upstream product i is useful to downstream firms i and j ($i \neq j$). Grossman and Helpman (1999) consider both cases in their model. Another implicit assumption in the present model is that firm i benefits from using software i in the amount u, regardless of whether firm j ($i \neq j$) is also using software j. This can be easily justified if the firms are in different industries. This assumption is also made by Bolton and Whinston (1993) and McLaren (1998) to avoid the strategic effect of upstream behavior on downstream competition in order to focus on the upstream-downstream relationship.

call the firm a v-firm. A v-firm's net value from software development is $\pi^v = u - w - v$, where w is the wage paid to the programmer and v is the cost associated with vertical integration.[29]

Second, firm i can offer a contract to a programmer for developing software i. This is referred to as outsourcing or as a contractual relationship. In this case, we call the firm a c-firm. To emphasize the role of contract enforcement, we consider the simplest contracting environment.[30] Assume the programmer can either put in effort e to produce the software with value u or put in zero effort to produce a software without any value. The c-firm can judge the quality of the software. A contract specifies a payment, denoted c, from the c-firm to the programmer after the delivery of the software with value u.[31] Assume no renegotiation.[32] Since payment is required only after the programmer has invested the irreversible effort e to develop the software, this contract inevitably leads to the hold-up problem. After the delivery of the software with value u, the c-firm may take one of two actions. First, it may honor the contract by paying the programmer the contracted amount c. Second, it may breach the contract by paying any amount less than c. Following Anderson and Young (2000), we use a parameter $\alpha \in [0, 1]$ to capture imperfect contract enforcement.[33] That is, for any contract in default, there is probability α that the contract will be enforced by a court. When the court enforces a default contract, the programmer receives c from the c-firm and the c-firm pays an additional penalty $t > 0$, which may include both monetary and moral losses to the society at large. When a default contract is not enforced by the court, the programmer receives nothing and the c-firm pays nothing. The two parameters, α and t, jointly represent the degree of contract enforcement, given which, a

[29]It has been well argued that vertical integration inevitably incurs many types of costs, such as monitoring costs. See Williamson (1971) and Stuckey and White (1993) for some discussions.

[30]See Whang (1992) for a useful discussion of a typical contract for customized software development based on his study of five real-world cases. We ignore those details, since our focus is not on optimal contract design.

[31]There are various types of contracts in reality. As cited by Arora *et al.* (1999), fixed fee contracts account for 58% of all contracts. Our contract is a fixed fee contract.

[32]We could have considered other types of contracts and allowed renegotiation. But this would significantly complicate the analysis, and the qualitative results obtained from our simple contract are unlikely to change. See Whang (1992), Wang *et al.* (1997), and Png and Tao (2000) for some interesting analyses of optimal (limited) contract design for customized software development.

[33]There are many reasons for enforcement imperfection. A technical reason is that the court is not completely sure about the quality of the software. An incentive reason is that it is very costly to verify the truth. In any case, we treat α as exogenous in this paper, as in Anderson and Young (2000).

c-firm's (expected) net value is given by

$$\pi^c = \begin{cases} u - c, & \text{if it honors the contract,} \\ u - \alpha(c+t) & \text{if it breaches the contract.} \end{cases} \tag{1}$$

Hence, in contrast to settings of incomplete contracting, complete contracts are possible in the present model because of possible enforcement.[34] We focus on complete contracts, since we are concerned with the issue of legal enforcement and interested in examining its implications for the equilibrium organizational mode of customized software development.

There is a third potential mode, in which firm i buys software i from the market. This is referred to as arm's length relationship or market. However, it is clear that this is not a viable mode in the present setup, since no programmer will invest his/her effort to produce a very specific product without any certainty of getting a reward.[35] This mode may be considered as an extreme case of the contractual relationship, namely when contract enforcement is completely absent ($\alpha = 0$).

2.2. Packaged Software

There is only one software package and it has multiple potential users.[36] Assume that the potential users are evenly and continuously distributed in the interval $[0, 1]$ in such a way that users with higher i derive lower utilities from using the software. Let $v(i)$ denote user i's utility level when the software's quality is equal to unity. Then $v'(i) < 0$. Furthermore, assume that $v(1) = 0$, $v''(i) < 0$, and if the packaged software's quality is q, then user i's utility is $v(i)q$.

A software developer (SD in short) hires programmers to develop packaged software.[37] The quality of the software is an increasing function of the number of programmers, denoted x, hired by the SD. For convenience, we treat x as a continuous variable. Assume the following properties

[34] Here we rely on legal enforcement to combat the holdup problem. In contrast, Klein *et al.* (1978) emphasize long-term contracts as the market enforcement mechanism.

[35] Both McLaren (1998) and Grossman and Helpman (1999) consider the cases where the product is not too specific to be sold on the market.

[36] The argument can be easily generalized to any fixed number of different software packages.

[37] We ignore the possibility of contracting for packaged software development. This is justified by the fact that many programmers are needed to design the software and it is difficult to organize them in decision making.

for the quality function: $q(x) \geq 0$, $q'(x) > 0$, and $q''(x) < 0$. Without loss of generality (and realism), we can assume that the marginal cost of production (after the first copy) is zero.

There is free entry to software development. However, because of copyright protection, the SD that is the first to develop successful software gets the monopoly rights. Hence, we can view software development as a racing game and refer to the winner as the publisher. When an SD enters, it incurs two types of irreversible investment costs. First, it needs to spend a fixed amount on physical capital, such as computer hardware.[38] This cost is assumed equal to f. Second, it needs to pay the programmers for developing the software. Software development and production exhibits very strong increasing returns to scale.

2.3. Sequence of Moves

There are three stages. At the first stage, each of the N firms determines its organizational mode of customized software development. All SDs determine whether to enter the race to develop packaged software. At the second stage, each c-firm sets the terms of its contract and offers it to a programmer, each v-firm hires a programme, and each SD hires a desired number of programmers. At the final stage, each of the N firms receives its customized software. The publisher sets the price for its packaged software and sells it in the market. All parties are assumed to be risk neutral.

3. Analysis of a Closed Economy

In this section we consider a single economy to derive the equilibrium prior to international trade. Specifically, in Sections 3.1 and 3.2 we analyze the partial equilibrium by treating the wage rate of the programmers as given. Then, in Section 3.3, we derive the general equilibrium when the wage rate clears the labor (programmer) market and free entry leads to zero expected profit for the SDs.

3.1. Customized Software

[38]For more discussion on this type of costs see Siwek and Furchtgott-Roth (1993) and Torrisi (1998).

Since the N firms are not related in any respect except that they rely on the same labor pool to develop software, we can analyze each firm's decision independently for any given $w \geq w_0$, where $w_0 \equiv 1 + e$. As all firms are identical, we need to focus on just one firm.

A. Contracts.

It is useful to define three critical values that are important in determining whether a contract will be honored or breached:

$$m_+ \equiv \alpha t/(1-\alpha), \quad m_- \equiv u/\alpha - t, \quad \text{and} \quad m_0 \equiv u/(u+t).$$

Note that an increase in m_+, or a decrease in m_-, implies that contract enforcement gets stronger. Moreover, the following three conditions are equivalent: $m_+ \geq u$, $m_- \leq u$, and $\alpha \geq m_0$, all indicating stronger enforcement.

We call a contract a *type-H contract* (H stands for honor) if it will be honored by the c-firm, and a contract a *type-B contract* (B stands for breach) if it will be breached by the c-firm. The firm never chooses outsourcing unless $\pi^c \geq 0$, which holds if $c \leq u$ for a type-H contract, and if $c \leq m_-$ for a type-B contract. Suppose $\pi^c \geq 0$ in the following analysis.

Equation (1) immediately implies that the c-firm prefers honoring the contract to breaching the contract if and only if $c \leq m_+$. As a result: (i) if $\alpha \geq m_0$, then the c-firm always honors the contract, and (ii) if $\alpha < m_0$, then the c-firm honors the contract when $c \leq m_+$ and breaches it when $m_+ < c < m_-$.[39] Hence, a programmer can correctly predict whether or not a contract will be honored. The next question is whether the c-firm should offer a type-H or type-B contract and how to determine the contract term c. On the one hand, the c-firm should set c as low as possible so long as the contract is acceptable to a programmer. On the other hand, in order to induce a programmer to accept the contract, the c-firm has to make it at least as attractive as the programmer's outside option, i.e., receiving w for being hired by a v-firm or an SD. Note that by accepting the c-firm's contract, the programmer's expected net return is $c - e$ for a type-H contract, but $\alpha c - e$ for a type-B contract. Hence, the optimal term is $c = w$ for a type-H contract (provided $w \leq u$) and $c = w/\alpha$ for a type-B contract (provided $w \leq u - \alpha t$).

[39]Since $u < m_-$, it is not possible for $c \geq m_-$.

When it is certain or almost certain that a contract will be breached ($\alpha = 0$ or $\alpha \approx 0$), no programmer will accept the contract. To see this, note that in this case both the price of the contract (w/α) and the penalty in the case of contract enforcement $[(w/\alpha) + t]$ are infinitely large, which becomes unrealistic. Accordingly, we rule out this possibility by assuming that there exists $\alpha_{\min} > 0$ such that no contract can be made for $\alpha < \alpha_{\min}$.[40]

Lemma 1 below describes the optimal contract at a given wage rate.

Lemma 1: *Suppose* $w \in [w_0, u)$. *Then for any given* $\alpha \geq \alpha_{\min}$ *and* t,

(i) *if* $\alpha \geq m_0$, *then the c-firm offers a type-H contract with* $c = w$;

(ii) *if* $\alpha < m_0$, *then the c-firm offers a type-H contract with* $c = w$ for $w \leq m_+$, *it offers no contract for* $w > max\{m_+, \alpha m_-\}$, *and it offers a type-B contract with* $c = w/\alpha$ *if and only if* $m_+ < \alpha m_-$ *and* $w \in (m_+, \alpha m_-]$.

Proof: See the Appendix.

The intuition behind the above results is easy to understand by noting that the conditions indicate various degrees of contract enforcement. There are three messages contained in Lemma 1. First, an increase in α raises the occurrence of type-H contracts and reduces that of breach contracts or no contract. Second, in the case of weak contract enforcement ($\alpha < m_0$), as the wage increases, the optimal contract switches from type-H to type-B and then to no contract. Finally, and most importantly, although imperfect contract enforcement may tempt the c-firm to breach its contract, this may actually hurt the firm. To see this, note that since the programmer has an outside option and can infer the type of the contract, he discounts the value of any type-B contract. As a result, if the c-firm is to offer a type-B contract, c must be sufficiently attractive, equal to w/α as opposed to w for a type-H contract.[41] Consequently, a type-B contract results in the expected value $\pi_b^c \equiv u - w - \alpha t$, lower than $\pi_h^c \equiv u - w$, which is the value of a type-H contract. When enforcement is weak (i.e., $\alpha < m_0$) and $w \in (m_+, \alpha m_-]$, should the c-firm have

[40]In fact, we can justify this by imposing a budget constraint for the c-firm, say $c + t \leq B$, which, on assuming B to be sufficiently large, means that the contract is possible only if $\alpha \geq \alpha_{\min}(w)$, where $\alpha_{\min}(w) \equiv w/(B - t) < 1$. In this paper, we choose a fixed $\alpha_{\min}$, which is independent of w and t, to have a clearer exposition. This simplification does not affect the qualitative results in any respect.

[41]This result, due to legal risk, is consistent with that observed by Whang (1992, p. 310) in contracts subject to technical and financial risks. Whang points out that in anticipation of the risks, the developer will demand a high premium for the contract for compensation.

been able to commit not to breach the contract, it would have been able to make a type-H contract and so obtain π_h^c. However, without the ability to commit, the c-firm is forced to offer a type-B contract [Lemma 1(ii)]. This is the negative externality existing in the environment with weak contract enforcement.

B. Optimal Mode of Organization.

If the firm chooses in-house development, then given $w \in [w_0,\ u-v]$, it hires a programmer and receives π^v. Comparing π^v to π^c, we immediately obtain Lemma 2, which describes the optimal choice of organizational mode.

Lemma 2: *Suppose $w < u-v$. A firm chooses vertical integration if and only if* (i) $\alpha < \alpha_{\min}$, *or* (ii) $\alpha \in [\alpha_{\min}, m_0)$ *and* $w > max\{m_+, \alpha m_-\}$, *or* (iii) $\alpha \in [\alpha_{\min}, m_0)$, $m_+ < \alpha m_-, w \in (m_+,$ $\alpha m_-]$ *and* $v < \alpha t$.

Lemma 2 says that for vertical integration to dominate contracts, at least one of the following conditions must be satisfied: contract enforcement is weak; the wage rate is high; and the cost of vertical integration is low. We could also state Lemma 2 in a different way: contracts are preferred to vertical integration when enforcement is very strong (more precisely, $\alpha \geq m_0$), or when it is not so strong but the wage rate is low ($w \in [w_0, m_+]$).

Because of the extra cost associated with vertical integration, contracts are more efficient. However, contracts are not always adopted due to imperfect enforcement, which actually imposes a cost on contracting. This cost does not exist for type-H contracts. As indicated by Lemmas 1 and 2, whenever a type-H contract dominates, vertical integration is never optimal.[42]

C. Contract Enforcement and Optimal Mode of Organization.

To analyze fully the effects of contract enforcement, we first depict recurring Lemmas 1 and 2 graphically in Figure 1. We measure α using the horizontal axis, v using the left-hand-side

[42]Lemma 2 also predicts that vertical integration and contracts may coexist only in a very special case, viz., when $v = \alpha t$. We have two remarks about this. First, Grossman and Helpman (1999) also find that in their model outsourcing (via the market) and vertical integration coexist only in a very special case. Second, in the present model, it is not difficult to see that if v or u is not homogeneous across the firms, then in many cases vertical integration and contracts coexist: vertical integration for some firms and contracts for the others.

vertical axis, and w using the right-hand-side vertical axis. We focus on $v \in (0, t]$, since when $v > t$, contracts always dominate vertical integration regardless of the level of α (provided $\alpha \geq \alpha_{\min}$). The curves $m_+(\alpha)$ and $\alpha m_-(\alpha)$ are plotted against the right vertical axis, while the straight line αt is plotted against the left vertical axis. Two critical points are automatically defined: $\alpha_1 \equiv v/t$, at which the two straight lines v and αt intersect, and $\alpha_2 \equiv w/(w+t)$, at which the two curves w and $m_+(\alpha)$ intersect. Using Lemmas 1 and 2, we observe that as a result of enhancing contract enforcement, the optimal mode of organization changes following a pattern: vertical integration (when $\alpha < \alpha_{\min}$) $\Longrightarrow$ type-B contract (when $\alpha \in [\alpha_{\min}, \alpha_1]$) $\Longrightarrow$ vertical integration (when $\alpha \in (\alpha_1, \alpha_2)$) $\Longrightarrow$ type-H contract (when $\alpha \geq \alpha_2$). Although this pattern is observed here for the special case drawn in Figure 1, it is general, as shown by Proposition 1 below.

<Figure 1 here>

Proposition 1: *Given $v \leq t$ and $w \in [w_0, u - v)$, there exist unique α_1 and α_2 such that the optimal mode of organization for customized software development is vertical integration when $\alpha \in [0, \alpha_{\min})$, a type-B contract when $\alpha \in [\alpha_{\min},\ min\{\alpha_1, \alpha_2\}]$, vertical integration when $\alpha \in (\min\{\alpha_1, \alpha_2\}, \alpha_2)$, and a type-H contract when $\alpha \in [\alpha_2, 1]$.*

Proof: See the Appendix.

It is possible that $\alpha_1 < \alpha_{\min}$, in which case $[\alpha_{\min}, \alpha_1] = \emptyset$, i.e., there exists no type-B contract for all levels of α. It is also possible that $\alpha_1 \geq \alpha_2$, in which case $(\min\{\alpha_1, \alpha_2\}, \alpha_2) = \emptyset$, i.e., there is no vertical integration for intermediate values of α. But in any case, the optimal mode of organization is always vertical integration for sufficiently small α ($\leq \alpha_{\min}$) and always a type-H contract for sufficiently large α ($\alpha \geq \alpha_2$). One may ask why, as α decreases, the optimal mode switches from vertical integration to a type-B contract. The reason is that as α decreases, the cost associated with vertical integration (v) becomes higher than the cost associated with a type-B contract (αt).

Given Proposition 1, we can easily examine the effects of changing v and w on the optimal mode of organization. From Figure 1, an increase in v shifts α_1 to the right. As a result, the range of α for a type-B contract expands and that for vertical integration shrinks. As

v continues to increase, the interval for vertical integration (α_1, α_2) eventually vanishes. The intuition is quite clear: an increase in the cost of vertical integration makes vertical integration less attractive.

An increase in w shifts α_2 to the right in Figure 1. This reduces the range of α for type-H contracts, to the benefit of vertical integration. To understand this, simply consider the point at $\alpha = \alpha_2$. An increase in w reverses the inequality $w \leq m_+$ at this point and so the c-firm can no longer commit to a type-H contract. As explained before (after Lemma 1), the negative externality forces the c-firm to offer a type-B contract or no contract, reducing the c-firm's value. Thus, at $\alpha = \alpha_2$, the optimal choice switches from a type-H contract to vertical integration as a result of wage increases.

From the above analysis, we summarize how the optimal mode of organization responds to changes in the cost of vertical integration and wage rates.

Corollary 1: *Suppose $v \leq t$ and $w \in [w_0, u)$. An increase in v reduces the case for vertical integration and raises the case for type-B contracts. An increase in w reduces the case for type-H contracts and raises the case for vertical integration or type-B contracts.*

Proof: See the Appendix.

3.2. Packaged Software

We now analyze the second-stage competition among the SDs and the third-stage decision of the publisher. The analysis of entry at the first stage is postponed to Section 3.3.

A. User's Behavior and Demand.

Due to piracy, demand for packaged software is very different from demand for ordinary commodities. Suppose the packaged software's quality is q and the price is p. Then, if a potential user i buys the software, he will receive a net benefit equal to $v(i)q - p$. However, the software can be copied at zero cost. When using the pirated software, the probability of being caught is $\mu \in (0, 1)$ and the penalty is equal to p.[43] Hence, the net expected benefit to user

[43]According to Gilman (1992), the Software Publishers Association (SPA) conducts audits, and if illegal software is found, then it "charges a fine equal to the price of the product and then destroys the copy."

i who uses the pirated software is $(1-\mu)v(i)q - \mu p$.[44] Government agents are responsible for detecting piracy and collecting the fines, which become part of the government's revenue. A person's gross and net benefit is zero if he does not use the software.[45] The degree of copyright protection is fully captured by the magnitude of μ. It is clear that in addition to its generic use, piracy is another distinguishing feature of packaged software, as opposed to customized software.

We are now ready to examine the users' decisions on buying, copying and not using packaged software. The decision is affected by the price-adjusted value $v(i)q/p$ and copyright protection μ. First, for user i, buying is preferred to copying if and only if

$$v(i)q/p \geq (1-\mu)/\mu. \tag{2}$$

Second, buying is preferred to not using if and only if $v(i)q/p \geq 1$. Finally, copying is preferred to not using if and only if

$$v(i)q/p \geq \mu/(1-\mu). \tag{3}$$

Combining (2) and (3) yields a necessary condition for the existence of copying: $\mu < \frac{1}{2}$. Lemma 3 below gives the conditions under which copying exists.

Lemma 3: *There is no copying if $\mu \geq \frac{1}{2}$. Copying occurs if and only if $\mu < \frac{1}{2}$ and there exists some i such that the price-adjusted value satisfies condition* (3).

When $\mu \geq \frac{1}{2}$, a further increase in μ will not have any effect on the users' behavior. Without loss of generality, in the rest of the paper, we confine the analysis to $\mu \leq \frac{1}{2}$, letting $\mu = \frac{1}{2}$ capture the case for all $\mu \geq \frac{1}{2}$.[46] We first consider demand for legitimate software at any given $\mu \leq \frac{1}{2}$

Alternatively, we could have considered a more general case where the fine may be different from the price, in which case the combination of μ and the fine would represent the degree of copyright protection.

[44]Therefore, as in Chen and Png (2000), we implicitly assume that the user is caught when he is making the copy and before he actually uses it. That is, he has not been able to derive any utility yet.

[45]Recently, Bakos *et al.* (1999) studied the interesting phenomenon of software sharing and its impact on the publisher's profit. We do not allow legal sharing in the present model.

[46]This critical point, $\frac{1}{2}$, of course depends on many factors, including risk neutrality, equal quality of legitimate software and pirated software and the magnitude of the fines. However, the present study does not emphasize the numerical value of this point, but only its existence. The qualitative results do not depend on whether this value is $\frac{1}{2}$ or $\frac{3}{4}$.

and q. Let $Q \in (0,1)$ denote the marginal user who is indifferent between buying and copying the software, i.e., $v(Q)q/p = (1-\mu)/\mu$, which defines the demand function $Q(p; q, \mu)$. Users with $i \leq Q$ purchase the software, and others either copy or do not use it. Using this definition and the properties of the value function $v(i)$, we can derive the properties of the demand function:[47] $\partial Q/\partial p < 0$, $\partial Q/\partial q > 0$, and $\partial Q/\partial \mu > 0$.

We now turn to copying. Let $Q^c \in [0,1]$ denote the marginal user who is indifferent between copying and not using the software, i.e., $v(Q^c) = \mu/(1-\mu)$, by (3), which defines a function $Q^c(p; q, \mu)$ with the following properties: $\partial Q^c/\partial p < 0$, $\partial Q^c/\partial q > 0$ and $\partial Q^c/\partial \mu < 0$. Recall from (2) and (3) that users with the price-adjusted values greater than or equal to $\mu/(1-\mu)$ but less than $(1-\mu)/\mu$ will use pirated software. Hence, users $i \in (Q, Q^c]$ use pirated software. People with $i > Q^c$ do not use the software (legitimate or pirated). Figure 2 summarizes the above analysis.

<Figure 2 here>

B. Software Developers' Decision on Quality and Price.

Suppose K SDs have entered the packaged software development race at the first stage. We shall work backward by first looking at the third-stage pricing decision. Since the publisher has a monopoly in the market, it faces the demand $Q(p; q(x), \mu)$, supposing its second-stage hiring decision is x. Thus, the publisher's operating profit at the third stage is $Q(p; q(x), \mu)p$. It chooses the price level $p^*(x, w, K, \mu)$ to maximize this profit. At stage two, given wage rate w and number of entrants K, each competing SD chooses its hiring level $x^*(w, K, \mu)$ to maximize its expected profit. Since all SDs are identical, we assume that nature randomly determines the winner, and hence the probability of winning by each SD is equal to $1/K$.[48] As a result, each SD's expected profit at the second stage is given by $Q(p^*; q(x), \mu)p^*/K - wx$. Because the publisher has a monopoly, it is easy to see that the above sequential decisions are equivalent to

[47]For more about other properties of the demand, see (A0) in the Appendix.

[48]In order to focus on the market outcome, we here ignore the details of the racing game by making the assumption on the end result of the game. There is a rich literature on innovation racing games. See Reinganum (1989) for a survey of the early models. The result that the probability of winning by each competitor falls as the number of competitors rises is common in various models of the literature.

a simultaneous decision on x and p, which is obtained from the following maximization problem:

$$\max_{(p,x)} \ \pi(p, x; w, K, \mu) \equiv Q(p; q(x), \mu)p/K - wx. \tag{4}$$

There is a unique interior solution (see proof of Proposition 2 in the Appendix), denoted $x^*(w, K, \mu)$ and $p^*(w, K, \mu)$, that satisfies the following first-order conditions:

$$\begin{aligned} \pi_x &= Q_q q'(x^*)p^*/K - w = 0, \\ \pi_p &= [Q_p p^* + Q(p^*; q(x^*), \mu)]/K = 0, \end{aligned} \tag{5}$$

where and hereafter the subscripts of Q represent derivatives with respect to the corresponding variables. Note that the wage rate affects the equilibrium consumption through quality and price:

$$\partial Q^*/\partial w = Q_q q'(x^*)(\partial x^*/\partial w) + Q_p(\partial p^*/\partial w). \tag{6}$$

Proposition 2: *Given μ and K, an increase in wage rate* (i) *reduces employment in the packaged software development,* (ii) *reduces the packaged software's quality, and* (iii) *lowers the packaged software's price. However, the final consumption for legitimate packaged software increases (is unchanged, decreases) if the quality is higher than (equal to, lower than) unity. More precisely,*

$$\frac{\partial x^*}{\partial w} < 0, \quad \frac{\partial q^*}{\partial w} < 0, \quad \frac{\partial p^*}{\partial w} < 0, \quad and \quad \frac{\partial Q^*}{\partial w} \begin{cases} < 0 & if \ \ q^* < 1, \\ = 0 & if \ \ q^* = 1, \\ > 0 & if \ \ q^* > 1. \end{cases} \tag{7}$$

Proof: See the Appendix.

The first three results of the proposition are easy to understand. As the cost of hiring programmers rises, each SD has to reduce its employment (x^*). As a result, the quality of the software decreases. Lower quality also results in lower demand for the product, and so the publisher's optimal price is reduced accordingly. However, the effect on consumption Q^* is less clear cut. There are two competing forces as shown in (6) where the first term is negative and the second term is positive. On the one hand, for the same price level, lower quality reduces demand and so causes Q^* to decrease (to shift to the left in Figure 2). On the other hand, for a given quality, the lower price generates higher demand and so causes Q^* to increase (to shift to the right in Figure 2). This result is interesting in that it conveys a message that better quality software may or may not be more popular. Since the publisher charges a higher price for high-quality software than for low-quality software, the resulting consumption for the high-quality software may end up lower than that for the low-quality software. The proposition shows that whether Q^* increases or decreases depends on the level of the quality in equilibrium.

3.3. Equilibrium

In this subsection, we first derive the equilibrium w for any given K and then analyze the first-stage equilibrium entry K.

A. Equilibrium Wage Rate w^*.

It is realistic to assume and also more interesting to focus on the case where $M > N$. Programmers can move freely among three different jobs: being employed by the SDs, being employed by the v-firms, or entering into contracts with the c-firms. Let us first examine the demand for programmers in customized software development. So long as $w < u$, all firms strictly benefit from using their respective customized software, regardless of the modes of organization adopted. Therefore, demand for programmers is equal to N. At $w = u$, since the only viable mode of organization is a type-H contract and the firms are indifferent between using and not using customized software, demand for programmers is between 0 and N. Obviously, there is no demand for programmers when $w > u$. Such a demand curve (formed by three straight lines) is depicted on the left panel of Figure 3.

<Figure 3 here>

We now examine the packaged software development and depict the results in the right panel of Figure 3. First, we derive the supply curve S. The supply of programmers is zero for wage rates below w_0. At $w = w_0$, the maximum number of programmers left over for packaged software is equal to $M - N$. Since the programmers are indifferent between being employed and not at this wage rate, the supply is between 0 and $M - N$. For $w_0 < w < u$, all programmers want to be employed and so the supply is equal to the number of programmers left over from customized software, i.e., $M - N$. At $w = u$, the supply is between $M - N$ and M. For $w > u$, the supply equals M.

Second, we turn to the demand for programmers by all SDs. Given K, the total demand, denoted $D(w; K, \mu)$, is equal to $Kx^*(w, K, \mu)$, which is decreasing in w, as shown by (7). Without more specifications of the model, we cannot exactly pin down the position of the demand curve in Figure 3. However, we can draw five possible representative demand curves, labeled $D_i,\ i = 0, ..., 4$. When demand is very low (D_0), we have $x^* = 0$ for $w \geq w_0$. In this case, packaged software does not exist in the economy. The equilibrium wage rate, denoted w^*, is totally determined by customized software and so $w^* = w_0$. At a higher level of demand (D_1), packaged software emerges, but since demand is not sufficiently strong, the wage rate remains the same and employment in customized software and in packaged software development are N and M_1, respectively. As demand continues to increase, the wage rate eventually starts to rise, and it continues to rise (e.g., $w^* = w_2$ when demand is captured by D_2) until it is equal to u. Within this range of demand, $w^* \in [w_0, u]$ and employment in customized software and in packaged software are N and $M - N$, respectively. As demand becomes higher, at D_3, we have $w^* = u$ and employment in the packaged software development reaches M_3, leaving the remaining $M - M_3$ for customized software. As demand increases further, the equilibrium wage rate remains equal to u before the employment in packaged software absorbs all M programmers. After this point, the wage rate has to go up, since the labor supply has reached its maximum. At a demand level such as D_4, the equilibrium wage ($w^* = w_4$) is totally determined by packaged software, and customized software does not exist.

The above graphical analysis clearly indicates the importance of demand $D(w; K, \mu)$ in affecting the equilibrium wage rate and employment. We next show how copyright protection affects such demand.

Let us first examine how a change in K affects the equilibrium wage rate by focusing on the most interesting case in which demand D_2 intersects the vertical segment of the supply curve, $S = M - N$, at w_2. The labor market is in equilibrium when the labor demand by the SDs is equal to the net labor supply,[49] i.e.,

[49]Most of the literature on imperfect competition has a partial equilibrium nature with regard to factor

$$Kx^*(w, K, \mu) = M - N, \tag{8}$$

which determines the equilibrium wage rate as a function of K and μ: $w^*(K, \mu)$. If $D(w; K, \mu)$ shifts upwards (downwards) as K increases, the wage rate will strictly increase (decrease). However, an increase in K has two opposing effects. On the one hand, given each SD's hiring, the demand increases, since there are more SDs to hire programmers. On the other hand, each SD cuts back its hiring, since the expected marginal return is lowered due to the probability of winning becoming lower. It turns out that the effect that dominates depends on the curvature of the quality function $q(x)$. In Lemma 4 below, we show that as a result of K increasing, the wage rate will also increase if the quality function is sufficiently concave (i.e., q'' is very small), but will decrease if the quality function is not sufficiently concave. The basic reason is that when the quality function is very concave, reducing x^* (the latter effect just mentioned) will lead to a big deterioration in quality. Because of this, each SD is more reluctant to reduce its hiring than otherwise. As a result, the latter effect is small, giving rise to an increase in total demand.

Lemma 4: *Suppose the equilibrium wage rate is within* (w_0, u). *Then, as a result of more SDs entering packaged software development, the equilibrium wage rate of the computer programmers increases if and only if the quality function is sufficiently concave. More precisely,*

$$\frac{\partial w^*}{\partial K} \begin{cases} > 0 & \textit{if } \; xq'' + q' < 0, \\ < 0 & \textit{if } \; xq'' + q' > 0. \end{cases}$$

Proof: See the Appendix.

Let us briefly consider all other possible demand curves. It is very clear that given a small change in K, the equilibrium wage rate will not change when demand is at a level like D_0, D_1, or D_3. For very high demand like D_4, the labor market equilibrium condition is the same as (8) except the right-hand side is equal to M. Nevertheless, the analysis for case D_2 completely applies here, and so Lemma 4 holds for this case.

prices. Dixit and Grossman (1986) and Helpman and Krugman (1985, pp. 88 − 95) are two exceptions. Similar to these two, the present paper has a general equilibrium framework.

B. Equilibrium Entry K^*.

Previous analysis has shown that given K, we can derive the equilibria w^*, x^*, and p^*. Using the equilibrium values in π given in (4), we obtain the expected profit as a function of K and μ: $\pi^*(K, \mu)$. Note from (4) that K affects the expected revenue directly and the cost indirectly through its effect on wage rates. The direct effect is clearly negative. However, the indirect effect is ambiguous. Nevertheless, we can show that the direct effect dominates the indirect effect and, as a result (see the Appendix for a proof),[50]

$$\partial\pi^*(K, \mu)/\partial K < 0. \tag{9}$$

Next, we hold K fixed to examine the partial effect of changing μ on π^*. On the one hand, an increase in μ results in higher demand for packaged software, and so the expected profit is also higher. On the other hand, this higher demand for the software translates to higher demand for programmers. Wage rates may or may not rise, depending on the actual position of D. In any case, this wage effect is secondary, and we can show (see the Appendix for a proof) that

$$\partial\pi^*(K, \mu)/\partial\mu > 0. \tag{10}$$

We are now ready to analyze each SD's entry decision. Given μ, because of the monotonicity property (9), there exists an entry if and only if $\pi^*(1, \mu) - f \geq 0$. The equilibrium number of entrants, denoted $K^*(\mu)$, is determined by the free-entry (zero expected profit) condition: $\pi^*(K^*, \mu) - f = 0$. Total differentiation, together with inequalities (9) and (10), yields

$$\mathrm{d}K^*/\mathrm{d}\mu = [\partial\pi^*(K, \mu)/\partial\mu]/\,[-\partial\pi^*(K, \mu)/\partial K] > 0.$$

With the above analysis, we can derive the equilibrium condition of copyright protection for the existence of packaged software in an economy. Denote $\overline{f} \equiv \pi^*(1, \frac{1}{2})$, and assume that

[50]This result is robust to a more general labor market in which labor supply increases as wages rise. Although, as shown in Lemma 4, wages may go down as K increases due to total labor demand reduction, if labor supply is not constant, the resulting reduction in the wage rate will be smaller. Hence the indirect effect becomes less important, reinforcing the inequality of (9).

$f < \overline{f}$. That is, the fixed cost is not too large to deter entry when copyright protection is very strong.

Proposition 3: *Suppose $f < \overline{f}$. Then there exists a unique $\mu_0 \in (0, \frac{1}{2})$ such that a country produces packaged software if and only if $\mu \geq \mu_0$.*

Proof: See the Appendix.

The intuition behind the above result is very simple. Without sufficient copyright protection, demand for packaged software is not guaranteed, and so no one will pay the fixed entry cost to enter the nonprotected industry. Although there are increasing returns to scale, weak copyright protection limits the scale and thus discourages entry.

4. Free Trade

Consider two countries, A and B, that are identical except that B has stronger copyright protection than A. Let μ^A and μ^B represent A and B's copyright protection, respectively. To have a sharper focus, assume $\mu^A < \mu_0 < \mu^B$. Therefore, in autarky, B produces packaged software, but A does not.

There are N (new) firms in each country demanding various types of customized software to increase their respective profits, and there is one potential (new) software package to be developed by SDs in any country.[51] Free trade brings in two new issues. First, customized software that is developed via a contract is internationally tradable. A firm in a country may offer a contract to a programmer in another country for development and delivery of the required software. Second, a country's copyright law protects the packaged software regardless of its country of origin according to WTO's two principles, namely the most-favored-nation treatment and national treatment. Thus, all SDs in both countries join the packaged software development race. The first one to develop the successful software becomes the publisher, who will register in both countries and receive protection (monopoly power) in both markets. As a result, all SDs face the same degree of copyright protection in both markets. Will this eliminate the comparative advantage enjoyed by the SDs in B? Which country will develop and export packaged software?

[51]Simply suppose the software that has been developed by B's publisher in autarky is a word processor (say, Word), and now the new software package is a spreadsheet (say, Excel).

We shall analyze packaged software first. Suppose that the numbers of SDs entering at the first stage are K^A in A and K^B in B. Let $\rho \in (0, 1)$ denote the probability of winning the software development race by each SD in A. There are three types of SDs in B: (i) the existing publisher of the old packaged software under autarky, referred to as publisher O, (ii) those SDs that participated in the old packaged software development race under autarky but failed to become the publisher, referred to as OSDs, and (iii) the new SDs who did not enter under autarky, referred to as NSDs. The NSDs are the same as those SDs in A, and so each one's the probability of winning the software development race is also ρ. However, the OSDs have accumulated additional knowledge in developing packaged software through learning-by-doing. Alternatively, we can consider the existence of network externalities.[52] But for brevity, we confine to the learning effects. Accordingly, assume that the probability of winning by each OSD is Θ (> 1) times as high as by an NSD, i.e., $\Theta\rho$. Publisher O not only was involved in developing the old packaged software, but also has experience in dealing with consumers. Therefore, it has the highest probability of winning. Specifically, we assume that publisher O's probability of winning is O (> 1) times as high as an OSD, i.e., $O\Theta\rho$. Since all SDs of B face the same entry and labor costs, if it is worth it for an NSD to enter, it is also worth it for publisher O and all OSDs.[53] The reverse is not true, however. For ease of exposition, let us now suppose $K^B \geq K^*$, i.e., publisher O and all OSDs do enter.[54] Then, since the sum of all probabilities must be equal to one, we obtain

$$\rho = 1/[K^A + K^B + (\Theta - 1)K^* + (O - 1)\Theta]. \tag{11}$$

In what follows we analyze the role of learning-by-doing by focusing on Θ.[55] From (11), we easily obtain

[52] Network externality is an important feature of software. See Shy and Thisse (1999) for an interesting analysis of software competition and protection in the presence of network externality.

[53] The OSDs and publisher O may save part (if not all) of the fixed cost f if the hardware associated with this cost can be used in the design of the new packaged software. Allowing this possibility will reinforce all the results obtained in this section.

[54] If only some of the OSDs enter, then the expression of (11) and the subsequent analysis can be straightforwardly adjusted, leaving the results unchanged.

[55] The other variable O also captures learning-by-doing, but its effect is simpler than that of Θ. We will discuss this later.

$$\partial\rho/\partial\Theta < 0, \quad \text{but} \quad \partial(\Theta\rho)/\partial\Theta > 0.$$

That is, stronger learning-by-doing lowers the probability of winning by each new entrant but raises the probability of winning by publisher O and each OSD. This will have important implications for the pattern of trade, which we analyze below.

Assuming markets A and B are segmented (as is common in the literature), if an SD wins the software development race at the second stage, then at the third stage it sells the software to market A at price p^A and to market B at price p^B. Thus, the expected profit (excluding entry cost f) for an SD from A is

$$\pi^A(p^A, p^B, x^A; w^A, \rho, \mu^A, \mu^B) \equiv \rho(Q^A p^A + Q^B p^B) - w^A x^A, \tag{12}$$

where $Q^i \equiv Q(p^i, q(x^A), \mu^i)$, $i = A, B$, is the demand for the legitimate software in market i. Given w^A, the SD chooses (p^A, p^B, x^A) to maximize this expected profit. Note that once ρ is given, π^A does not depend on the number of entrants in either country. However, the number of entrants does affect the value of ρ, as shown by (11).

In B, the expected profit for an NSD, denoted $\pi^{BN}(p^A, p^B, x^B; w^B, \rho, \mu^A, \mu^B)$, can be obtained similarly to (12), with w^A replaced by w^B and x^A by x^B. The expected profit for each OSD, denoted π^{BO}, is the same as π^{BN} except ρ is replaced by $\Theta\rho$. The expected profit for publisher O, denoted π^{BP}, is the same as π^{BN} except ρ is replaced by $O\Theta\rho$. Given w^B, each SD in B chooses (p^A, p^B, x^B) to maximize its expected profit.

Following the procedure of analyzing packaged software under autarky, we can also examine the equilibrium entry for any given (μ^A, μ^B) when trade is open. However, in the rest of this section we focus on a specific case where learning-by-doing is sufficiently strong. We will show that for sufficiently large Θ, there is no SD from A entering the software development. To prove this result, let us first examine the optimal decision of an SD from A. Maximizing the expected profit (12) gives the Kuhn-Tucker first-order conditions:

$$\begin{aligned}&\partial\pi^A/\partial x^A = \rho(Q_q^A q' p^A + Q_q^B q' p^B) - w^A \le 0 \ \text{ and } \ (\partial\pi^A/\partial x^A)x^A = 0,\\ &\partial\pi^A/\partial p^A = \rho(Q^A + Q_p^A p^A) \le 0 \ \text{ and } \ (\partial\pi^A/\partial p^A)p^A = 0,\\ &\partial\pi^A/\partial p^B = \rho(Q^B + Q_p^B p^B) \le 0 \ \text{ and } \ (\partial\pi^A/\partial p^B)p^B = 0.\end{aligned} \tag{13}$$

Since it is never optimal to set either p^A or p^B equal to zero, the second and third conditions of (13) are equalities, which define the optimal $p^A(x^A, \mu^A, \mu^B)$ and $p^B(x^A, \mu^A, \mu^B)$. Substituting these two functions into the first condition of (13), we then observe that the resulting $(Q_q^A q' p^A + Q_q^B q' p^B)$ is independent of ρ. Therefore, for sufficiently strong learning-by-doing, which implies a sufficiently small ρ, $\partial\pi^A/\partial x^A < 0$, and so $x^A = 0$. As a result, for sufficiently large Θ, the expected profit is negative, $\pi^A - f < 0$. No SD from A will enter the race.

However, in the absence of learning-by-doing ($\Theta = \Theta_p = 1$), it is easily seen that at least some SDs from country A will enter. This is because now all SDs (from A or B) are facing the same probability of winning ($1/K$) but the winner gets two markets. Hence, for the same K, an SD of A now in trade gets a higher expected profit than that obtained by an SD in B under autarky. Since the SDs of country B enter under autarky, these SDs of A have more incentive to enter under free trade.

The above simple analysis provides the intuition for the following result: when there is very weak learning-by-doing, some SDs from A enter the software development race, but when there is sufficiently strong learning-by-doing, no SDs from A will enter. Proposition 4 below goes further to show that there exists a threshold of learning-by-doing that determines whether any SD from country A will enter the race.

Proposition 4: *Suppose* $\mu^A < \mu_0 \le \mu^B$. (i) *There exists* $\Theta_A > 1$ *such that no SD of country A will enter the packaged software development iff* $\Theta > \Theta_A$. (ii) *When* $\Theta > \Theta_A$, *only country B develops and therefore exports packaged software, and only country A exports customized software.*

Proof: See the Appendix.

This proposition predicts the pattern of specialization and trade in software development in the case of strong learning-by-doing. The intuition is simple. Because of learning-by-doing, publisher O and the OSDs in B can maintain their comparative advantages over the new entrants from A. When these comparative advantages are sufficiently strong, the chance for the new entrants to win the software development is so small that it is not worth it to them to pay the fixed entry fee.[56] As a result, B is the only country that will develop and thus export packaged software.[57]

Since the two countries differ only in their respective copyright protection, the above pattern of trade must stem from such a difference.[58] To appreciate the importance of cross-country differences in copyright protection, let us digress for a moment by supposing $\mu^A = \mu^B$. First, suppose $\mu^A = \mu^B \geq \mu_0$. Then, under autarky, there are equal numbers (K^*) of SDs entering in each country to develop the old packaged software, and eventually there are two publishers, one in each country. When trade is open, the two publishers definitely enter the new packaged software development race, because they have equal probability to win, this probability is higher than that of any other SD, and they face the same labor costs in their countries.[59] Because of the symmetry, if it is worthwhile for an OSD in A to enter, it is also worthwhile for an OSD in B to enter. The same logic applies to the NSDs. Thus, the probability that the winner is from A is equal to the probability that it is from B. Consequently, which country develops and exports the new packaged software is completely indeterminate. Second, suppose $\mu^A = \mu^B < \mu_0$. Then there is no SD entering the packaged software development in either country under autarky. However, when countries are open for trade, since the potential publisher faces two markets, some SDs in both countries may enter the race. Again, in this case, the probability that the winner is from country A (B) is equal to $\frac{1}{2}$.

[56]We expect that network externalities can also play a similar role as learning-by-doing. With network externalities, publisher O has advantages over other SDs since the publisher enjoys a larger customer group, *ceteris paribus.*

[57]Publisher O continues to produce and now starts to export the old packaged software if demand still exists.

[58]Feenstra *et al.* (1999) examine empirically the implications of differences in business organization (i.e., vertical integration in Japan and Korea vs. independent firms in Taiwan) for exports. In the present paper, we go one step further and show that a difference in legal institutions between countries leads to a difference in business organization, which shapes the pattern of international trade.

[59]As for trade in the old packaged software, the trade pattern depends on the nature of the software. If A's software and B's software are sufficiently different that both of them can be protected under the same copyright law, then A will export its packaged software to B and B will export its packaged software to A. Otherwise, country A's (B's) copyright law will protect its own software by banning sales of country B's (A's) software. In this latter case, no trade occurs.

In summary, if the countries have the same copyright protection, although they may still have trade in packaged software, the pattern of trade is indeterminate. In contrast, Proposition 4 says that when $\mu^A < \mu_0 \leq \mu^B$ and learning-by-doing is sufficiently strong, the pattern of trade is clearly determined.[60]

Turning to customized software, we realize that it is simpler to assume away learning-by-doing. However, it would not be difficult to see later that including learning-by-doing will only reenforce our result. If all firms choose to have in-house development under free trade, the opportunity for trade does not affect the customized software development at all. If, however, it is optimal to rely on a contract for delivery of customized software, then they will give contracts to the programmers in the country that has the lower wage rate. Because of the demand for programmers by the SDs in B, the wage rate in B is not lower than that in A. Hence, it is possible that the firms in B offer contracts to programmers in A, resulting in a customized software export from A to B.

Let us examine whether the wage rate in B is strictly higher than that in A. For sufficiently strong learning-by-doing ($\Theta > \Theta_A$), there is no new SD (either from A or B) entering the race. Note that in this case, the OSDs face no more competition in software development than under autarky but have one more market in which to sell their product. This seems to suggest that the incentive for them to enter the race is higher when there is trade than under autarky and the total labor demand in B is also larger when there is trade than under autarky. However, this is not necessarily the consequence. Let us look at the entry issue first. Suppose that publisher O and all OSDs enter. Then $\rho = 1/[\Theta(K^* - 1 + O)]$ and so the probability of winning by each OSD is $1/(K^* - 1 + O)$ and that by publisher O is $O/(K^* - 1 + O)$. Note that the probability of winning by each OSD is smaller than that under autarky, which is $1/K^*$. Moreover, if publisher O's learning-by-doing (namely O) is sufficiently large, the winning probability of each OSD is close to zero. In this extreme case, only publisher O will enter the race, and so the labor demand for the packaged software development need not be higher when there is trade than

[60]This explanation for the pattern of trade is in line with the observation of Arora *et al.* (1999). Based on their study of the Indian software industry, Arora *et al.* conjecture that both the weak intellectual property rights and lack of experience are important culprits for the failure of the Indian firms to develop successful packaged software.

under autarky.

Finally, let us view the consequences of trade in two sequential steps, by first allowing trade in packaged software only and then allowing trade in customized software also. Since there is no entry into the packaged software development in A but there is in B, if trade in customized software is ruled out, then the wage rate in A is equal to the minimum (w_0), but that in B is not lower than this level. According to Corollary 1, the B firms are more likely to adopt vertical integration than the A firms. Nonetheless, the B firms will offer contracts to A's programmers if and only if (i) the wage rate in B is higher than that in A and (ii) it is optimal for the B firms to adopt contracts. A direct implication of this is that in *equilibrium* the wage rate in A cannot be greater than that in B. If the B firms do make contract offers to A's programmers, demand for programmers in B decreases and that in A increases, leading to a possible wage rate drop in B and a rise in A. Thus, the wage gap may be reduced. Since with trade the wage rate in A cannot be lower than that under autarky, by Corollary 1 we immediately know that trade may reduce the cases where contracts are adopted for customized software development in A. This can occur only if A exports the "contracted" customized software. The above analysis is summarized below.

Corollary 2: *Suppose $\Theta > \Theta_A$.* (i) *In equilibrium, $w^B \geq w^A$. Trade reduces the gap between the computer programmers' wage rate in B and that in A.* (ii) *Trade reduces the cases where contracts are adopted for customized software development in country A.*

5. Conclusion

We have developed a model to analyze software development and trade. We emphasize the effects of contract enforcement on the organizational mode of customized software development and the effects of copyright protection on the pattern of trade for both packaged software and customized software.

Software has several distinguishing features compared with common commodities like automobiles and therefore deserves special attention. In this paper, we have emphasized the product specificity of customized software, which results in holdups, and the low (or zero) marginal cost of reproducing packaged software, which leads to piracy. Because of these features and the resulting problems, legal environments are very crucial in shaping this industry's development and trade.

As an extension to check the robustness of the results obtained in this paper and to derive new results, we could introduce some other features of software into the present model. One example is the network externality in the packaged software. When using the software, a user's utility, $v(i)$, depends on the number of people who are also using it (legally or illegally). This externality will affect the publisher's pricing strategy in an interesting way. Inevitably, the investment and entry decision will also be affected. However, it is conceivable that the results obtained in this paper about the packaged software are unlikely to be altered qualitatively.

Another direction of an extension is to explore more closely the copyright spillover effect from packaged software to customized software. In the present model, the cross-product effect is realized through changes in the wage rate. We could also consider the case when productivity in customized software development may be enhanced by the existence and the quality of packaged software.

References

Anderson, J.E. and L. Young, 2000, "Trade and contract enforcement," Mimeo, Boston College.

Arora, A. and J. Asundi, 1999, "Quality certification and the economics of contract software development: A study of the Indian software service companies," NBER Working Paper 7260.

Arora, A., V.S. Arunachalam, and J. Asundi, 1999, "The Indian software service industry," Mimeo, The H. John Heinz III School of Public Policy and Management.

Baba, Y., S. Takai, and Y. Mizuta, 1995, "The Japanese software industry: The 'hub structure' approach," *Research Policy*, 24, 473–486.

Bakos, Y., R. Brynjolfsson, and D. Lichtman, 1999, "Shared information goods," *Journal of Law and Economics,* 42(1), 117–155.

Besen, S.M. and L.J. Raskind, 1991, "An introduction to the law and economics of intellectual property," *Journal of Economic Perspectives*, 5(1), 3–27.

Bolton, P. and M.D. Whinston, 1993, "Incomplete contracts, vertical integration, and supply assurance," *Review of Economics Studies*, 60, 121–148.

BSA, 1998, "The contribution of the packaged software industry to the European economies," A study conducted by PricewaterhouseCoopers, commissioned by the Business Software Alliance (http://www.bsa.org).

BSA, 1999a, "1998 Global Software Piracy Report," A study conducted by International Planning and Research Corporation for the Business Software Alliance (http://www.bsa.org).

BSA, 1999b, "Forecasting a robust future: An economic study of the U.S. software industry," A study conducted by PricewaterhouseCoopers, commissioned by the Business Software Alliance (http://www.bsa.org).

Che, Y.-K. and D. Hausch, 1999, "Cooperative investments and the value of contracting," *American Economic Review*, 89(1), 125–147.

Chen, Y. and I. Png, 2000, "Software pricing and copyright: Enforcement against end-users," Mimeo, National University of Singapore.

Correa, C.M., 1996, "Strategies for software exports from developing countries," *World Development*, 24(1), 171–182.

Dixit, A. and G. Grossman, 1986, "Targeted export promotion with several oligopolistic industries," *Journal of International Economics*, 21, 233–249.

Feenstra, R., T.-H. Yang, and G. Hamilton, 1999, "Business groups and product variety in trade: Evidence from South Korea, Taiwan and Japan," *Journal of International Economics*, 48, 71–100.

Fortune, 1994, "Competitiveness: How U.S. companies stack up now," April 18.

Gilman, J., 1992, "When the SPA comes, a-knockin" *Computerworld*, (December 7, 1992), 108.

Grossman, G. and E. Helpman, 1999, "Incomplete contracts and industrial organization," Mimeo, Princeton University.

Helpman, E. and P. Krugman, 1985, *Market Structure and Foreign Trade,* The MIT Press (Cambridge, Mass).

Klein, B., R. Crawford, and A. Alchian, 1978, "Vertical integration, appropriable rents, and the competitive contracting process," *Journal of Law and Economics*, 21(2), 297–326.

McLaren, J., 1998, "'Globalization' and vertical structure," *American Economic Review* (forthcoming).

Png, I. and Z. Tao, 2000, "Installment payments in contracts for systems development," Mimeo, National University of Singapore.

Reinganum, J.F., 1989, "The timing of innovation: Research, development and diffusion," Chapter 14 in R. Schmalensee and R.D Willig (eds.), *Handbook of Industrial Economics*, vol. 1, North-Holland (Amsterdam), 849–908.

Shy, O. and J.-F. Thisse, 1999, "A strategic approach to software protection," *Journal of Economics and Management Strategy*, 8(2), 163–190.

Siwek, S.E. and H.W. Furchtgott-Roth, 1993, *International Trade in Computer Software*, Quorum Books (London).

Stuckey, J. and D. White, 1993, "When and when *not* to vertically integrate," *Sloan Management Review*, Spring, 71–83.

Tirole, J., 1999, "Incomplete contracts: Where do we stand?" *Econometrica*, 67(4), 741–781.

Torrisi, S., 1998, *Industrial Organisation and Innovation: An International Study of the Software Industry*, Edward Elgar (Cheltenham, UK).

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

Whang, S., 1992, "Contracting for software development," *Management Science*, 38(3), 307–324.

Williamson, O., 1971, "The vertical integration of production: Market failure considerations," *American Economic Review*, LXI(2), 112–123.

Zhang, J.X. and Y. Wang, 1995, *The Emerging Market of China's Computer Industry*, Quorum Books (London).

Zhang, J.Z. and T. Zhu, 2000, "Verifiability, incomplete contracts and dispute resolution," *European Journal of Law and Economics*, 9(3), 281–290.

Customized Software

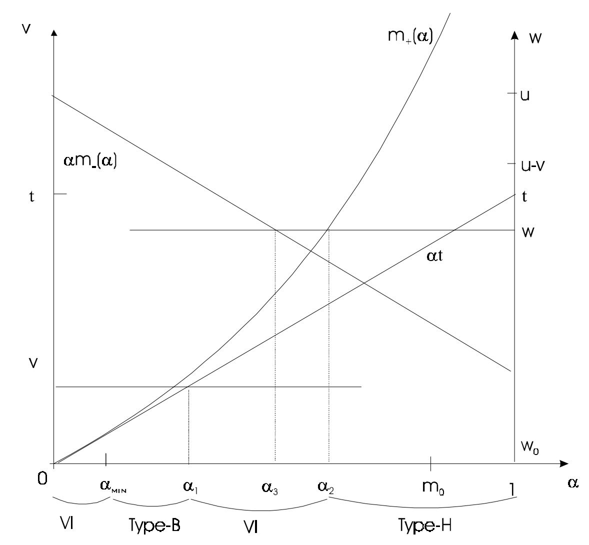

Figure 1:

Packaged Software

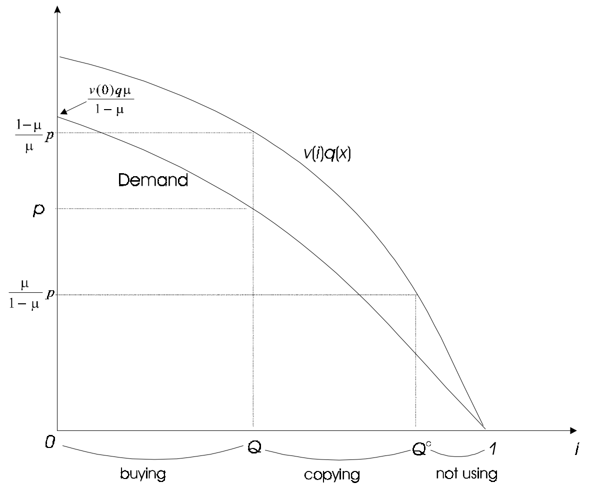

Figure 2:

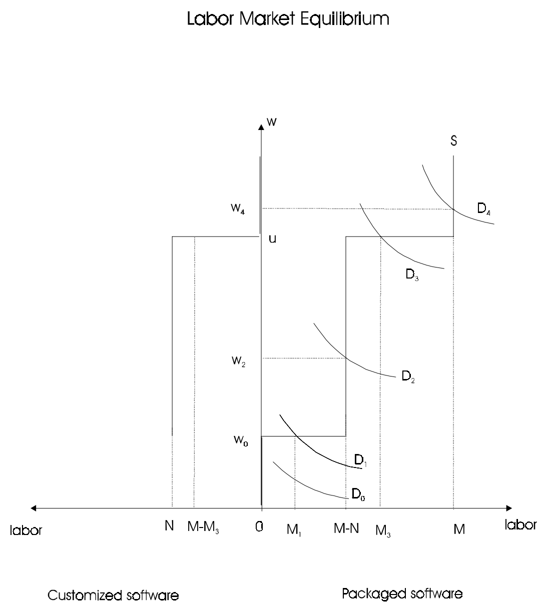
Labor Market Equilibrium
w
S
w4
D4
u
D3
w2
D2
w0
D1
D0
labor
N
M-M3
0
M1
M-N
M3
M
labor
Customized software
Packaged software

Figure 3: